FICSR: Feedback-based $I$n$C$on$S$istency Resolution for Query Processing on Misaligned Data Sources

ABSTRACT

An important challenge when integrating knowledge is that knowledge from different sources may often be conflicting. In this paper, we present a novel feedback-based approach ($FICSR$[1]) to inconsistency resolution when integrating data sources. We highlight that conflict-resolution methods, which aim to achieve integration and cleaning before the query processing, can be costly and, many times, ineffective. Instead, we propose a ranked interpretation of the data which enables users to observe and resolve conflicts by considering the context provided by the queries. We show both theoretically and experimentally that (a) system feedback regarding the conflicts in the most likely candidate results can inform the user regarding data and relationship-constraints critical to a given query and (b) user feedback regarding the ranked interpretation can be exploited to inform the system regarding user's domain knowledge as applicable within the context of a given query. To support such bi-directional ($data \overset{informs}{\longleftrightarrow} user$) feedback, we develop data structures and novel algorithms to enable efficient off-line conflict/agreement analysis of the data as well as on-line query processing, candidate result enumeration, and validity analysis.

1. INTRODUCTION

Query processing on data from different sources includes *matching/alignment* and *integration* tasks. The problem of automated matching, which takes two data or schemas as input and produces a mapping (or alignment) between elements, has been investigated in scientific, business, and web data integration [35] contexts. For example, [29] uses structures (schema graphs) for matching. [10] explores matching within the context of hierarchical data and metadata with fuzzy, many-to-many mappings. Clio [28], LSD [11], SKAT [30], Cupid [25], and DIKE [33] are other matching systems. In particular, [21] proposes a language that allows users to specify alternative semantics for mapping tables between sources and show that a constraint-based treatment of mappings leads to efficient mechanisms for inferring new mappings. [5] proposes to use DTDs and source-to-target dependencies (STDs) to eliminate inconsistent data translation from one schema to another.

In many cases, however, the alignment across the available data sources is not perfect. Two sources may not agree on the existence of a particular data entity or its relationships with other entities. For example when a scientist is trying to work under multiple, conflicting assumptions or hypotheses, disagreements in the integrated knowledge or interpretations are unavoidable. For example, Figure 1 shows two alternative mammal taxonomies that are available to a scientist, each representing a different view of how the categorization should be performed.

[1]Pronounced as “fixer”.

1.1 Misalignments and Conflicts

In this paper, we *develop a novel methodology for dealing with imperfectly aligned data and algorithms to assist users to pose questions and explore alternative answers when the alignment of data is not perfect and contains conflicts*.

A major challenge when dealing with conflicts is that, in many cases, resolution is an ill-defined problem: there may be multiple ways to resolve conflicts and the appropriate conflict resolution strategy may be user-, and query context-dependent. In many cases (such as the scientist working under potentially conflicting interpretations as in Figure **??**), overly-eager conflict resolution may be detrimental to the effective use of the available knowledge. Thus, we argue that since conflicts between sources may highlight different interpretations of the base facts, it may be important to keep the original views of the data, even though there might be conflicts. Especially, in information mashup scenarios [**?**], where quick integration is more desirable than complete and clean integration, striving for full conflict-resolution may be counter-productive. Furthermore, given the many alternatives, identification of consistent models or result enumeration may become extremely expensive.

In this paper, instead of trying to achieve fully-consistent integration, we aim to use query instances to provide contexts in which conflicts are resolved. Like us, Piazza [17] and HepToX [6] also recognize that it is unrealistic to expect an independent data source entering information exchange to agree to a global mediated schema or to perform heavyweight operations to map its schema to every other schema in the group. Piazza presents a mediation language for mapping both the domain and document structures and focuses on *certain answers* that hold for every consistent instance, while we consider conflicts explicitly and only aim to resolve them within the context of a given query. HepToX, on the other hand, focuses on automated mapping rule generation, without explicitly considering conflicts. **[**FROM SELCUK: **other work...widom]**

1.2 Alternative Interpretations of Data with Conflicts

As exemplified above [?], traditionally, a consistent interpretation (or *model*) of the integrated data is defined as a maximal, self-consistent subset of the data.

DEFINITION 1.1 (MODEL-BASED INTERPRETATION). *A* model *(or* model-based interpretation*) of a given knowledge base*

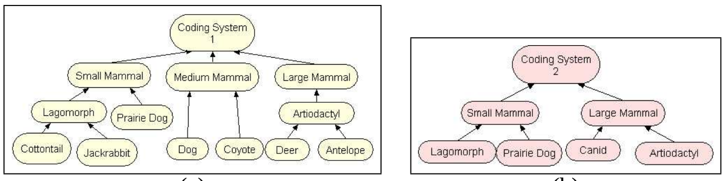


Figure 1: Two different IS-A hierarchies (taxonomies) available to a scientist represent different views on what the correct mammal categorization should be

D is a subset D' of the knowledge base ($D' \subseteq D$) such that there exist no other consistent set D'', where $D' \subset D'' \subseteq D$.

Thus, any restoration of consistency in the integrated knowledge base by selecting a consistent model could lead to loss information. Furthermore, in many cases, the user may not have enough information (domain knowledge) to select an appropriate model among all the alternatives implied by D.

In this paper, instead of characterizing the user's interpretation as a maximally consistent portion of the available information that she commits as being certain, we argue that a more flexible definition of interpretation, *which captures the likelihood that a given asserted statement about the data can be considered as holding*, may be more suitable in these scenarios.

DEFINITION 1.2 (RANKED INTERPRETATION). *Let D be the data and $\mathcal{S}$ be a set of statements (i.e., propositions) on the data. Then, a total ranking of statements in $\mathcal{S}$ is a ranked interpretation of the data D.*

Note that the model-based interpretation of the data is a special case of ranked interpretation, where the rank of all certainly (based on the model) true statements is better than that of the rank of all certainly false statements.

Data alignment is a subjective process in that the mappings capture the user's interpretation of the data sources and application requirements. These aspects are captured by *subjective* ranked interpretations (i.e., $\preceq_{D,U}$, implicitly capturing the user, U's domain knowledge or preferences), as opposed to *objective* interpretations (i.e., $\leq_D$, measuring the degree of agreement of the different data sources about a given statement).

DESIDERATUM 1 (OBJECTIVE-SUBJECTIVE CORRESPONDENCE). *It is preferred that, for all $S_1, S_2 \in \mathcal{S}$, it holds that*

$$(S_1 \preceq_{D,U} S_2) \quad \longleftrightarrow \quad (S_1 \leq_D S_2)\,.$$

Query processing over data with conflicts require the gap between objective and subjective interpretations of the data with conflicts to be bridged.

1.3 The Use of Feedback in Information Retrieval

In this paper, we first note that, in the area of information retrieval (IR [?]), researchers face a similar challenge of objective-subjective gap: when processing an information retrieval query,

- which features of the data are relevant (and how much so) for the user's query may not be known in advance, and
- the number of candidate matches in the database can be potentially very large.

In the IR context, these challenges are dealt effectively through relevance feedback cycles (Figure 2). The relevance feedback process enables the information retrieval system to learn the user's interests

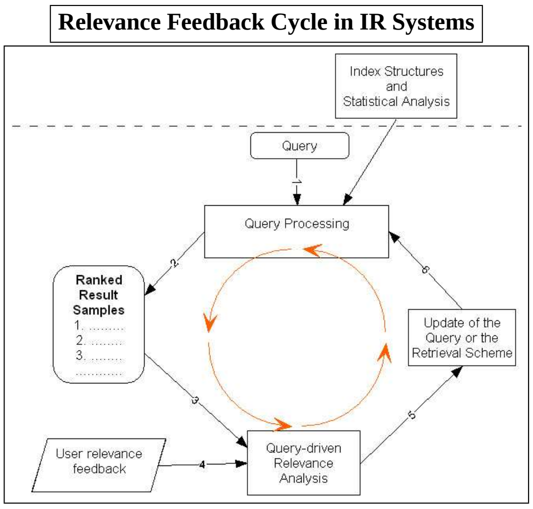

Relevance Feedback Cycle in IR Systems
Index Structures and Statistical Analysis
Query
1
Query Processing
2
Ranked Result Samples
1. ....
2. ....
3. ....
....
6
Update of the Query or the Retrieval Scheme
3
User relevance feedback
4
Query-driven Relevance Analysis
5

Figure 2: Overview of the relevance feedback process commonly used in information retrieval when the available data has alternative (user- and query-dependent) interpretations

and focus to a suitable feature set through a query-driven, transparent, and iterative process. In particular, (1) given a query, using the available index structures, the system (2) identifies an initial set of candidate results. Since the number of candidates can be large, the system presents a small number of samples to the user. (3) This initial sample and (4) user's relevance/irrelevance inputs are used for (5) learning user's interests (in terms of relevant features) and this information is used for (6) updating the user query or the retrieval/ranking scheme. Steps 2-5 are then repeated until user is satisfied with the ranked result samples returned by the system.

We propose to benefit from a similar feedback-based approach in the context of query processing in the presence of alternative interpretations. The system will rely on the *objective-to-subjective* implication ($\longleftarrow$) to inform the user about the more likely (i.e., highest source agreement) interpretations for the given data. Then, the *subjective-to-objective* implication ($\longrightarrow$) will be leveraged to inform the system about the user's own interpretation.

DESIDERATUM 2 (CONTEXT-INFORMED INTERPRETATION). *Since the feedback process can be computationally costly and since the user may have neither the need or nor sufficient domain knowledge to interpret the entire data, instead of considering all possible statements, it is preferable to focus on only those statements relevant within the context specified by a user query.*

1.4 Proposed Approach: Feedback-driven Query Processing and Conflict Resolution in the Presence of Imperfectly Aligned Data

In this paper, we develop data structures (and algorithms to enable feedback-based conflict resolution during query processing on imperfectly aligned and integrated data. Figure 3 illustrates the overall process.

1.4.1 Alignment Conflicts and Objective Agreement

First, an initial alignment between the input data is obtained through semi-automated techniques, such as [**?**]. The result of the alignment is a set of mapping rules, such as those described in [?]. These rules along with the integrated data are then represented in the form of a set of *constraints*. In this paper, we classify the constraints into two major classes: (a) *relationship constraints* describe

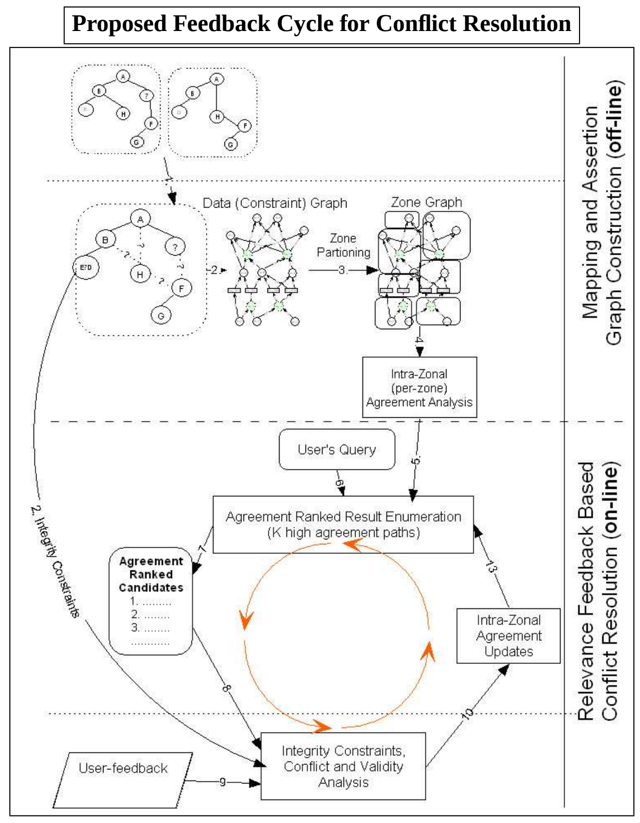

Proposed Feedback Cycle for Conflict Resolution
Mapping and Assertion Graph Construction (off-line)
Relevance Feedback Based Conflict Resolution (on-line)
Data (Constraint) Graph
Zone Graph
Zone Partioning
Intra-Zonal (per-zone) Agreement Analysis
User's Query
Agreement Ranked Result Enumeration (K high agreement paths)
Agreement Ranked Candidates
Intra-Zonal Agreement Updates
Integrity Constraints, Conflict and Validity Analysis
User-feedback
2. Integrity Constraints

Figure 3: Overview of the query-driven feedback-based conflict resolution process to deal with imperfectly aligned data

how the individual data objects/entities relate to each other, while (b) *integrity constraints* describe the general rules these data entities have to obey.

Unless the constraints are conflict free, there will be multiple solution *models*. Consequently, "*the ratio of the models (i.e., alternative interpretations) of the data where a particular data fragment is consistent with the rest*" can be used as a measure of *agreement* of the data sources on this data fragment. Therefore, informing the user regarding the objective ranking involves identifying and enumerating results with high agreements.

1.4.2 Query Processing and Relevance Feedback-based Conflict Resolution

The agreement-based ranking task can be computationally complex if the system would need to enumerate all alternative models (in Section 5.1, we show that the problem is NP-complete even in highly specialized cases). Therefore, a particular challenge in query processing in the presence of conflicts is to postpone the computation of complete solution models until absolutely necessary. In order to deal with the cost of the agreement value computation in the presence of conflicts, we divide the task into three stages:

Off-line Analysis: We first represent the relationship constraints in the form a constraint graph which enables off-line *conflict/agreement* analysis as well as ranked candidate result enumeration. In order to compute the agreement values efficiently, we further partition the data graph into small-sized constraint *zones*, each consisting of an mutually-dependent set of relationship constraints. The agreement values are then computed for each zone separately and combined efficiently for paths that span multiple zones during query processing.

Candidate Enumeration and Ranking: Given a query, the system first identifies and ranks an initial subset of matches, using these *zonal* agreement values. Once presented with a ranked set of the results, the user can pick and choose between available results.

Integrity Constraints and Feedback: The remaining complex integrity constraints (such as "no-cycles are allowed in data") are used for verifying the *validity* of the ranked interpretation obtained through zonal agreement values. In particular, through the analysis of integrity constraints within the context provided by the candidate result sets as well as user feedback, candidate results as well as integrity constraints themselves are assigned *validity values*. Once these validity values computed are propagated back to the objective agreement values (completing the feedback cycle), the user can be provided with a new subset of *ranked* results.

1.5 Contributions of the Paper

The proposed system brings together various innovative techniques to deal with the computational complexity and the ill-defined nature of the conflict resolution problem:

- First of all, we propose a novel, feedback-driven approach to query processing in the presence of conflicts. The feedback process relies on a novel ranked interpretation of the data. The objective-subjective correspondence of the ranked interpretation enables the user to explore the available data within the context of a query and be informed regarding data and relationship-constraints critical to a given query before providing feedback.

- We provide data structures and algorithms that enable efficient off-line analysis of the data for agreement analysis and online query processing, candidate result enumeration, and result pruning and compatibility analysis. We represent data in the form of relationship and integrity constraints (Sections 2 through 4):
  - the relationship constraints lend themselves to efficient partitioning into *independent* constraint sets (called zones, Section 4.1). The small sizes of the zones enable efficient off-line agreement (Section 5) and their independent nature enables efficient on-line composition (Section 6).
  - the top-k nature of the on-line candidate result enumeration process lets the user focus on high-agreement parts of the data, up on demand (also in Section 6).
  - the cost of the *integrity* analysis and feedback stage is kept low through the small size of candidates that need to be verified as well as the use of the query-context that sets the scope of the compatibility checks (Section 7).

[FROM MLS: **..incremental update]**

2. DATA REPRESENTATION

Since our goal is to maximize the applicability of the algorithms to diverse application domains, we keep assumptions from the data low and simply take that the data, D, can be represented in the form of an entity-relationship graph (G) and associated integrity constraints (IC).

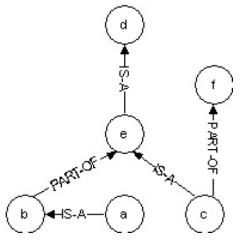


Figure 4: Basic data graph example

2.1 Data Relationship Graphs

A basic data (relationship) graph captures the set of rules that describe the objects/entities in a data source and their inter-relationships.

DEFINITION 2.1 (BASIC DATA GRAPH). *A basic data graph, $G(V, E)$ is a node and edge labeled directed graph, where*

- *each node, $v \in V$, corresponds to an entity (or data object) and*
- *each edge, $e \in E$, corresponds to a relationship between two entities and labeled with a relationship name.*

Each relationship name has a corresponding arity constraint `1-1` *(one-to-one),* `1-N` *(one-to-many),* `N-1` *(many-to-one), or* `M-N` *(many-to-many).* ◇

In a sense, each node in the graph *asserts* the existence of a distinct object and each directed edge is a constraint which *asserts* the existence of a relationship, (such as IS-A, PART-OF, WORKS-AT) between two objects. For example, the IS-A relationships between objects and their immediate ancestors in a taxonomy are constrained to be `N-1` (leading to tree-structured class hierarchies [?]). Figure **??** presents an example basic data graph.

A *data path* is, then, a sequence of relationship edges on this graph describing how two entities are related.

DEFINITION 2.2 (DATA PATH). *A data path, dp, on the data graph, $G(V, E)$, is a sequence of edges, $dp = \langle e_1, e_2, \ldots, e_{length(dp)} \rangle$, where*

$$\forall_{i<length(dp)} \;\; dest(e_i) = source(e_{i+1}), \;\; \text{and}$$

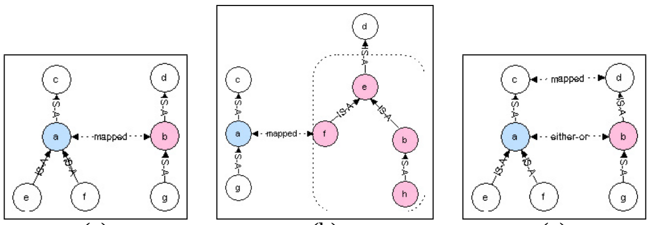
where $source(dp) = source(e_1)$ and $dest(dp) = dest(e_{length(dp)})$ are both data nodes (corresponding to distinct objects/entities).

The set of data paths on the example data graph in Figure 4 includes $a \overset{IS-A}{\rightsquigarrow} b \overset{PART-OF}{\rightsquigarrow} e$ and $c \overset{IS-A}{\rightsquigarrow} e \overset{IS-A}{\rightsquigarrow} d$[2]. Since they describe the relationships between entities, in this paper, we take data paths as the basic *statements of interest.* Thus, in query processing, we mainly focus on queries about data paths between given two entities. In fact, such data path based treatment of queries is common in richly structured data, such as OO [?] and XML [?].

2.2 Integrity Constraints

The data graph described above captures the data objects and their stated relationships (subject to the associated arity constraints), while it cannot capture more general integrity constraints to be enforced at the source or in the integrated domain. For example, requirements about the *acyclic* nature or the tree-hierarchical structure of a given data can only be captured using additional integrity constraints, IC, associated with the data. This treatment is analogous to the *data* vs. *integrity constraints* differentiation common in database management systems, and motivated in a similar fashion for preserving query processing efficiency [?].

[2]Note that in the rest of the paper, we simply omit the relationship names whenever they are not relevant to the discussion.



(a) (b) (c)

Figure 5: Example mappings of two IS-A hierarchies (a) node-to-node, (b) node-to-tree, and (c) either-or mapping

3. ALIGNMENT AND CONFLICT EXAMPLES

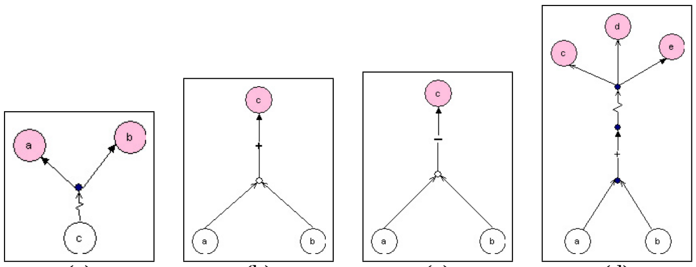
In this section, we highlight the need for extending the basic graph-based data representation presented above to allow for conflicts when working with imperfectly aligned data.

In the literature, there is a multitude of data and/or schema matching algorithms. In this paper, we refrain ourselves from assuming any particular matching or alignment strategy. However, for the sake of example, we consider a mapping scenario, where two concept hierarchies (consisting of concepts and IS-A relationships, with `N-1` arities, among them) are being integrated.

EXAMPLE 3.1 (NODE TO NODE MAPPING). *Let us assume that two nodes $n_{1,i}$ and $n_{2,j}$ from different IS-A hierarchies are identified as representing the same concept. Naturally, once these two hierarchies are integrated, these two nodes must be represented as a single node, n'. In other words, n' needs to preserve all the relationships that $n_{1,i}$ and $n_{2,j}$ have in their respective hierarchies. For example, all the children of these two nodes need to become the children of the combined node. However, preserving the original information after integration (while maintaining the appropriate arity constraints) is not always as easy. To see this, consider Figure 5(a), where a and b are two nodes that are mapped to each other. In this example, (since more than one immediate ancestor is not allowed in the integrated IS-A hierarchy) unless c and d are also identified as representing the same concept during mapping, the integrated hierarchy will contain an inconsistency. In particular, in a consistent world, only one of the alternative nodes, c or d, can be an immediate ancestor (parent) of the combined node.*

EXAMPLE 3.2 (MAPPING OF GROUPS OF NODES). *Figure 5(b) provides an example where a node in one IS-A hierarchy is mapped to an entire subtree in the second IS-A hierarchy. Consequently, the first node, a, and the root of the subtree, e, create a combined node in the integrated graph. Furthermore, after the integration, the child of a in the first IS-A hierarchy must become a child of one of the four nodes in the subtree corresponding to a in the second IS-A hierarchy. Thus, in resulting graph , $h \rightsquigarrow b \rightsquigarrow a \rightsquigarrow c$, $g \rightsquigarrow a \rightsquigarrow d$, are independently acceptable paths. On the other hand, these paths cannot be accepted together.*

EXAMPLE 3.3 (EITHER-OR MAPPINGS). *In many integration scenarios, the user may want to describe* either-or *type of mappings which (positively or negatively) relate the choices for different alternatives. Figure 5(c) provides an example of this type of*



(a) (b) (c) (d)

Figure 6: Example data graph segments with (a) choice, (b) positive coordination, (c) negative coordination, and(d) hybrid requirements

mapping: in this example, two nodes are mapped with the constraint that, the children of the two nodes are not compatible. Unlike the previous examples, this creates a graph where only one of the nodes can belong to a path.

The basic data graph construct described in the previous section does not have appropriate constructs for capturing alternative interpretations and coordination requirements that arise in such misalignment cases. In the next section, we describe how the need for making choices among alternative interpretations can be captured by suitably extending the constraint graph model.

4. DATA REPRESENTATION EXTENDED WITH ALTERNATIVES AND COORDINATION SPECIFICATIONS

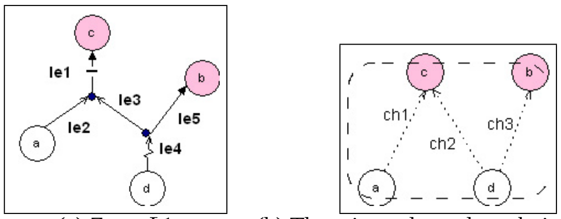
As illustrated by the alignment examples in the previous section, representing data with conflicts require constructs which *assert* the need for making choices among the various alternatives. Thus, in this section, we first extend the data graph with such constructs:

- Figure 6(a) presents a data graph with choice semantics. This graph contains a special edge leaving c, which can belong to only one path in the data; thus, in this example, either path $c \rightsquigarrow a$ or $c \rightsquigarrow b$ can be interpreted by the user to be true in the data, but not both.
- In a data graph with coordination, on the other hand, the alternatives associated with one or more edges can be further coordinated. In other words, coordination statements *assert* the need for making the same (or different) choices on involved edges. Figure 6(b) presents a positive coordination (where, if $a \rightsquigarrow c$ is interpreted to hold, then $b \rightsquigarrow c$ must also hold), while
- Figure 6(c) presents a negative coordination requirement (where, $a \rightsquigarrow c$ and $b \rightsquigarrow c$ can not simultaneously hold).
- Of course, the various choice and coordination constraints can be combined to obtain more complex scenarios. Figure 6(d) provides an example with hybrid choice and coordination requirements. This graph asserts that, in the given data, a and d have the same successor, and the shared successor is one of the c, d, and e data nodes.

In the above graphs, the various edges collectively enforce a set of mutually-dependent relationship constraints. We build the extended data graph on such blocks (or *zones*) of inter-dependent relationship constraints.

4.1 Zones of Relationship Contraints in a Data Graph

We refer to the building blocks of data graphs with choice and coordination requirements as *zones*. Intuitively each *zone* describes a set of inter-dependent choices in the data.



(a) Zone $L1$ (b) Three inter-dependent choices of $L1$

Figure 7: Zone example: (a) zone L_1 and (b) the three inter-dependent choices of $L1$

DEFINITION 4.1 (ZONE). *A zone (denoting a set of locally inter-dependent constraints) is a directed acyclic graph $L(Src, Snk, LV, LE)$, where*

- *the sources (Src) and sinks (Snk) are all data nodes (objects, entities),*
- *none of the (internal) vertices (LV) is a data node,*
- *LE are directed edges which connect sources, sinks, and internal nodes to each other. There are four types of edges in LE:*
  - exclusive edges (marked with \),
  - positive coordination edges (marked with +),
  - negative coordination edges (marked with -), *and*
  - regular edges (unmarked).
- *for any given pair of nodes, $v_i, v_j \in Src \cup Snk \cup LV$, there exists an undirected path ($v_i \rightsquigarrow_{undir} v_j$) in L that does not pass through any sources or sinks (i.e., data nodes).*

Figure 7(a) depicts an example zone, L_1. In this example, *source nodes* (lightly shaded) and *sinks* (darkly shaded) are connected through various choice and coordination edges. More specifically, $L_1(Src_1, Snk_1, LV_1, LE_1)$ is such that

- $Src_1 = \{a, d\}$,
- $Snk_1 = \{b, c\}$, and
- $LE_1 = \{le_1, le_2, le_3, le_4, le_5\}$

This zone effectively describes a number of *choices* that the data allows: the paths, $a \rightsquigarrow c$ and $d \rightsquigarrow c$, cannot be in the same data due to the negative coordination edge, le_1, while $d \rightsquigarrow c$ and $d \rightsquigarrow b$ are incompatible due to edge, le_4.

DEFINITION 4.2 (CHOICES OF A ZONE). *Given a zone, $L(Src, Snk, LV, LE)$, with k sources and l sinks, each path, $ch = i \rightsquigarrow j$, from the i^{th} source to j^{th} sink is said to be an available* choice *for L.*

In the above example, $ch_1 = a \rightsquigarrow c$, $ch_2 = d \rightsquigarrow c$, and $ch_3 = d \rightsquigarrow b$ are three inter-dependent choices (Figure 7(b)).

4.2 Zone-Graphs

Since we aim to use zones as the building blocks of the data with conflicts, we consider data graphs that can be partitioned into *zones*. Such graphs are referred to as zone-graphs:

DEFINITION 4.3 (ZONE-GRAPH). *A zone-graph, $G(V, E)$, consists of a set, $\mathcal{L}$, of zones, where*

- *$V = \bigcup_{L_i \in \mathcal{L}} (Src_i \cup Snk_i \cup LV_i)$, and*
- *$E = \bigcup_{L_i \in \mathcal{L}} LE_i$.*

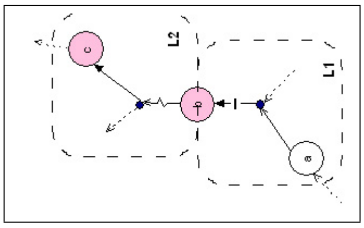


Figure 8: The data path from node a to node c is passing through two zones; i.e., it can be split into two zone-segments. We denote this data path as $a \rightsquigarrow b \rightsquigarrow c$

Different zones are allowed to share (and connect through) source and sink data nodes; i.e., $\forall L_i, L_j \in \mathcal{L}$, $Src_i \cap Src_j \supseteq \emptyset$, $Src_i \cap Snk_j \supseteq \emptyset$, $Snk_i \cap Src_j \supseteq \emptyset$, and $Snk_i \cap Snk_j \supseteq \emptyset$. On the other hand, the internal, non-data vertices of the zones or their edges can not be shared; i.e., $\forall L_i, L_j \in \mathcal{L}$, $LV_i \cap LV_j = \emptyset$ and $LE_i \cap LE_j = \emptyset$.

Each zone, $L_i \in \mathcal{L}$, of a zone graph has an associated relationship label, $rel(L_i)$, such as IS-A, WORKS-AT, or LIVES-AT.

Intuitively, each zone describes the alternative choices and coordination requirements for a mutually-related set of relationship edges (with the same label). The various zones of the graph are separated from each other by their shared data nodes. Conversely, we can also state that the individual zones of a zone-graph are connected to each other through their shared data nodes.

THEOREM 4.1. *Given data-graph $G(V, E)$, its zones can be computed and enumerated efficiently, in $O(E)$ time.*

PROOF. Due to the undirected connectivity requirement in Definition 4.1, the process of identifying zones can be done in $O(E)$ time, using a connected-components type of an algorithm and treating data nodes as *boundaries* of zones. $\square$

Note that in a basic data graph without conflicts (e.g., Figure 4), each edge between two data nodes is a zone with a single source, a single destination, and a single regular edge.

4.3 Data Paths on a Zone-Graph

Data paths on a zone-graph are defined similarly to the data paths on a basic data graph (i.e., Definition 2.2). In this more general case, on the other hand, a data path can pass through one or more zones. Thus, we can segment a given data path, dp, into a sequence of zone-segments. Intuitively, each zone-segment corresponds to a possible relationship between two data nodes (subject to the constraints of the corresponding zone).

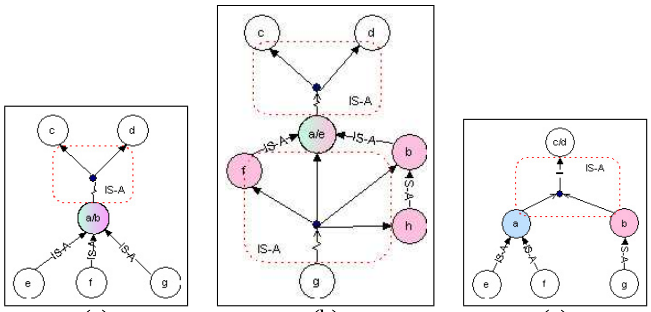
DEFINITION 4.4 (ZONE-SEGMENTS OF A DATA PATH). *A data path, $dp = \langle e_1, e_2, \ldots, e_{length(dp)} \rangle$, can be segmented into a sequence of zone-segments*

$$dp = \langle ls_1, ls_2, \ldots, ls_l \rangle,$$

where each $ls_i = source(ls_i) \rightsquigarrow dest(ls_i)$ is a data path from a source to a sink within the corresponding zone.

Each zone-segment on a data path corresponds to a *choice* made within the corresponding zone; therefore, in the rest of the paper we will use the terms *zone-segment* and *choice* interchangeably.

EXAMPLE 4.1. *Figure 8 depicts a data path, $a \rightsquigarrow b \rightsquigarrow c$, from data node a to data node c through data node b. In this example, this data path passes through two zones (L_1 and L_2) and, hence, it consists of two zone-segments (or choices).*



(a) (b) (c)

Figure 9: Zone-graphs obtained through the mapping examples in Section 3, Figure 5

4.4 Zonal-Graph Examples

In this subsection, we reconsider to the mapping examples (Examples 3.1 through 3.3) in Section 3 to show how zonal representation can capture conflicts that arise during data mapping process.

EXAMPLE 4.2 (NODE TO NODE MAPPING). *Example 3.1 (Figure 5(a)), illustrated a case where only one of the data nodes, c or d, can be a valid immediate ancestor (parent) of a combined node, due to the* `N-1` *arity constraint of IS-A hierarchies. This situation can be captured using the proposed zone-graph as shown in Figure 9(a): children of a and b can use the combined node as their parents and the combined node can have either c or d as its immediate ancestor, but not both. On the resulting zone-graph, some sets of paths, such as $\{e \rightsquigarrow b \rightsquigarrow d, f \rightsquigarrow b \rightsquigarrow d, g \rightsquigarrow a \rightsquigarrow d, e \rightsquigarrow a \rightsquigarrow d\}$ are consistent, while others, such as $\{e \rightsquigarrow b \rightsquigarrow c, g \rightsquigarrow b \rightsquigarrow d\}$ or $\{e \rightsquigarrow b \rightsquigarrow c, g \rightsquigarrow a \rightsquigarrow d\}$, are inconsistent.*

EXAMPLE 4.3 (MAPPING OF GROUPS OF NODES). *Example 3.2 (Figure 5(b)) provided a case where, after the integration, a number of paths (including $h \rightsquigarrow b \rightsquigarrow a \rightsquigarrow c$, $g \rightsquigarrow a \rightsquigarrow d$) are independently valid, but mutually incompatible paths. Figure 9(b) illustrates how these requirement are captured in a zone-graph using exclusive edges.*

EXAMPLE 4.4 (EITHER-OR MAPPINGS). *Example 3.3 (Figure 5(c)) provided an example of an integration scenario, where the user provided explicit* either-or *mappings which (positively or negatively) relate the choices for different alternatives. Figure 9(c) shows how this captured using zones and coordination edges. In this example, while both $e \rightsquigarrow a \rightsquigarrow c$ and $g \rightsquigarrow b \rightsquigarrow d$ are independently valid paths, they cannot be valid together due to coordination requirements.*

Finally, let us reconsider our motivating example of a scientist working under alternative hypothesis (described by two different taxonomies in Figure 1) in the Introduction of this paper. In these two coding systems, although some of the concepts have the same name, they may not be the same in terms of the corresponding semantics.

EXAMPLE 4.5 (TAXONOMY INTEGRATION). *Let us reconsider the two taxonomies in Figure 1 and let us assume a mapping scenario where the term 'Large Mammal' in the coding system 2 is different from the 'Large Mammal' in the coding system 1 and actually corresponds to both 'Medium Mammal' and 'Large Mammal':*

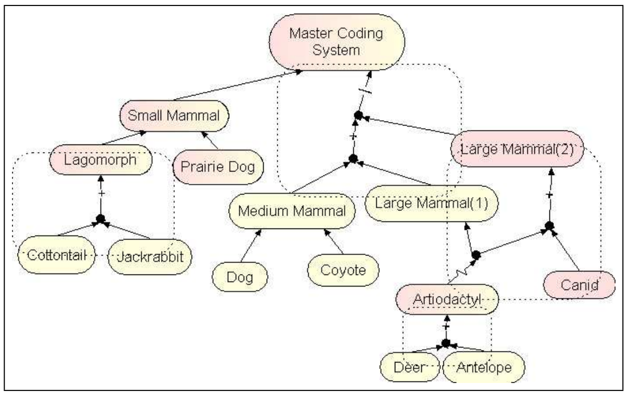


Figure 10: A zone graph integrating the two IS-A hiearchies in Figure 1

- *The concepts 'Cottontail' and 'Jackrabbit' co-exist in the coding system 1, but not in the coding system 2. This means that their acceptance in a given interpretation should be coordinated: they both should be accepted in the merged ontology (just as they do in the coding system 1) or rejected (just as they do in the coding system 2). We use a positive coordination edge (+) to describe this coordination constraint.*
- *The concept 'Artiodactyl' might belong to the large 'Large Mammal' class as defined in the coding system 1 or as defined in the coding system 2. We use an exclusive edge to describe the underlying choice.*
- *'Large Mammal' in the coding system 2 and 'Medium Mammal' in the coding system 1 are not compatible, since they correspond to different criteria. That means that either of them can be used in the consistent interpretation of the merged ontology, but not both. We use a negative coordination edge (-) to describe this requirement.*

Figure 10 presents a data graph describing the combined coding systems.

4.5 Zonal-Graphs and Integrity Constraints

As described in Section 2.2, zonal-graphs cannot capture all relevant constraints describing the data and their alignments. For example, multi-zonal statements, such as "*there exists no path with a cycle*" or "*there exists no path from a to b*", require constraints beyond what can be expressed within the data graph and used for generating candidate paths in the previous steps. Such constraints, which require path- or multi-path level evidence to verify or reject, are kept outside of the scope of the zone-graph specifically to ensure the efficacy of the agreement computation and agreement-based ranking processes. They are instead treated as *integrity constraints* at a post-processing step, along with the user's feedback. We will revisit the *integrity constraints* and their use in candidate result pruning and relevance feedback in Section 7.

5. OFF-LINE ANALYSIS OF ZONAL AGREEMENTS OF DATA SOURCES

As discussed in Section 1.2, in the presence of conflicts in the data, the proposed approach uses a feedback-driven mechanism to deal with alternative interpretations. In particular, the system relies on the *objective-to-subjective* implication ($sub \leftarrow obj$) of the Desideratum 1 to inform the user about the more likely (i.e., highest source agreement) interpretations for the given data. Intuitively,

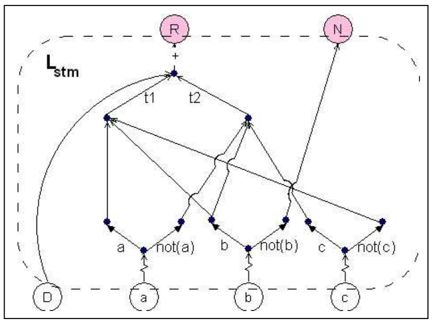


Figure 11: 3-SAT to zone reduction for the statement $stm = (a \vee b \vee \neg c) \wedge (\neg a \vee b \vee c)$

the *agreement* values measure how much different models of the data agree on stated relationships, with the default assumption that sources agree unless they explicitly conflict on data or integrity constraints. In this section, we focus on the off-line analysis of the zone graph for the computation of such *agreement* values on the choices corresponding to the individual zones.

5.1 Models of a Zone and Agreement Values of the Corresponding Choices

We define the models of an individual zone in a way parallel to the definition of models of the data with conflicts (Definition 1.1).

DEFINITION 5.1 (ALTERNATIVE MODELS OF A ZONE). *Given a zone, $L(Src, Snk, LV, LE)$, with k sources and l sinks, a set, $\mathcal{C}$, of choices of L is said to be a model if all the choices in $\mathcal{C}$ are pairwise consistent with the constraints associated with the edges of L and there is no other consistent set $\mathcal{C}' \supset \mathcal{C}$ of choices of L (i.e., $\mathcal{C}$ is set-maximal in L).*

Given these models, the zonal agreement of a choice *objectively* measures the degree of agreement among alternative models of the zone on this given choice.

DEFINITION 5.2 (ZONAL AGREEMENT ON A CHOICE). *Given a zone, $L(Src, Snk, LV, LE)$, with k sources and l sinks, the zonal alignment agreement value associated with a choice $ch = i \rightsquigarrow j$ is defined in terms of the possible alternative* models *in which the choice path, ch, is valid versus the total number* models *of the zone L:*

$$agr(ch) = \frac{\#\ models\ of\ L\ in\ which\ ch\ is\ a\ valid\ path}{\#models\ of\ L} = \frac{nm(ch, L)}{nm(L)}.$$

EXAMPLE 5.1. *Consider again the zone-graph in Figure 7. In this graph, there are three possible source/sink pairs, i.e., choices $ch_1 = a \rightsquigarrow c$, $ch_2 = d \rightsquigarrow c$, and $ch_3 = d \rightsquigarrow b$. Aong these, $a \rightsquigarrow c$ and $d \rightsquigarrow c$ are incompatible, $d \rightsquigarrow c$ and $d \rightsquigarrow b$ are incompatible, while $a \rightsquigarrow c$ and $d \rightsquigarrow b$ are compatible with each other. Thus, the two alternative models of this zone are $\{a \rightsquigarrow c, d \rightsquigarrow b\}$ and $\{d \rightsquigarrow c\}$ Another way to look at this is as follows: the choice $d \rightsquigarrow c$ is valid in only half of all the possible alternative models. Similarly, the choices $a \rightsquigarrow c$ and $d \rightsquigarrow b$ are both valid in only half of all the alternative models. Thus, in this example, we can conclude that $agr(a \rightsquigarrow c) = agr(d \rightsquigarrow c) = agr(d \rightsquigarrow b) = 1/2$.*

5.2 Off-line Agreement Analysis of a Zone

Figure 12 presents the outline of an algorithm which computes the zonal agreement value of a given choice in L. **[**OMITTED

```
Algorithm $agr(L, c)$
Given a zone, $L(Src, Snk, LV, LE)$ and a choice $ch$ of $L$ do
  1. Let $C$ be a constraint which enforces that $ch$ is in the model
  2. $nm_1 = countModels(L, C)$          /* $ch$ is in the model*/
  3. $nm_2 = countModels(L, \emptyset)$          /* all models*/
  4. return $(\frac{nm_1}{nm_2})$          /* return the $agr$ value of the choice $c$*/
```

Figure 12: Algorithm for computing the intra-zonal alignment agreement value associated with a given choice

EXAMPLE **]** Since the definition of the zonal agreement relies on a model-based interpretation of the zone itself, the computation complexity of this task reflects the cost of computing models of data with conflicts.

THEOREM 5.1 (PER-CHOICE COMPLEXITY). *Given a zone graph, L, and a choice, ch, the problem of counting the number of models in which ch occurs (i.e., computing $nm(ch, L)$) is NP-Complete.*

PROOF SKETCH 5.1. *The proof of NP-completeness of the problem of counting the number of models in which ch occurs is through a reduction from the well-known NP-complete 3-SAT problem, which asks the satisfiability of a boolean expression written in conjunctive normal form with 3 variables per clause. While, the complete reductive proof is outside of the scope of this paper, we provide a sketch the idea through an example. Consider the boolean expression*

$$stm = (a \vee b \vee \neg c) \wedge (\neg a \vee b \vee c).$$

The reduction step converts this statement (in polynomial time) into the zone, L_{stm} shown in Figure 11. Given this graph, we repose the satisfiability problem as follows:

$$satisfiable(stm) \longleftrightarrow (nm(D \rightsquigarrow R, L_{stm}) \geq 1)\,.$$

Due to the set-maximality requirement, the models which contain the choice path $D \rightsquigarrow R$ will be exactly those models which also use edges t_1 and t_2; i.e., satisfying both terms of the conjunctive statement, stm. Thus stm is satisfied iff there is at least one model of L_{stm}, where the path $D \rightsquigarrow R$ is used. □

COROLLARY 5.1 (COMPLEXITY OF ALGORITHM `agr()`). *Given a zone with μ edges, the complexity of the zonal agreement evaluation Algorithm in Figure 12 is $O(2^{\mu})$.*

PROOF SKETCH 5.2. *The proof of this corollary is based on the fact that the depth of the recursion is at most the number of edges in the given zone. Since, the zonal value agreement computation is NP-complete per Theorem 5.1, we do not expect to find a polynomial solution for this task.* □

Nevertheless it is important to note that, although by Corollary 5.1 the cost of the agreement analysis is exponential in the size of the zones, the initial zone-partitioning of the data graph to obtain small zones[3] and modular (per-zone) nature of the agreement-analysis prevents this step from becoming costly.

COROLLARY 5.2 (COMPLEXITY OF ZONE-GRAPH ANALYSIS). *The complexity of the zonal agreement evaluation for a complete data graph G, with ζ zones, where the largest zone has μ edges is $O(\zeta \times 2^{\mu})$.*

[3]Normally, each zone represent a single or closely related few relations, while the data graph can be composed of many relationships.

Thus, zone partitioning significantly reduces the cost of this off-line process: $O(\zeta \times 2^{\mu}) \ll O(2^{\zeta \times \mu})$, which would have been the cost of agreement computation without zone partitioning. **[**FROM SELCUK: **incremental updates??]** Note also that the zone-graph analysis is an off-line pre-processing process which is performed on the integrated data graph once, at the bootstrap phase. This off-line nature of this process ensures that the interactive relevance feedback process is not affected from the cost of the static agreement analysis.

6. QUERY PROCESSING AND RANKED CANDIDATE ENUMERATION

Since the goal is to help the user identify the best matches to her query based on the available data, at the first stage of the query processing and conflict resolution, the user is provided with a set of high-agreement candidate matches. These candidates help the user not only in observing the best matches to her query based on the current state of integration and conflict resolution, but also in seeing the critical conflicts that affect these most-agreed upon (i.e., objectively most likely) results to the query. This, we refer to as the *objective-to-subjective* ($sub \leftarrow obj$) flow of feedback which informs the user about the more likely (i.e., highest source agreement) interpretations for the given data (Section **??**).

Definition **??** of ranked interpretation calls for a set, $\mathcal{S}$, of statements to be ranked. Although this set of statement can include any general statement about the data, since they describe the relationships between entities, in this paper, we take *data paths* as the basic *statements of interest.* Thus, given a source, n_{src}, and destination, n_{dst}, nodes, the data paths from n_{src} to n_{dst}, are ranked and a small ($\leq k$) subset of candidate data paths from n_{src} to n_{dst} are chosen to be presented to the user based on the agreement values (Figure 3 in the Introduction section).

6.1 Computation of the Agreements on Data Paths

Since, per Definition 4.4, each zone-segment of a path corresponds to a possible interpretation of an alignment constraint, given a data path, $dp = \langle ls_1, ls_2, \ldots, ls_l \rangle$, its overall *(multi-zonal) agreement value* can be defined in terms of the agreements of the alignment choices involved in it. In particular, since

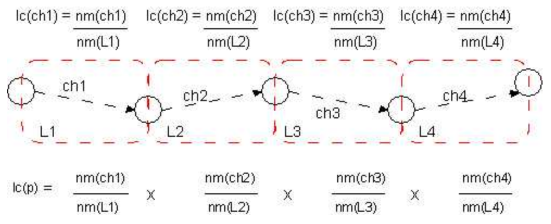
- the agreement value of a *choice* in a given zone is the ratio of the number of models of the zone in which the choice is valid to the number of all possible models of the zone, and since
- each zonal alignment choice is independent from the choices of the other zones (modulo the integrity constraints that will be enforced at a later stage, in Section 7)

we can treat the agreement ratios as independent selection probabilities and, thus, define the agreement value of a data path as the multiplication of the agreements of all the involved zonal choices (Figure 13).

DEFINITION 6.1 (AGREEMENT OF A DATA PATH). *Given a data path, $p = \langle e_1, e_2, \ldots, e_{length(p)} \rangle$, and its zone-segment representation, $p = \langle ls_1, ls_2, \ldots, ls_l \rangle$, we define the* agreement value *of p as*

$$agr(p) = \prod_{0 \leq i \leq l-1} agr(ls_i).$$

In Section 7, we will relax the independence assumption and consider the affects of multi-zonal integrity constraints that tie choices in a given zone to the choices in the other zones.



lc(ch1) = nm(ch1) / nm(L1)
lc(ch2) = nm(ch2) / nm(L2)
lc(ch3) = nm(ch3) / nm(L3)
lc(ch4) = nm(ch4) / nm(L4)
ch1
ch2
ch3
ch4
L1
L2
L3
L4
lc(p) = nm(ch1)/nm(L1) X nm(ch2)/nm(L2) X nm(ch3)/nm(L3) X nm(ch4)/nm(L4)

Figure 13: A path which passes through four zones involves four independently made alignment choices along the way. Thus, the ratio of the models of the data where this path is valid can be computed by multiplying the intra-zonal agreements of the individual choices

```
Algorithm $K-HighAgreementPaths(G, n_{src}, n_{dst}, k)$
Given a zone-graph $G(V,E)$, a source node, $n_{src}$, and a destination node, $n_{dst}$, and a
positive integer $k$, do
  1. Let $V' = \emptyset$ and $E' = \emptyset$                    /* Construct a dual graph*/
  2. Let $\mathcal{L}$ be the set of all zones of $G$,
  3. for all $L_i \in \mathcal{L}$ do
     (a) $V' = V' \cup Src_i \cup Snk_i$;
     (b) for all $ch = src(ch) \rightsquigarrow dest(ch)$, where $ch$ is a choice path in $L_i$;
         i. Let $e_{ch}$ be a new edge with length $length(e_{ch}) = -log(agr(ch))$;
         ii. $E' = E' \cup \{e_{ch}\}$
  4. $resultPaths = YenKShortestPath(V', E', length, n_{src}, n_{dst}, k)$;
  5. return $resultPaths$;          /* Return $k$-shortest paths of the dual*/
```

Figure 14: Algorithm for ****

6.2 Agreement-Ranked Enumeration of Candidate Paths

To provide the user with the most likely paths based on the available zonal conflict/agreement analysis, the system needs to identify the highest-agreement data paths. We formally pose this task in the form of a k *highest-agreement data paths problem*.

DEFINITION 6.2 (k HIGHEST-AGREEMENT DATA PATHS). *Given a zone-graph $G(V,E)$, a source node, n_{src}, and a destination node, n_{dst}, identify the k* highest-agreement *data paths from the source node, n_{src}, to the destination node, n_{dst}.*

Figure 14 presents an algorithm to solve the k high-agreement data paths problem by translating it into the *k-shortest simple paths*, a classical problem in graph theory. Due to its application in various application domains, such as transportation or networking, this problem has been studied extensively and a number of algorithms have been developed [?]. Among these, Yen's algorithm [?] is well-known due to its general and optimal nature [?]. This algorithm uses a tree-based data structure to [FROM SELCUK: **complete**]. Various other algorithms [?] build on this approach, yet their improvements are limited to specific application domains[FROM SELCUK: **we need to say a little bit more about the newer algorithms and their complexities**]. In the algorithm presented in Figure 14, we use Yen's k-shortest simple paths algorithm for enumerating k *highest-agreement* data paths.

THEOREM 6.1 (CORRECTNESS). *The k-highest agreement algorithm presented in Figure 14 is correct.*

PROOF. Given a graph $G(V,E)$, the algorithm presented in Figure 14 constructs a dual graph, $G'(V',E')$, where the choices in each zone are replaced with explicit edges between the corresponding source/sink pairs. Thus, it is trivial to show that there is a one-to-one mapping between any data path on $G(V,E)$ and a path on $G'(V',E')$.

In Step 3(b)i of the algorithm, given a choice ch in G, the length of the corresponding edge in G' is set to

$$length(e_{ch}) = -log(agr(ch)),$$

which is always a positive number. Thus, the k-shortest path algorithm ran on G' (Step 4) will return k simple paths, such that the term

$$\sum_{ch \in path} -log(agr(ch)) = log\left(\prod_{ch \in path} \frac{1}{agr(ch)}\right),$$

is minimized. Since the log function is monotonic, this corresponds to the minimization of the term $\prod_{ch \in path} \frac{1}{agr(ch)}$ or, equivalently, the maximization of the term

$$\prod_{ch \in path} agr(ch).$$

By Definition 6.1, this term is equal to the agreement of the data path on the original zone graph G. Hence, these enumerated paths are also the k *highest-agreement* data paths in G. $\square$

THEOREM 6.2 (COMPLEXITY). *The worst case execution time of the k* highest-agreement *data paths algorithm in Figure 14 is $O(kn(m^2 + nlogn))$.*

PROOF. Given a dual graph, G', with n' nodes and m' edges, Yen's algorithm would identify the k-shortest simple paths in G' with the worst case time of $O(kn'(m' + n'logn'))$ [?].

Given a graph, G, with n nodes and m edges, the algorithm in Figure 14 creates a dual graph, G', with at most $n' = n$ nodes and $m' = m^2$ edges. The worst case occurs when a zone in G with u edges leads to $O(u^2)$ individual choices, each with a corresponding edge, in G'.

In the first phase of the algorithm (Steps 1 through 3), the dual graph creation costs $O(m^2)$ time. In the second phase (Step 4), given the dual graph, G', with $m' = O(m^2)$ and $n' = n$ nodes, Yen's algorithm costs $O(kn(m^2 + nlogn))$ time. Thus, the worst case execution time of the algorithm in Figure 14 is $O(kn(m^2 + nlogn))$. $\square$

The k highest-agreement paths that are generated at this step are the k best matches to the user's query based on the zone-graph. However, as described in Section 4.5, the integrity constraints which require path- or multi-path level evidence to verify or reject candidates are kept outside of the scope of the zone-graph and, thus, necessitate further post-processing.

7. CONFLICTS WITH MULTI-ZONAL INTEGRITY CONSTRAINTS AND RELEVANCE FEEDBACK

Given the k highest-agreement paths based on the zone-graph (which are created based on the assumption that the zonal choices are independent from each other), multi-zonal integrity constraints need to be considered to verify the *validity* of the ranking generated in the earlier stage.

EXAMPLE 7.1 (ACYCLICITY CONSTRAINT). *Consider an integrity constraint which asserts that the data is acyclic. Such a constraint can be commonly found in concept (IS-A) hierarchies.*

Note that, although they can be independently simple *(i.e. acyclic), two paths in a given candidate set may imply a cycle in* *the data when considered together. This means that (a) either the zonal-constraints that were used in the production of these paths or (b) the integrity constraint asserting the acyclicity of the data were overly trusted. Thus, the knowledge of this conflict needs to be reflected back to the* validity *of the involved constraints and if necessary on the involved candidate paths.*

When it is found that highest-agreement paths are conflicting with the integrity constraints, this may result in (a) updates on the agreement values computed in the earlier processing stages or (b) adjustment of the validity of the integrity constraints themselves. The user feedback, i.e., *subjective-to-objective information flow ($sub \rightarrow obj$)*, is used to decide the appropriate adjustment.

7.1 Degree of Conflict with Integrity Constraints

Since data paths are weighted with *agreement* values, quantifying their likelihood based on the zone-graph, the conflicts between available candidate paths and the integrity constraints must also reflect the *likelihood* of their inconsistencies. This leads to the concept of *degree of conflict* between a set of paths and constraints.

DEFINITION 7.1 (DEGREE OF CONFLICT). *Given a set of paths P (on a data-graph G) and constraints C, the degree of conflict between P and C (denoted as $conf_G(P,C)$) is defined as*

$$conflict_G(P,C) = 1 - \frac{nm(P\ collectively\ satisfy\ C\ in\ G)}{nm(G)}$$

As an example, let us consider the acyclicity constraint, "*there exists no path with a cycle*," and computation of the degrees of acyclicity conflicts.

EXAMPLE 7.2 (DEGREE OF ACYCLICITY CONFLICTS). *Given two paths, $P_1 = n_{1,1} \rightsquigarrow \ldots \rightsquigarrow n_{1,k}$ and $P_2 = n_{2,1} \rightsquigarrow \ldots \rightsquigarrow n_{2,l}$, we can define the acyclicity, $noCyc(P_1,P_2)$, as follows:*

$$\begin{aligned} noCyc(P_1,P_2) = \forall_{i,j}\ &(\exists path(n_{1,i},n_{2,j}) \rightarrow \nexists path(n_{2,j},n_{1,i})) \wedge \\ &(\exists path(n_{2,j},n_{1,i}) \rightarrow \nexists path(n_{1,i},n_{2,j})) \end{aligned}$$

Then, we can define an acyclicity violation as

$$\{P_1,P_2\} \rightarrow \neg\ noCyc(P_1,P_2).$$

Thus, given the agreement *values (i.e., likelihood of existence) for all relevant paths among the nodes of P_1 and P_2, the degree of conflict (Definition7.1) can be computed probabilistically. Note that such agreement values can indeed be computed in advance (off-line) using an all-pairs shortest path algorithm (such as Floyd-Warshall's [?]) in $O(V^3)$. Given these off-line computed values, the computation of the degree conflict between two paths P_1 and P_2 can be done in $O(length(P_1) \times length(P_2))$.*

The concepts of *integrity constraints* and *degrees of conflicts* introduced above together with the concepts of *zones, choices,* and *zonal agreements* introduced in Section **??** enable us to articulate the basic axioms that these need to obey to be accepted *valid*.

7.2 Axioms of Validity

The following axioms characterize the dependencies between zonal-choices of the candidate paths and the conflicts implied by the set of integrity constraints:

AXIOM 1 (ZONAL CONFLICTS AND VALIDITY). *If two valid paths are conflicting within a given zone, then both corresponding zonal choices cannot be simultaneously* valid*: i.e., it holds that $\forall_{p_1,p_2 \in P}$*

$$valid(p_1) \wedge valid(p_2) \wedge conflict_G(\{p_1,p_2\},z) \rightarrow \neg valid(zc_1) \vee \neg valid(zc_2)$$

AXIOM 2 (INTEGRITY CONSTRAINTS AND VALIDITY). *If a given set of paths and constraints are in conflict, then it can not be true that all paths and all the constraints are valid: i.e., it holds that $\forall_{P' \subseteq P, C' \subseteq C}$*

$$conflict_G(P',C') \rightarrow \left(\bigvee_{c \in C'} \neg valid(c)\right) \vee \left(\bigvee_{p\ in P'} \neg valid(p)\right)$$

In addition, the validity predicate satisfies $valid(\neg u) = \neg valid(u)$, $valid(u \vee v) = valid(u) \vee valid(v)$, and $valid(u \wedge v) = valid(u) \wedge valid(v)$.

7.3 User Feedback and Subjective Ranked Interpretation

Although the axioms above provide a framework in which the validity of the candidate paths as well as zonal and integrity constraints can be assessed, they do not do so unambiguously; in other words, these axioms can be satisfied in multiple ways. Thus, the user feedback is necessary to inform the system to decide on the appropriate adjustment (i.e., *subjective-to-objective information flow, $sub \rightarrow obj$*).

Given the initial set of candidates, their agreement values, and their degrees of conflicts with the integrity constraints, the user can provide validity feedback in the form of preferred validity rankings. Examples include, $valid(p_i) > valid(p_j)$, $valid(c_i) > valid(c_j)$, and $valid(p_i) > valid(c_j)$. These describe users' ranked interpretation of the statements regarding the paths and the constraints.

7.4 Relationship between Validity and Agreement

Both agreement and validity values assess the rank the likelihood of a given statement. *Agreement* describes the objective ranked interpretation, based on the zonal graph, while the *validity* includes integrity constraints as well as user feedback to reflect the subjective ranking. Thus, the subjective to objective correspondence (Desideratum *) implies that validity values should be related to the objective agreement values computed in the previous steps, unless there are multi-zonal conflicts or user intervention.

DESIDERATUM 3. *$\forall p \in P$, it should be that*

$$valid(p) \sim 1 - \frac{log(agr(p))}{log(agr_{min})},$$

and

$$\forall_{zc \in zonalchoice(p)}\ valid(zc) \sim 1 - \frac{log(agr(zc))}{log(agr_{min})}.$$

Here, $agr_{min} > 0$ is the smallest agreement value computed for the paths and the zonal choices input to the integrity constraint analysis and user feedback phase.

For the smallest agreement value in the input, $1 - \frac{log(agr_{min})}{log(agr_{min})}$ is equal to 0; for any choice or path with agreement value 1, $1 - \frac{log(1)}{log(agr_{min})}$ is also equal to 1. Thus, intuitively, this desideratum implies that validity should always be between 0 and 1 and should increase monotonically with the agreement value, agr. The desideratum uses $\sim$ as opposed to $=$, because agreement values are computed without considering multi-zonal constraints and conflicts with such complex constraints and user feedback may affect the validity. Baring such conflicts and user intervention, validity should mirror agreement values

Furthermore, the desired relationship between the agreement and validity values necessitates a correspondence between the validity of a path and the validity of the choices involved in it.

PROPOSITION 7.1 (ZONAL CHOICES AND PATH VALIDITIES). *A path's* validity *is related to the validity of its zonal choices; in particular, it holds that* $\forall_{p \in P}$

$$valid(p) = 1 + \sum_{zc \in zonechoice(p)} (valid(zc) - 1).$$

PROOF SKETCH 7.1. *The use of* $\sum$ *to relate validity of a path to the validity of the corresponding zonal choices is similarly motivated to the use of* $\sum$ *in the computation of* k *highest agreement paths in Section* **??**. *In particular, Desideratum 3 and the fact that* $agr(p) = \prod_{zc \in zonalchoices(p)} agr(zc)$ *together imply that*

$$1 - \frac{log(agr(p))}{log(agr_{min})} = 1 - \left(\sum_{zc \in zonalchoices(p)} \frac{log(agr(zc))}{log(agr_{min})} \right)$$

Thus, $valid(p)$, *i.e., the right hand side of the equation, should be equal to*

$$1 + \sum_{zc \in zonalchoices(p)} \left(\left(1 - \frac{log(agr(zc))}{log(agr_{min})} \right) - 1 \right),$$

i.e., the left hand side of the equation). This term can be rewritten as $1 + \sum_{zc \in zonechoice(p)} \ (valid(zc) - 1)$. □

7.5 Measuring Validity and Updating Zonal Agreements

Measuring validity based on the above axioms require a compatibility and conflict analysis scheme. Maximal clique-based model enumeration (used in most truth maintenance work [27] for compatibility and conflict analysis), could be extremely costly. even within the limited context of candidate results and the relevant integrity constraints. Furthermore, the conflict and agreement values associated with the paths and constraints are non-boolean. Thus, for measuring validity based on the above axioms, we can not rely on boolean or set-based schemes. Instead, we use the conflict and agreement values by translating the axioms and user-feedback into a fuzzy constraint program.

A fuzzy set, F, with domain D is defined [39] using a membership function, $F : D \rightarrow [0, 1]$. A fuzzy predicate then corresponds to a fuzzy set; instead of returning *true(1)* or *false(0)* values for propositional functions, fuzzy predicates return the corresponding membership values. The meaning of the constraint program depends on the fuzzy semantics chosen for the logical operators $\wedge$ and $\vee$. It is well established that the only fuzzy semantics which preserves logical equivalence of statements (involving conjunction and disjunction) and is also monotonic is the `min` semantics, where $a \wedge b = min(a, b)$, $a \vee b = max(a, b)$, and $\neg a = 1 - a$. Therefore, we use fuzzy `min` semantics to translate the validity axioms into a constraint program as shown in Figure 15.

Given this constraint program, we search for a solution with *maximal integrity constraint validity and better achievement of Desideratum 3*; i.e., the **maximization** function for solving the constraints in Figure 15 is

$$\sum_{ic \in IC} valid(ic) - \sum_{zc \in ZC} \left| valid(zc) - \left(1 - \frac{log(agr(zc))}{log(agr_{min})} \right) \right|$$

Once the validity values are computed, the final step in the relevance feedback process is the update of the zonal agreement values of the choices based on the validity assessments obtained in this step: i.e, using the computed validities and Desideratum 3, we can compute new agreemeent values, agr_{new}, as follows:

$$log(agr_{new}(zc)) = (1 - valid(zc)) \times (log(agr_{min})).$$

After this update, the system is ready re-compute the k highest agreement paths for the next cycle.

7.6 Time to Compute Trust Values

Table 1 presents the execution times for a constraint solver (LINGO on a 2GHz Pentium4 with 1GB main memory) under different scenarios and using different optimization strategies. In particular, the table lists results by LINGO's global solver (Glb.) as well as by its *multistart* (MS) solver, which selects a subset of candidate starting points for local optimizations.

The first thing to note is that the local optima found by the multistart solver are very close to the optimal score obtained by the global solver. In fact multistart with few initial starting points is in many cases as good as multistart ran with a larger number of initial starting points[4]. Thus, less than optimal, but fast, multistart solver can be used effectively when the optimal trust values are too expensive.

The second thing to notice is that, for all the experimented scenarios the execution time of the multistart solver is less than 20 seconds. In particular, the multistart solver with two initial points returns results in less than 3 seconds. The global solver (Glb.) on the other hand works very fast (under 4 seconds) for scenarios with small number of conflicts. But, as the number of conflicts in the assertions increases, the global solver becomes too slow for interactive use. Therefore, for such cases, `QUEST` relies on the multistart solver which approximates the objective value well. Note also that, interestingly, when the number of conflicts in the results are also large, the global solver works faster than the case with only assertion conflicts. This points to the fact that conflicts in results indeed inform the solver.

8. RELATED WORK

In this section, we review the related work on different aspects related to possibly conflicting data integration.

In the area of *collaborative data sharing*, the system *Orchestra* [41,42] focuses on managing disagreement (at both schema and instance levels) among different (relational) collaborating members, and addresses the problem of propagating updates possibly occurring at any information source. No global consistency requirement is imposed in the collaborative system. Acceptance of tuples from other sources is seen as an option, and no participant is required to update its data instance to agree with the other participants. Similarly, updates are propagated, through a series of *acceptance rules*, only to those participants who trust their sources and/or their values.

In *TRIO* [43], the existence of alternative database instances is captured through the notion of uncertainty associated to the available data. An uncertain database represents multiple possible instances, each representing a possible state for the database. *Probability* values are attached to the data stored in uncertain databases, and *lineage* captures data items' derivation (and it can be used for understanding and resolving uncertainty). Similarly to our approach, the probability associated to attribute values in uncertain databases can be interpreted as a way of capturing the likelihood for the data values, in the alternative (possibly mutually inconsistent) scenarios, and we can see *lineage* as correlating uncertainty in query results with the uncertainty in the input data, thus providing the "context of validity" for the derived data. As a major difference with our approach, in *TRIO* the probabilities associated to data items are taken as known at the storage time (derivation time

[4]Due to the randomness of the initial points, multistart with a larger number of starting points can sometimes return a lower objective value.

Given paths, P, zones, Z, zonal choices, ZC, and integrity constraints, IC,

$$\begin{array}{ll}
\forall c \in ZC \cup IC & 0 \le valid(c) \le 1; \\
\forall p \in P & 0 \le valid(p) \le 1; \\
\forall p \in P & valid(p) = 1 + \sum_{zc \in zonalchoices(p)} (valid(zc) - 1) \\
\forall p_1, p_2 \in P, z \in Z & \min\{valid(p_1), valid(p_2), conflict(\{p_1, p_2\}, z)\} \le 1 - min\{valid(zc_1), valid(zc_2)\} \\
\forall P' \subseteq P, C' \subseteq IC & conflict(P', C') \le max\{max\{1 - valid(c) | c \in C'\}, max\{1 - valid(p) | p \in P'\}\}
\end{array}$$

Figure 15: Fuzzy constraint program capturing the validity axioms

Scenario						**Time to Compute (seconds)**					**Percentage of Glb.'s Objective Value**			
$\|R\|$	$\|E\|$	$\|A^+\|$	$\|A^-\|$	Conflicts in pairs of Rs	Conflicts in pairs of As	**MS2**	**MS4**	**MS8**	**MS16**	**Glb.**	**MS2**	**MS4**	**MS8**	**MS16**
10	2	50	2	2%	2%	1.3	2.6	4.4	11.5	0.8	97.6%	99.3%	99.7%	100.0%
10	2	100	2	2%	2%	2.3	5.0	10.4	13.0	2.1	99.5%	99.5%	100.0%	100.0%
20	4	100	4	2%	2%	1.4	4.3	9.5	15.8	3.2	97.9%	98.9%	99.5%	99.2%
20	4	100	4	10%	2%	1.4	14.2	16.4	16.8	9.8*	95.4%	95.1%	96.0%	94.0%
20	4	100	4	2%	10%	1.9	17.7	17.7	17.6	$>$10mins	98.5%	98.1%	98.4%	98.6%
20	4	100	4	10%	10%	1.6	16.1	15.6	16.1	161.2**	99.1%	99.1%	98.8%	99.1%
50	5	250	5	2%	2%	2.3	15.8	15.2	15.4	$>$10mins	99.4%	99.4%	99.1%	99.4%
50	5	250	5	10%	10%	5.6	36.1	36.1	35.4	518.4*	99.9%	99.9%	99.9%	99.8%

***** For this configuration, LINGO global solver took more than 10minutes in 40% of the runs. The reported average execution time for Glb. are of the remaining runs.

****** For this configuration, LINGO global solver took more than 10minutes in 20% of the runs. The reported average execution time for Glb. are of the remaining runs.

Table 1: Trust value computation using LINGO optimization software. For each scenario, the table lists the objective scores for the local optima by LINGO's multistart solver (MS), relative to the optimum by its global solver (Glb.).

for derived items), and they do not reflect user's feedback. Moreover, in the derivation process, lineage guarantees that a coherent derivation flow is maintained, and there is no way for the user to specifically choose to trust and combine results that are not coherent with the original lineage derivation policy, i.e., the constraints dictated by the derivation strategy cannot be overruled by users' data interpretations.

From a more theoretical (i.e, system independent) perspective, [**?**, 44, 45] investigate on formal characterizations of the notion of consistent data in a possibly inconsistent database. They introduce a number of alternative repair semantics, where a *repair* of a (relational) database instance is another database instance over the same database which *maximally close* to the given one (different semantics correspond to different definition of the closeness metric), and they discuss on the complexity of consistent query answering in integrated databases. Similarly, [**?**] addresses modeling and query answering complexity issues in data integration systems, with a global architecture providing a reconciled, integrated, virtual view over the real data sources. Also in these cases, integrity constraints are taken as valid, and the focus is on the problem of returning answers to the queries while guaranteeing consistency with respect to the given constraints. In our case, we give the user the option of assessing her belief in the validity of the integration constraints, as well as on the integrated data sources.

In their work on nondeterministic choices in logic programming languages, Zaniolo [40] and his colleagues suggest that in logic database languages, one may often wish to express the fact that only one of several possible ways of satisfying an atom is nondeterministically selected. They then use this to define a choice semantics for logic programs with negation. In contrast, in our work we are not dealing with logic programming languages, and we use choice (*feedback*) mechanisms to assign *trust* values to underlying *data sources* and *constraints*.

In the Artificial Intelligence community, the problem of dealing with incomplete and/or inconsistent information led to the definition of alternative, multiple model semantics, and to the development of non monotonic reasoning systems. Multiple model semantics, like the *stable model semantics*, [15], associate multiple, equally likely, models to the given knowledge base, each one corresponding to a possible context, or a possible consistent scenario described by the knowledge base. A query (*goal*) succeeds if at least a context (*model*) is found in which the query is true. Problem solvers interact with *truth maintenance systems* (TMSs) [12], that record and maintain the justifications for the possible context (*belief sets*) under consideration. Dependency networks allow the detections of the possible reasons for conflicts, and in the presence of a conflict a knowledge base revision process starts, restoring consistency. Unlike the related work in this area, we choose to tolerate conflicting information in the knowledge base, and we propose efficient *ranking* algorithms to enable the user to explore and revise the knowledge within the context of a given query.

To enable this, we propose an assertion (constraint)-based model of knowledge, (

9. CONCLUSION

In this paper, we presented innovative techniques to deal with the computational complexity and the ill-defined nature of the conflict resolution problem. In particular, we presented a novel, feedback-driven approach to query processing in the presence of conflicts. The novel feedback process relies on a ranked interpretation of the data, as opposed to more traditional model-based interpretations. The objective-to-subjective correspondence of the ranked interpretations enables the user to explore the available data within the context of a query and be informed regarding data and relationship-constraints critical to a given query before providing feedback. In a similar fashion, the subjective-to-objective correspondence of the ranked interpretations inform the system regarding user's preferences and domain knowledge within the context of a query. We provided data structures and algorithms that enable an efficient and effective implementation of the proposed feedback cycle.

Document object model http://www.w3.org/TR/REC-DOM-Level-1/.

Xquery. http://www.w3.org/TR/xquery/.

R. Agrawal, R. J. Cochrane, and B. G. Lindsay. On maintaining priorities in a production rule system. VLDB 1991.

S. Al-Khalifa, *et al.* Structural joins: A primitive for efficient XML query pattern matching. *ICDE*, 2002.

M. Arenas and L. Libkin. XML data exchange: consistency and query answering. In *PODS*, pages 13–24, 2005.

A. Bonifati, E.Q. Chang, and L.V. Lakshmanan. Heptox: Marrying XML and heterogeneity in your p2p databases. In *VLDB*, 2005. Demo.

R. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. SWAT, 1990

P. Buneman, W. Fan, and S. Weinstein. Query optimization for semistructured data using path constraints in a deterministic data model. *DBPL*, 1999.

K.S. Candan, J. Grant, and V. Subrahmanian. A unified treatment of null values using constraints. *Information Systems Journal*, 98(1-4):99–156, May 1997.

K.S. Candan, *et al.* Discovering mappings in hierarchical data from multiple sources using the inherent structure. *J. of KAIS*, accepted 2005.

A. Doan, P. Domingos, and A. Y. Levy. Learning source description for data integration. In *WebDB*, pages 81–86, 2000.
J. Doyle. A truth maintenance system. *J.of Artificial Intel.*, 12: 231–272, 1979.
S. Flesca, *et al.* Repairs and consistent answers for xml data with functional dependencies. *Xsym* pages 238–253, 2003.
D. Florescu and D. Kossman. Storing and Querying XML Data using an RDBMS. *IEEE Data Eng. Bulletin*,22(3):27-34, 1999.
M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. *International Conference and Symposium on Logic Programming*. 1988.
L. Giordano, A. Martelli and M.L. Sapino". Extending negation as failure by abduction: a 3-valued stable model semantics. *J. of Logic Programming*, 1996.
A. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management. In *ICDE*, 2003.
T. Imielinski and W. Lipski. Incomplete information in relational databases. *JACM*, 31(4):761–791, 1984.
Y. E. Ioannidis and T. K. Sellis. Conflict resolution of rules assigning values to virtual attributes. SIGMOD, 1989.
D. S. Johnson and C. H. Papadimitriou. On generating all maximal independent sets. *Info. Proc. Letters*, 27, 1988.
A. Kementsietsidis, M. Arenas, and R. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. *SIGMOD* 2003.
K. W. Kintigh *et al.* Workshop on cybertools for archaeological data integration. http://cadi.asu.edu/, December 2004.
K.-C. Liu and R. Sunderraman. A generalized relational model for indefinite and maybe information. TKDE, 3(1), 1991.
M. Liu and T. W. Ling. A data model for semistructured data with partial and inconsistent information. *LNCS*, 1777, 2000.
J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, pages 49–58, 2001.
J. McHugh, *et al.* Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3):54–66, 1997.
R.Mercer and V.Risch. Properties of maximal cliques of a pair-wise compatibility graph for three nonmonotonic reasoning system. *Answer Set Prog.*,2003.
R. Miller, L. Haas, and M. Hernandez. Schema mapping as query discovery. *VLDB*, pages 77–88, 2000.
T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In *VLDB*, pages 122–133, 1998.
P.Mitra, G.Wiederhold, and M.Kersten. A graph oriented model for articulation of ontology interdependencies. *EDBT*'00.
J.W. Moon and L. Moser On cliques in graphs. Israel Journal of Mathematics, 3, 2328, 1965.
W. Ng. Repairing inconsistent merged xml data. *DEXA*, pages 244–255, 2003.
L. Palopoli, D. Sacca, and D. Ursino. An automatic technique for detecting type conflicts in database schemes. *CIKM*, 1998.
Y. Qi, K.S. Candan, M.L. Sapino, and K. Kintigh.Quest:QUery-driven Exploration of Semistructured Data with ConflicTs *submitted* to CleanDB 2006.
E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4), 2001.
K. Spielmann, J. Driver, D. Grayson, E. Reitz, S. Kanza, and C. Szuter. Faunal working group. http://cadi.asu.edu.
I. Tatarinov, *et al.* Storing and querying ordered XML using a relational database system. *SIGMOD*, pages 204–215, 2002.
Y. Wu, J. M. Patel, and H. V. Jagadish. Structural join order selection for XML query optimization. In *ICDE*, 2003.
L. Zadeh. Fuzzy Sets. Information and Control, pp. 338-353, 1965.
C. Zaniolo. *A United Semantics for Active and Deductive Databases*, chapter Rules in Database Systems. 1994.
Z. G. Ives, N. Khandelval, A. Kapur, and M.. Cakir ORCHESTRA: Rapid, Collaborative Sharing of Dynamic Data CIDR 2005.
N. E. Taylor and Z. G. Ives Reconciling while tolerating disagreement in collaborative data sharing SIGMOD 2006-11-02
O. Banjelloun, A. Das Sarma, A. Halevy, and J.. Widom ULDBs: Databases with Uncertainty and Lineage VLDB 2006.
A. Lopatenko and Leopoldo Bertossi. Complexity of Consistent Query Answering in Databases under Cardinality-Based and Incremental Repair Semantics. Technical Report, 2006
L. E. Bertossi, B. Loreto Consistent Query Answers in Virtual Data Integration Systems. Inconsistency Tolerance, LNCS 3300, 2004
M. Lenzerini Data Integration: a Theoretica Perspective. PODS2002