EXTRACTING INFORMATION FROM HETEROGENEOUS INFORMATION SOURCES USING ONTOLOGICALLY SPECIFIED TARGET VIEWS

Joachim Biskup[1], David W. Embley[2]

[1]Fachbereich Informatik, Universität Dortmund, 44227 Dortmund, Germany

[2]Department of Computer Science, Brigham Young University, Provo, Utah 84602, U.S.A.

March 28, 2001

Abstract — Being deluged by exploding volumes of structured and unstructured data contained in databases, data warehouses, and the global Internet, people have an increasing need for critical information that is expertly extracted and integrated in personalized views. Allowing for the collective efforts of many data and knowledge workers, we offer in this paper a framework for addressing the issues involved. In our proposed framework we assume that a target view is specified ontologically and independently of any of the sources, and we model both the target and all the sources in the same modeling language. Then, for a given target and source we generate a target-to-source mapping, that has the necessary properties to enable us to load target facts from source facts. The mapping generator raises specific issues for a user's consideration, but is endowed with defaults to allow it to run to completion with or without user input. The framework is based on a formal foundation, and we are able to prove that when a source has a valid interpretation, the generated mapping produces a valid interpretation for the part of the target loaded from the source.

1. INTRODUCTION

As the amount of information continues to explode and find its way into more and more repositories, we are faced with an ever increasing challenge to extract and integrate this information and make it useful for end-users. In light of these circumstances, we must search for ways to ease the extraction and integration process. Fully automated techniques may not be possible, but we should seek to automate as much as is possible, and we should seek for synergistic solutions that significantly aid extractors and integrators and yet demand as little as possible from them.

In light of the importance of this problem it is not surprising that much thought and effort has been expended in trying to resolve the issues involved. Many researchers have addressed this problem from several different vantage points, and the body of literature on this topic is large. Database researchers working on multidatabases one or two decades ago encountered many of these issues. This work is summarized in [21] with details provided in [42, 51]. The work in database schema integration also addresses these issues and has a longstanding research history. Much of this work was surveyed in 1986 [5], but continuing research has also led to many later results (e.g. [39, 8, 54, 30]). As the Web has become more prominent, a host of data-extraction work has appeared (e.g. [19, 44, 32]) and has been highlighted in several workshops (e.g. [1, 16, 55]). Beyond extraction, the desire to integrate results has led to a number of initiatives on integrating heterogeneous data from wrapped data sources [2, 37, 40, 20, 31, 33, 17, 38], including a theory-based comparison [56] between two of these initiatives, [33] and [37]. In the context of integration, researchers have studied ways to semi-automate the discovery of semantic relatedness [52, 14, 29, 50, 46, 36, 49, 48, 6], which is the central hard problem in automating information integration. Similar issues arise in integrating information for data warehouses [11, 12, 28]. A recent issue of *SIGMOD Record* included a special section on semantic interoperability that summarizes many of the results that have been achieved [6, 26, 47, 53, 27].

The work presented in this paper differs from previous work in its particular approach to the general problem. We focus on extracting information from heterogeneous sources for a particular predefined target view[†]. Although we do not denigrate the more traditional approaches to inte-

[†]In the parlance of [56], the target view acts as a mediator and the position our framework takes is "global-as-view."

gration (each has its place), we do point out several advantages to the approach we take. (1) Since a target view limits the scope of integration to a predefined collection of object and relationship sets, using a target view makes the problem more manageable than the general source integration problem. (2) The use of a target view coincides with the desire of many to integrate information from a variety of sources and present it in a personalized view for a user—see for example the work on structured maps [18]. (3) Since we map each source to the target independently, changes in a particular source affect only the mapping of that source to the target. Thus, the approach is scalable and can be applied to an environment such as the Web where scalability is paramount.

We define the target view conceptually, as an ontology [9, 10], independent of any of the sources. Others have suggested ontologies as a means to resolve many of the issues (see [47] and the earlier work cited in [47]), and ontologies are not without problems [45], but we believe that they offer one of the best approaches to resolving the issues.

We use OSM [24] as our conceptual model for describing an ontology for our target view. We also use OSM to model each source. Sources may be heterogeneous (e.g. legacy database systems, database systems under different models, semistructured text), but once we wrap a source in OSM, we have uniformity which provides us with a solid foundation from which to proceed with the work of extracting source information and loading it into the target view and with the work of integrating data from the various sources. OSM provides several advantages: (1) it captures conceptually the objects, relationships, constraints, and textual representations required in the target ontology and the source wrappers, and (2) it maps directly to a specialized subset of predicate calculus, which we can use for proving properties about our extraction and integration work. OSM can be seen as a high-level, graphical abstraction of a specialized subset of predicate calculus with augmentations to describe type characteristics for atomic tokens.

The work presented here also provides a broad framework into which the complementary work of others can nicely fit and be brought together. The framework specifies the process, identifies the issues and works with a user to resolve these issues, automates as much as possible, provides defaults so that the process can be fully automatic, catalogs the results including reasonable alternative results, and places confidence values on the results. We do not repeat the complementary work of others, but instead show where this work fits within our framework. At the same time, we do introduce some new ideas (1) for improved semantic matching, (2) for semantic transformations, and (3) for formal foundations. Specifically, for (1), the ontology we specify allows us to consider database values and context keywords for improving semantic matching. We describe these ideas here, but do not describe the work of others on using thesauri (e.g. [6, 49]) and on using structure (e.g. [48]) which we consider to be complementary because they also improve semantic matching and which we would suggest adapting for use within the framework we present. For (2), the semantic transformations we present focus on values, including value coercion, value decomposition and aggregation, object identifiers (OID's), and generalizations and specializations of value sets. We also provide more traditional transformations similar to those found in [54], but in terms of OSM, which has a two-component view of the world in terms of objects and relationships rather than the more prevalent three-component view in terms of entities, relationships, and attributes. OSM's two-component view serves to simplify the catalog of transformations. Finally, for (3), the formalisms we present are consistent with standard model-theoretic formalisms [4]. As such, we are able to build on the foundation laid by many years of solid theoretical work.

In all of this work and in our own work, the central, difficult issue is "semantic relatedness." Is the intended meaning of sets of objects in various sources or in a source view and a target view the same? If two related sets of objects do not have exactly the same meaning, how are they related—is one a generalization of the other, or do they overlap in some way? The same questions arise for sets of relationships except that the issues can be even more complex because relationship sets are often derived rather than given. Furthermore, similar questions arise at different levels of granularity: is an individual object or subobject identified in one source the same as an object or subobject identified in another source? When we attempt to automate the answers to these semantic questions, we only have "syntactic material" available to us. In attempting to answer semantic questions from syntactic clues, we usually assume that if the "expected syntactic properties" are satisfied, we have discovered a "semantic relationship." We must admit, however, that no

algorithm can ever decide with complete accuracy about the notion of semantic relatedness—a notion which is hidden in the mind of humans and subject to successful communication among them. Instead of giving up, however, we proceed synergistically, using syntactic properties (including mapping requirements, given and implied constraints, context keywords and data values—indeed, all available syntactic material) to infer semantic properties. We are modest enough to believe, however, that when the syntactic clues are "too poor" (e.g. are questionable, are contradictory, are ambiguous) we ask for human input. When asking, we try to do so synergistically, using available syntactic clues to explain the issues, to make a best default guess at a solution, and to suggest some alternative solutions.

In light of these challenges, opportunities, and precautions, we provide an overview for the rest of the paper and characterize the work we present here as follows.

Our primary focus is on generating a mapping that yields a way to extract data from a source OSM model instance and load it into a target OSM model instance.

- Section 2 defines target-to-source mappings over OSM model instances. The inverse of a target-to-source mapping leads to a procedure to load source information into a target scheme.

- Section 3 describes semantic matching for objects and relationships whose scheme declarations exist in both source and target specifications. The section discusses ways to match target object sets with source object sets and target relationship sets with source relationship sets. We base our matching on values and keywords, but do not exclude the possibility of using techniques developed by others. This section also discusses target-source mismatches in type-compatibility, cardinality-constraint compatibility, and generalization/specialization compatibility.

- In addition to direct correspondences between target and source object sets and between target and source relationship sets, target-to-source mappings allow for a variety of derived data, including missing generalizations and specializations, aggregated and decomposed values, object identifiers, and query-generated sets of objects and relationships. Section 4 provides these details.

- Trying to extract and integrate information from heterogeneous sources raises many issues, which may not be fully resolvable without human input. We isolate these issues and present them as issue/default/suggestion (IDS) triples. These IDS triples increase our ability to be focused in our interactions with a user and also provide a way to proceed in the absence of user input. These IDS issues arise naturally and are presented throughout our entire discussion.

- Given potential correspondences, as discussed in Sections 3 and 4, we present an algorithm to generate target-to-source mappings in Section 5. The mapping generation algorithm produces a set of tables that provides a way to record reasons for decisions and confidence measures. This provides for a clear elaboration and assessment of the process.

- In Section 6 we prove that if the source interpretation is valid, then our procedure always yields a valid interpretation for the part of the target model instance populated from the source. Our proof relies on much of the previous discussion. Indeed, although we do not write the paper in this style, Sections 2 through 5 constitute the "lemmas" for this proof. In Section 6 we summarize these "lemmas" in a table and provide the proof as a case analysis.

Our secondary focus is on two additional issues: (1) the issue of how to merge data from several sources, and (2) the issue of how to obtain OSM model instances for sources.

- Because we use OSM for both target and source modeling, we can provide a formal, solid way to merge sources. Nevertheless, there are a number of issues left to resolve. In Section 7 we raise these issues, but we leave their resolution for future work.

- The modeling approach we use provides a way to integrate a variety of heterogeneous sources, including databases of various types, data captured in exchange formats such as XML, semistructured data, and data-rich unstructured data, but we must first convert these sources to OSM model-instance views. In Section 8 we illustrate this initial conversion process by giving an example, leaving a full resolution to future work.

Two separate but closely related interests guide our presentation: (1) we introduce and discuss the steps of our procedure to generate target-to-source mappings, and (2) we prove that the procedure meets the desired formal postcondition for the target. Both of these interests require considerable detail, which must all appropriately come together. As a summary guide to (1), Figure 1 shows a global overview of the process. The grey areas depict areas to be filled in by the procedure. The paper thoroughly discusses all the components in Figure 1. As a summary guide to (2), we present in Section 6 a table giving a snapshot of the various cases that must be considered to prove that for any given source instance with a valid interpretation, we can produce a populated target instance with a valid interpretation.

2. TARGET-TO-SOURCE MAPPINGS—DEFINITIONS

Our objective is to obtain a populated target model instance that represents facts found in one or more sources. We achieve this objective by producing a collection of mappings, one for each source, and then merging the results of these mappings. Each function in the collection maps the target model instance to one of the source model instances, such that the inverses of these mappings determine which source facts become which target facts. To make this work, the functions must have several restrictive properties and need to correspond semantically to the meanings intended in the target and sources. The generator that produces these target-to-source mappings provides a way to satisfy these requirements. The generator also provides a way to measure the credibility of each of the ordered pairs in the individual mappings and thus a way to measure the overall credibility for the collection of mappings.

We begin in this section by first describing model instances. We then define mappings between model instances and begin to enumerate our requirements for these mappings.

2.1. Model-Instance Description

We use OSM [24, 22] to represent the target and the sources for our mappings. An OSM model instance includes a set of object sets and a set of relationship sets. The union of these two sets in a target model instance constitutes the elements of the domain for our mappings. The co-domain for any particular source in a target-to-source mapping also includes a union of the object and relationship sets, but it additionally includes any needed derived object sets and relationship sets. In addition to providing the elements of the domains and co-domains for our mappings, OSM provides predicates, with which we can state the facts of interest, and a restricted set of predicate-calculus formulas, with which we can state and check the integrity constraints of the various model instances.

Figure 2 shows a graphical representation of the OSM model instance we use as the target in our sample application for this paper. Each labeled rectangle represents an object set. *Country* and *Travel Photo : Image*, for example, are object sets. The part of the label to the left of a colon, or the entire label if there is no colon, is the name for the object set. We elide spaces in an object-set name when we need the name to be an identifier. *TravelPhoto*, for example, is the identifier for the name of the object set labeled *Travel Photo : Image*. The part of the label to the right of the colon is the type.[†] The default type is *String* if the object set is displayable (denoted by a dotted rectangle) and is *OID*, standing for Object IDentifier, if the object set is nondisplayable (denoted by a solid rectangle). In Figure 2 *Location* is nondisplayable, and all other object sets are displayable.

[†]In this paper, "type" denotes only an intensional set of objects (or values).

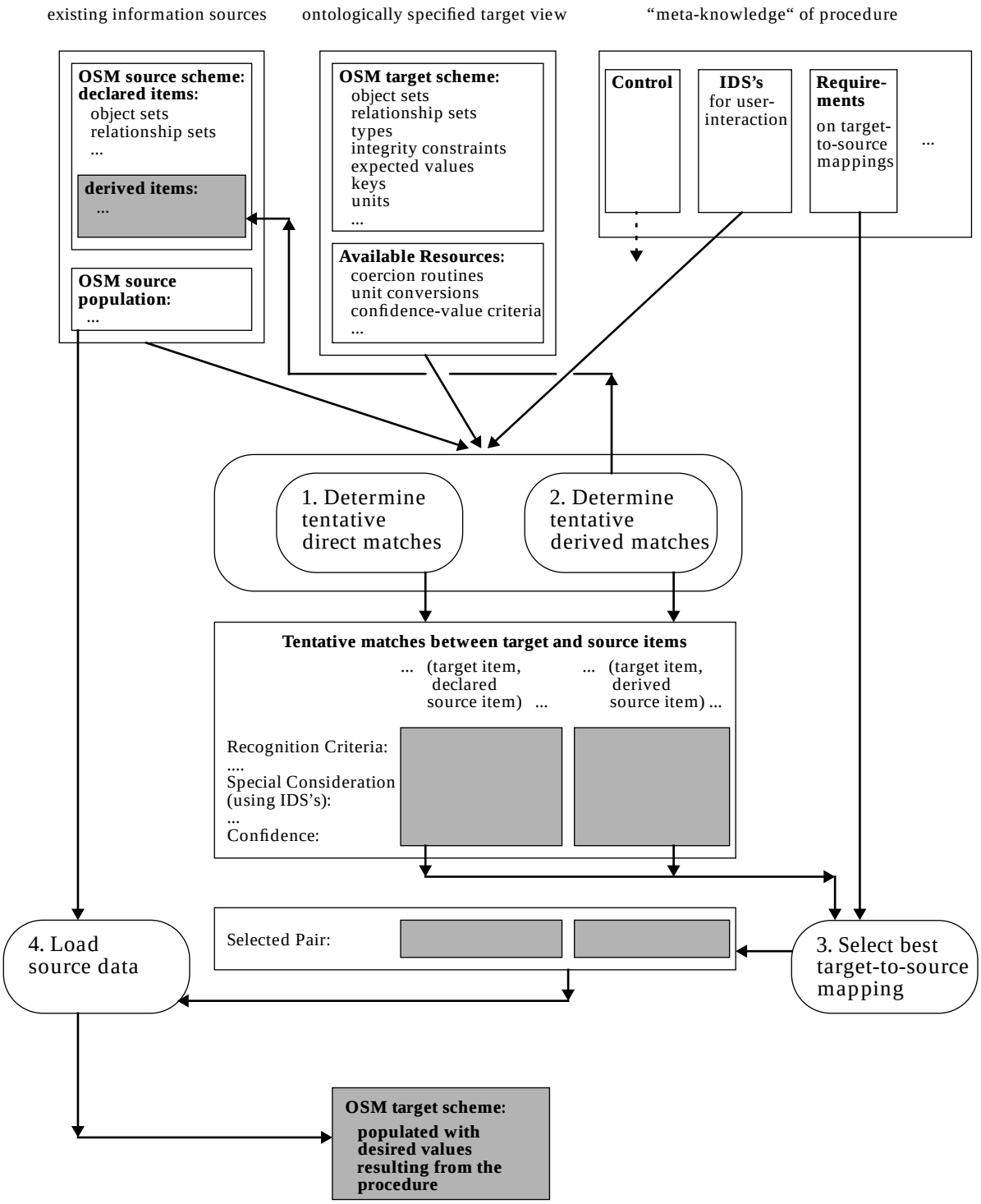

Fig. 1: Global overview.

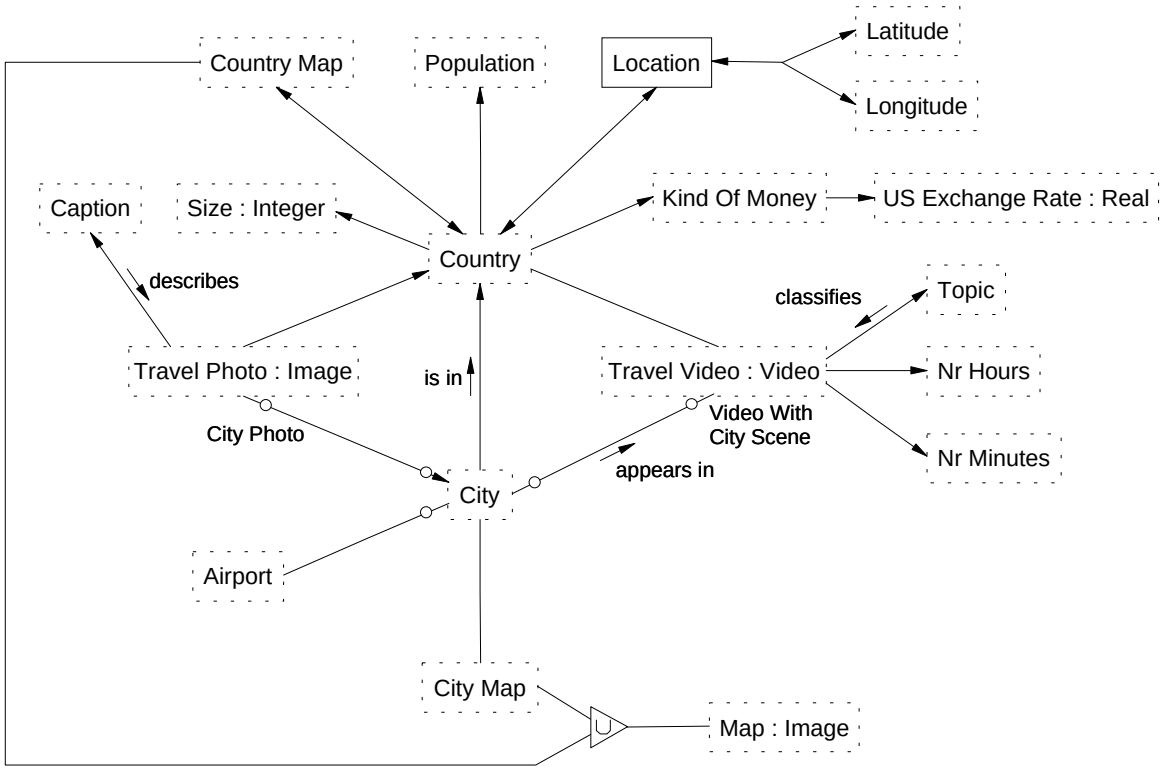

Fig. 2: Graphical representation of target model instance.

Lines that connect object sets represent relationship sets. Relationship sets may be binary or n-ary, $n > 2$. A binary relationship set may have a label with a reading-direction arrow. In this case the name of the relationship set is an ordered, space-separated, concatenation consisting of the object-set name on the tail side of the reading-direction arrow, the label associated with the reading-direction arrow, and the object-set name on the head side of the reading-direction arrow. *City is in Country* and *Topic classifies Travel Video* are relationship sets in Figure 2. A name for an n-ary relationship set must include the names of all its associated object sets. Relationship sets without labels have default names: one of *has* or *is for*, between the object set names for binary relationship sets, or a space-separated concatenation of the associated object-set names in alphabetical order for either binary or n-ary relationship sets. *Latitude Location Longitude*, for example, is the default name of the ternary relationship set in Figure 2. To make diagrams less cluttered, we typically do not specify default *has* and *is for* names for binary relationship sets in the graphical representation, only in the textual representation (which we present next). When we need relationship-set names to be identifiers, we elide spaces in object-set names and replace the remaining spaces by an underscore character.

Figure 3 shows the textual representation for the target model instance. The textual representation provides the full specification, but the graphical representation is often better for exposition. In addition to specifying all object sets, relationship sets, and constraints in the graphical representation, the textual representation provides the default names for binary relationship sets, allows the specification of expected values and keywords for object and relationship sets, and provides units for potential unit-conversions. We specify sample values and keywords we expect to see in the sources by regular expressions (using Perl-like syntax). Since an or-separated list is a regular expression, we may specify a simple list of possible values, such as the sample list of countries in Line 2 of Figure 3. An *(i)* that precedes a regular-expression specification, as it does in Line 3, denotes that the regular expression is case insensitive. In our example, Line 3 actually need not be included, because it is the default specification for keywords—a case insensitive expression

```
Country
   values { France | Germany | Italy | USA | United\s*States }
   keywords (i) { Country }
end;
Population values { [1-9]\d{0,2}\,\d{3}(\,\d{3})? } end;
Country [1] has Population [1..*]
   values (i) { < USA , [2,3]\d{2}\,d{3}\,\d{3} > < Germany , [6-9]\d\,\d{3},\d{3} > }
end;
Location : OID keywords (i) { (Center)?\s*Location } end;
Country [1] has Location [1];
Longitude
   values { [1-9]\d{0,2}\s*{E | W} }
   keywords (i) { Longitude | Long\.? }
end;
Latitude
   values { [1-9]\d{0,2}\s*{N | S} }
   keywords (i) { Latitude | Lat\.? }
end;
Latitude [1..*] Location [1] Longitude [1..*];
Latitude, Longitude -> Location;
Country Map [1] is for Country [1];
Size : Integer
   values { [1-9]\d{5,9} | [1-9]\d{0,2},\d{3}(,\d{3}) }
   keywords (i) { (sq | square)\s*((mi | miles) | (km | kilometers)) }
   units { square miles }
end;
Country [1] has Size [1..*];
Kind Of Money values { DM | Mark | Peso | Pound } end;
Country [1] has Kind Of Money [1..*];
US Exchange Rate : Real
   values { \d{1,5}\.\d{1,5} }
   units { US$ per KindOfMoney }
end;
US Exchange Rate [1..*] is for Kind Of Money [1]
   values { < \d{1,5}\.\d{1,5} , DM | Mark | Peso | Pound > }
end;
Travel Photo : Image keywords (i) { Photo | Picture } end;
Travel Photo [1] is for Country [1..*];
Caption;
Caption [1..*] describes Travel Photo [1];
City Photo : Travel Photo;
City values { Berlin | New\s*York | Paris } end;
City Photo [1] is for City [1..*];
City [1] is in Country [1..*]
   values { < Berlin , Germany >, < New\s*York , USA | United\s*States >, < Paris , France > }
end;
Airport values { ATL | JFK | FRA } end;
Airport [1..*] is for City [1..*]
   values { < ATL , Atlanta > < JFK , New\s*York >, < FRA , Frankfurt > }
end;
City Map [1..*] is for City [1..*];
Travel Video : Video
   keywords (i) { Video }
end;
Travel Video [1..*] is for Country [1..*];
Topic keywords (i) { Subject } end;
Topic [1..*] classifies Travel Video [1];
Nr Hours
   values (i) { [1-9](h\.? | hr\.? | hrs\.? | hours)? }
   keywords (i) { Length | hrs\.? | hours }
   units { hours }
end;
Nr Hours [1..*] is for Travel Video [1];
Nr Minutes
   values (i) { ([1-5][0-9] | [1-9])(m\.? | min\.? | minutes)? }
   keywords (i) { Length | min\.? | minutes }
   units { minutes }
end;
Nr Minutes [1..*] is for Travel Video [1];
Video With City Scene : Travel Video;
City [1..*] appears in Video With City Scene [1..*];
Map : Image;
Country Map, City Map ISA(union) Map;
Map [1] has Map Name [1:*];
```

Fig. 3: Textual representation of target model instance.

consisting only of the name of the object or relationship set. To specify tuples in a relationship set, we use angle brackets with commas to separate the regular expressions. The *Country has Population* relationship set starting in Line 6 shows an example. Observe here how we use regular expressions to be imprecise about the exact population of a country, but at the same time to be more precise about the values we expect to see than just allowing some arbitrary integer of any size. For potential unit conversion, we provide units for target values (e.g. *square miles* for *Size* in Line 25. We assume that standard unit-conversion tables are available for all units of interest.

OSM model instances allow the specification of integrity constraints. Bracketed numbers and min-max ranges in relationship declarations are participation constraints. *Country [1] has Population [1..*]* (Line 6 in Figure 2), for example, declares that country values participate exactly once in the relationship set and that population values participate at least once, but have no designated maximum participation. When participation constraints specify that an object in an object set participates at least once, or more generally, at least a specified nonzero number of times, we say that the participation is *mandatory*. A zero-minimum, on the other hand, lets the participation be *optional*. In the graphical notation, we denote optional participation by the letter "o", which appears as a small circle on a relationship set's connection to an object set. Airports, for example, are optional for cities as designated by the "o" in Figure 2 next to *City* for the *Airport is for City* relationship set.

When objects in an object set S participate at most once in a relationship set R, S is a *key* for R. We are thus able to derive functional dependencies (FD's) from participation constraints. These derived FD's appear as directed edges in the graphical notation; thus, for example, the edge from *Country* to *Population* in Figure 2 is directed. We may also directly declare FD's for relationship sets, although we normally only declare those that cannot be specified with participation constraints—those with compound left-hand sides or those whose set of mentioned object sets is a proper subset of the object sets of a relationship set. The FD *Latitude Longitude* $\rightarrow$ *Location* in Figure 3 (Line 20) is an example of a specified FD.

A colon denotes an *ISA* constraint.[†] In both Figures 2 and 3 we have *Travel Photo : Image*, which declares that a travel photo is an image. A triangle in the graphical notation, which corresponds to "ISA" in the textual notation (Line 73 in Figure 3), also denotes an *ISA* constraint. An *ISA* constraint requires one set of objects (called a *specialization*) to be a subset of another set of objects (called a *generalization*). The graphical triangle notation and textual ISA notation also allow us to state additional constraints among object sets. The "$\cup$" in the triangle in Figure 2 constrains *Map* to be a union of *Country Map* and *City Map*. Other ISA constraints are "+" for a mutual exclusion among specializations and "$\uplus$", a combination of "$\cup$" and "+", for a partition. Several (one or more) ISA constraints may form a collection of ISA constraints, called an *ISA hierarchy*. Although OSM does not require an ISA hierarchy to be a tree, for our work here we consider only ISA hierarchies that are trees.

Observe in Figure 3 (Line 41) that we have *City Photo : Travel Photo*, declaring the set of city photos to be a subset (a specialization) of the set of travel photos. In the graphical notation in Figure 2 *City Photo* appears as a *role*. In a populated OSM model instance, a *role* of an object set S for a relationship set R denotes the set of objects in the projection on the S objects of the relations in R. Since referential integrity always holds for populated OSM model instances, however, this implies the simpler definition that a *role* for a relationship set R connected to an object set S denotes the subset of objects of S participating in relations of R. In our example, the travel photos related to a city are exactly the subset of the travel photos that are city photos. Moreover, since the role *City Photo* is a set of objects, we consider it to be an object set and thus we name the connecting relationship set *City Photo is for City*. When roles are present, we use the role name, rather than the connecting object-set name, in the relationship-set name.

Figures 4, 5, and 6 are the source OSM model instances for our sample application. Observe in Figure 4 that roles provide convenient a way to resolve the meaning of cyclic relationship sets, which connect two or more times to the same object set. We require all but one of the connections to the same object set to have a role. Since the user controls the target, the target can always

[†]In keeping with our notion for this paper that "type" merely denotes a set of objects, *ISA* denotes only a subset constraint (nothing less or more).

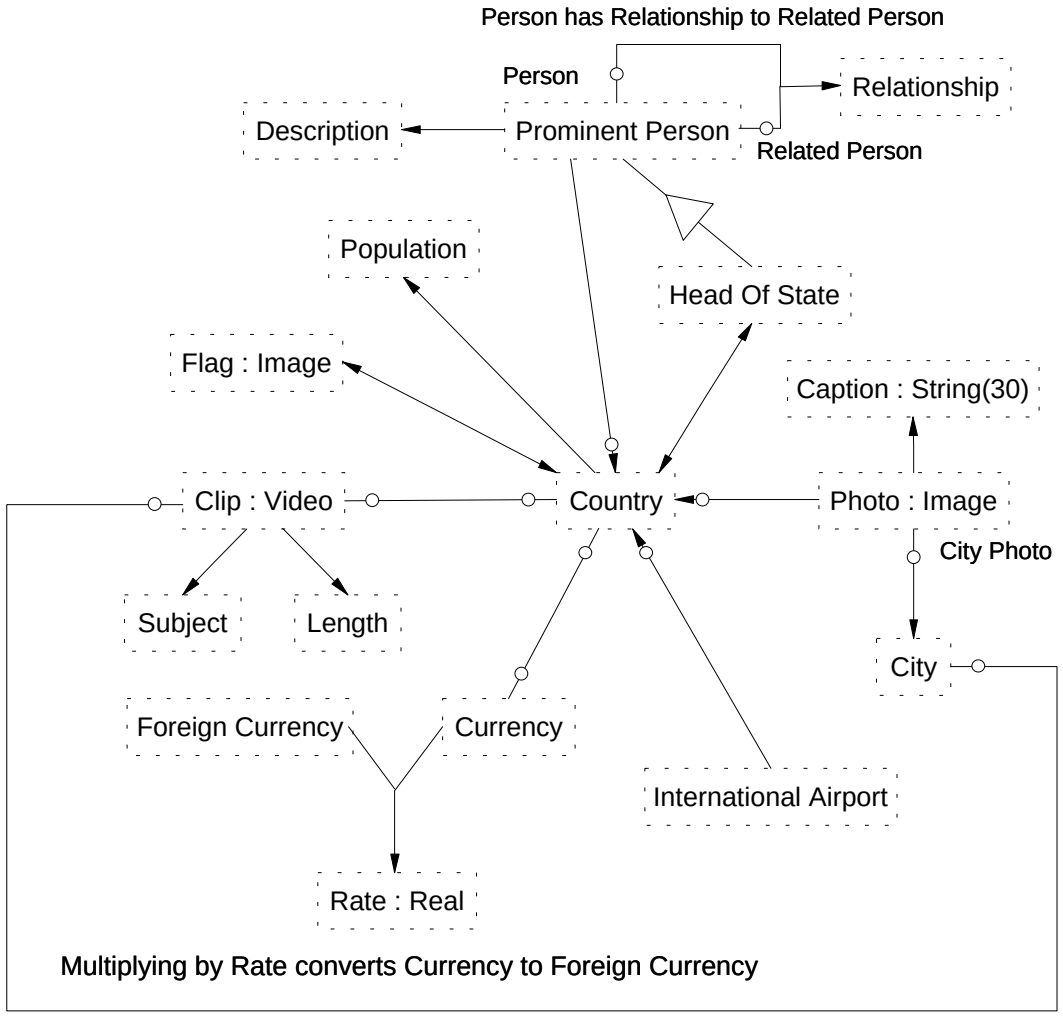

Fig. 4: World Countries—source model instance.

satisfy this requirement. For sources, which the user does not control, we can always derive object sets for any needed roles and thus satisfy this requirement for sources as well.

Source model instances have data, obtained from the database instances they represent. Figure 7 shows a (partial) sample database instance for the model instance in Figure 4. For an OSM model-instance database, each object set and each relationship set is a table. If we let the names of the object and relationship sets be predicate identifiers, we immediately obtain the *ground facts* with respect to the source database. According to the database instance in Figure 7, some facts for the model instance in Figure 4 are *Country(Canada)*, *Country(Germany)*, *Country_Population(USA, 280,000,000)*, *Currency(DM)*, and *Currency_ForeignCurrency_Rate(DM, US$, 0.5)*.

We can express queries over the ground facts to derive other facts, called *derived facts*. We call a set of facts derived by a query whose result has a single attribute a *derived object set*. For example, we can produce a role object set *Video With City Scene* for the connection to *Clip* in the *City Clip* relationship set in Figure 4 by the query $\rho_{Clip \leftarrow VedeoWithCityScene}\pi_{Clip}City\_Clip$. We call a set of facts derived by a query whose result has two or more attributes a *derived relationship set*. For our application, the attributes must be identifiers of object sets in the model instance. The name of a derived relationship set is often its default name, which can be renamed if desired or if necessary to distinguish a derived relationship set from any other (derived or given) relationship

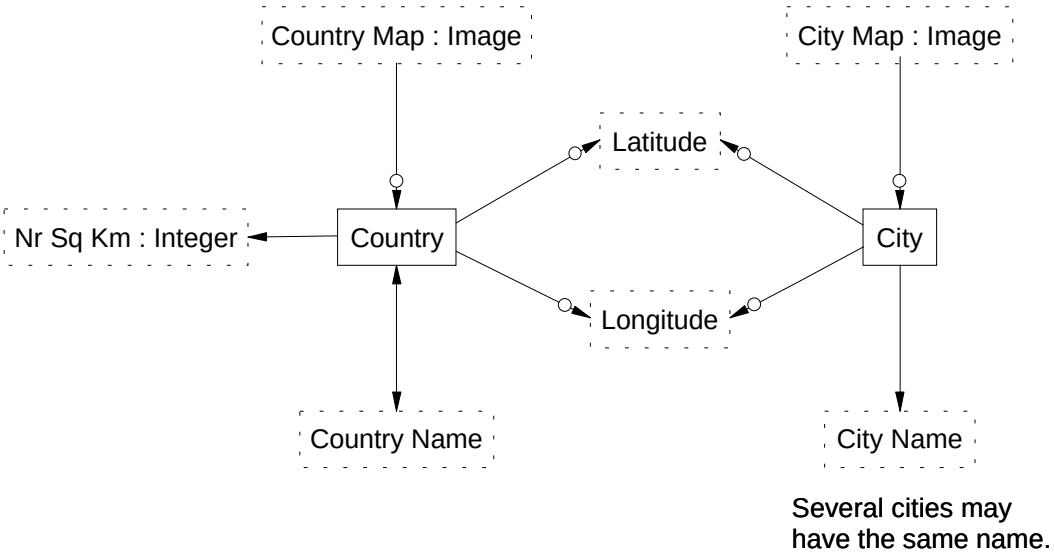

Fig. 5: World Maps—source model instance.

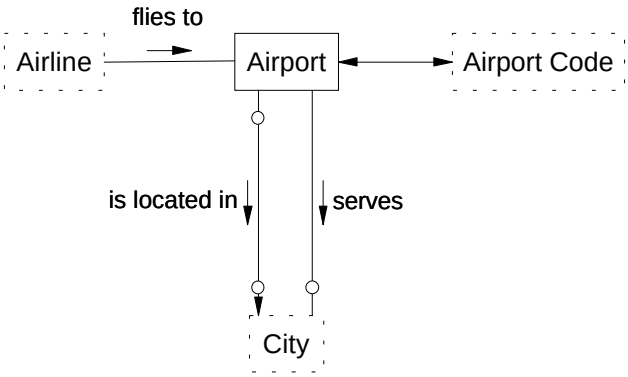

Fig. 6: World Airports—source model instance.

```
Country          Population          Country  Population
-------          ----------          -------------------
Canada            82,000,000         Germany  82,000,000
Germany          280,000,000         USA     280,000,000
Mexico
USA

Currency          ForeignCurrency          Rate
--------          ---------------          ----
DM                DM                       0.5
US$               US$                      2.0

Currency ForeignCurrency Rate          Length
-----------------------------          ------
DM       US$               0.5         1 hr 15 min
US$      DM                2.0         35 min
                                       1 hr 30 min
                                       2 hrs

City          InternationalAirport          Country   InternationalAirport
----          --------------------          -------------------------------
Atlanta       FRA                           Germany   FRA
Berlin        JFK                           USA       JFK
Frankfurt
London
New York

...
```

Fig. 7: A (partial) sample database instance.

set in the model instance. For Figure 4, for example, the query $\pi_{City\ Country}(Country\_Photo \bowtie \rho_{CityPhoto \leftarrow Photo} City\_CityPhoto)$ produces a derived relationship set between *City* and *Country*.

We can write the integrity constraints over the ground facts as predicate-calculus formulas. (See [24] or [22] for a complete explanation.) For example,

$$\forall x \forall y (Country\_Population(x, y) \Rightarrow Country(x) \wedge Population(y))$$

is a referential-integrity constraint,

$$\forall x (CityPhoto(x) \Rightarrow Photo(x))$$

is a subset constraint, and

$$\forall x (Country(x) \Rightarrow \exists^1 y Country\_Population(x, y))$$

is a participation constraint, where "$\exists^1 P(x)$" denotes that "there exists exactly 1 x such that P holds."

A populated model instance M with its predicates and closed predicate-calculus formulas, as defined here, constitutes an *interpretation* for M. If all the closed formulas hold for the interpretation, the interpretation is a *model* in model theory [4]. Because we are already using the term "model" in several ways, we choose to call a model, in the model-theoretic sense, a *valid interpretation*.

2.2. Target-to-Source Mappings

We seek a way to produce a valid interpretation for a given target model instance based on an (assumed) valid interpretation for a source model instance. In this subsection we define what

we mean by a mapping whose inverse can provide the basis for transforming source facts into target facts. In later sections we provide a way to produce these mappings so that the target facts obtained constitute a valid interpretation for the target.

Each function f, in the set of mappings we seek, maps an OSM target model instance t to an OSM source model instance s. The domain of f consists of the union of the object-set names and the relationship-set names in t. The co-domain of f consists of the union of the object-set names in s, the relationship-set names in s, and the names of any needed derived object sets and relationship sets in s. We need a derived object set or a derived relationship set if and only if the execution of the target-to-source mapping generator produces a derived object set or a derived relationship set for s. Our initial requirements for f follow.

Req. 1 *f must be a (partial) injective function.*

Req. 2 *If $\langle a,\ b\rangle$ is an ordered pair of f, a and b must both be object sets or both be relationship sets.*

Req. 3 *If $\langle a,\ b\rangle$ is an ordered pair of f and a and b are relationship sets, a and b must have the same arity.*

Req. 4 *Referential-integrity guarantee: If $\langle a,\ b\rangle$ is an ordered pair of f, a and b are relationship sets, a is the hyperedge $\{a_1,\ \ldots\ a_n\}$, and b is the hyperedge $\{b_1,\ \ldots,\ b_n\}$, then if g is the restriction of f to $\{a_1,\ \ldots\ a_n\}$, the range of g must be $\{b'_1,\ \ldots,\ b'_n\}$ where $b_i \subseteq b'_i$, $1 \leq i \leq n$. Here, $b_i \subseteq b'_i$ if $b_i = b'_i$, if $b_i\ ISA\ b'_i$, or if there exist object sets $c_1,\ \ldots,\ c_k$ such that $b_i\ ISA\ c_1\ ISA\ \ldots\ ISA\ c_k\ ISA\ b'_i$, for $k \geq 1$.*

Normally, a function f will be partial[†], although it certainly may be total. Thus, we are usually seeking to obtain n partial functions for n sources such that the preimages of these partial functions cover the target domain. Sometimes we may not even be able to cover the target domain, in which case we will be unable to populate some part of the target model instance. As a general notion of where we are headed, consider as an example the ordered pairs $\langle Caption,\ Caption\rangle$, $\langle Travel\ Photo,\ Photo\rangle$, and $\langle Caption\ describes\ Travel\ Photo,\ Caption\ Photo\rangle$, which may be part of a function f mapping the target in Figure 2 to the source in Figure 4. The inverse function f^{-1} tells us to load *Caption* facts in the target from *Caption* facts in the source, to load *TravelPhoto* facts from *Photo* facts, and to load *Caption describes Travel Photo* facts from *Caption Photo* facts.

Observe that we have an asymmetry in the domain and co-domain for our function. The domain allows only object and relationship sets whereas the co-domain, in addition to object and relationship sets, allows derived object and relationship sets. This asymmetry reflects our assumption that the target model instance is fixed and atomic. We make this assumption because the purpose of our application is to populate the target with data, and moreover to populate the target with data as desired by the user (or client of the user). We thus assume that the user knows what is wanted in the target model instance[‡]. This assumption has the consequence that when there are mismatches in a target to source mapping, we make "changes" in the source, not the target. Since we usually have no authority or ability to change any source, these changes must be virtual. Indeed, these virtual changes are precisely why we need derived object and relationship sets in our sources[§].

[†]We do not concern ourselves further with designating these functions as being partial, except to ensure clarity where necessary.

[‡]In making this assumption, we are not saying that a target model instance can never change, only that during the time we generate target-to-source mappings, it does not change. A user may decide, for example as a result of seeing a potential loss of data from a source, that the target should be altered. In this case, we either backtrack or restart the target-to-source generation algorithm.

[§]These virtual changes are in the spirit of previous theoretical work on scheme inclusion and translation schemes. (See, for example, [3, 7, 35, 43].) In this work researchers investigated conditions for one scheme (here, the source model instance) to be appropriate to *support* another scheme (here, the target model instance). In this context, "support" essentially means that the source scheme can be mapped by queries (here, derived object and relationship sets) to structures corresponding to the target scheme, or, equivalently, that queries from the target scheme can be posed against the source scheme.

3. MATCHING RULES—EXISTING OBJECT AND RELATIONSHIP SETS

To produce a proper functional correspondence f between a target t and a particular source s, we must respectively match object and relationship sets in t with existing or derived object and relationship sets in s. In addition to the basic requirements enumerated in the previous section each matching pair $\langle a,\ b\rangle$ in f must satisfy certain syntactic and semantic requirements. In this section we address the problem of satisfying syntactic requirements by considering type compatibility and constraint compatibility, and we address the problem of satisfying semantic requirements by considering context keywords and data values. In the next section we consider derived object and relationship sets.

3.1. Type Compatibility

We require the following basic restriction for type compatibility.

> **Req. 5** *Let f be a mapping from a target t to a source s. If $\langle a,\ b\rangle$ is an ordered pair of f for object sets a and b and the type of object set a is type(a) and the type of object b is type(b), there must exist an agreed-on (possibly trivial) conversion function c such that c converts values of type(b) to values of type(a).*

Requirement 5 ensures that we can extract values from a source object set and load them into a corresponding target object set. This requirement holds for both displayable and nondisplayable types. We first consider only displayable types and then extend the discussion to nondisplayable types.

Displayable Types

To aid in satisfying Requirement 5 for displayable types, our mapping generator requires a type hierarchy as auxiliary input. The type hierarchy is a partial ordering based on semantic domain inclusion; that is, it is based semantically on *ISA* so that, for example, $Integer \subseteq Real$, independently of how the sets *Integer* and *Real* are represented in an implementation. For the mapping generator described in this paper, we assume (1) that the hierarchy is a tree (or a forest of trees), (2) that it includes (at least) all types found in both target and source model instances, and (3) that default coercion routines exist (or can be created when needed) in both directions for each *ISA* edge in the type hierarchy. Figure 8 shows a possible type hierarchy for our sample application.

Given an ordered pair $\langle a,\ b\rangle$ of object sets for a target-to-source mapping with $type(a)$ and $type(b)$ both in the established type hierarchy for displayable types, there are four possible subset/superset relationships: (1) $type(a) = type(b)$, (2) $type(a) \supset type(b)$, (3) $type(a) \subset type(b)$, and (4) $type(a) \not\sim type(b)$, where $\not\sim$ denotes that none of (1) through (3) is satisfied. Each of these cases leads to different possibilities, which we now discuss.

Case 1, $type(a) = type(b)$. Initially, this case appears to be straightforward because we can trivially satisfy Requirement 5. Since the types are the same, we can simply load the values in the source object set into the target object set. However, making the types the same does not imply that the semantics are the same. Just because *Airport* in Figure 2 and *Head Of State* in Figure 4 are both typed as *String* does not mean that airports and the names of government leaders are semantically the same. This, of course, is the main issue we address in Section 3.4.

Besides these obvious semantic differences, it is also possible for the types to be the same and for the semantics to be "the same," but for which it is still incorrect to simply copy source values as target values [50]. Consider, for example, *Size* in Figure 2 and *Nr Sq Km* in Figure 5, which are both integers, and assume that the units are square miles and square kilometers respectively. Whenever the types involve units, the units need to be checked.

In general, we provide a way for the user to check assumptions and make alterations when necessary through specific statements and questions directed to the user. We denote these requests for user insights, clarifications, or qualifications by **IDS i**. In general, an IDS consist of three statements: (1) a statement that explains the issue, I, (2) a statement that explains the default action, D, and (3) a suggestion, S, about what to do to resolve the issue. We note that the wording

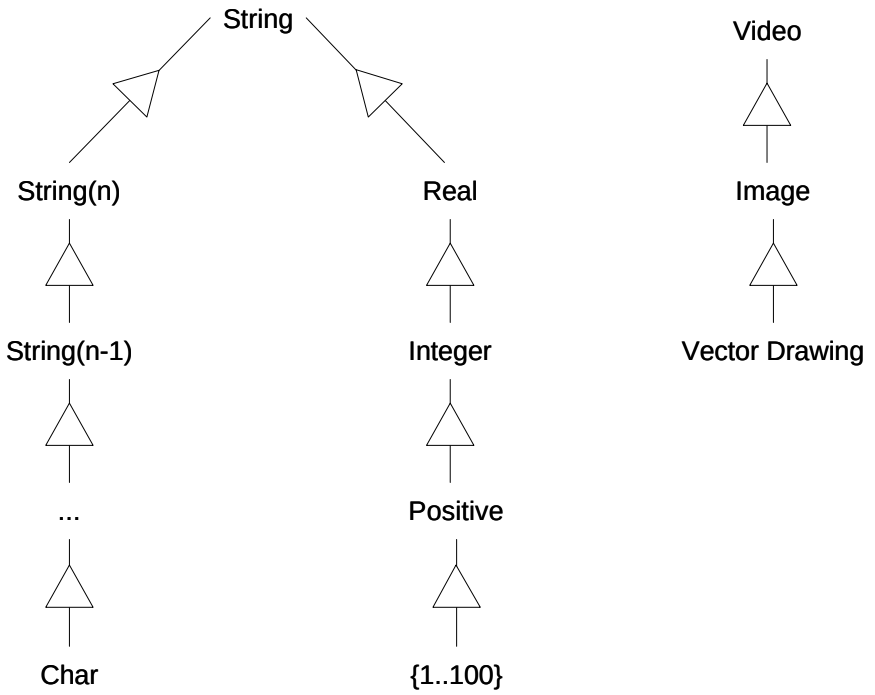

Fig. 8: Sample type hierarchy.

of IDS's can be adjusted to suit the taste of various user groups. Here we provide the wording in terms of the vocabulary and ideas presented in this paper.

For Case 1, when the target specifies units, we issue the following IDS.

> **IDS 1** ***Issue****: The target type has units and may need a unit-conversion routine for transforming source values to target values.* ***Default****: If no conversion routine is specified, no unit conversion will take place.* ***Suggestion****: If a unit-conversion routine is required, please specify which conversion routine to use.*

We assume that all standard unit conversions are readily available, so that the user only needs to select one. We also point out that whenever we need to load a target object set that has units, we pose this question, not just when the types are the same. In a sense, once the coercion is performed, the types *are* the same, and the question of unit conversion for identical types still remains.

Besides units, another way for the types to be the same and for the semantics to be "the same," but for which simply copying source values as target values may not give expected results is to have values in formats that are different than what is wanted or expected. The date "01/02/2000", for example, may be "January 2, 2000" or "1 February 2000". For extraction from a single source, this may not present much of a problem because the values should, at least, be consistent among themselves. When we extract values for an object set from several sources, however, we may obtain values that represent the same object but are not equal (synonyms) or values that represent different objects but are equal (homonyms). Some work has been done on normalizing values (e.g. [34]), and we may be able to adopt or adapt this work, but this is not an issue we resolve in this paper.

Besides units and value normalization, there are still more difficulties in some application areas. Scientific work involves granularity of results, scientific ontologies, and a host of other concerns [26]. [53] contains an interesting discussion of some military issues and includes as an example altitude, which for spacecraft is the distance above the center of the earth, but which for aircraft is the distance above the surface of the earth. Many of these issues are currently under investigation, but, for the most part, they remain open research issues.

Case 2, $type(a) \supset type(b)$. For any *ISA* relationship (direct or indirect) in our partial ordering on types, we can always naturally coerce a specialization value in a source to a generalization value in a target because an object in a specialization semantically *is* an object in the generalization. In our type hierarchy in Figure 8, for example, we can coerce a *Char* to be a *String(2)* by appending a space, and we can coerce an *Image* to be a *Video* by having every frame be the same image.

The more interesting question for Case 2, however, is will the results be what a user expects? Presumably, a user expects a value with greater discriminating power among the objects than the source provides. A *Real* distinguishes more number objects than does an *Integer*, and a *Video* provides more viewing possibilities than an *Image*. For some applications this may matter, and for others this may not matter. Thus, when Case 2 arises, we issue the following IDS.

> **IDS 2** ***Issue****: The target type has greater discriminating power than the source type.* ***Default****: Coercion routines will add arbitrary additional discriminating information to source values.* ***Suggestion****: If this is not acceptable, a different source object set, most likely in a different source, should be found.*

Case 3, $type(a) \subset type(b)$. The coercion for this case, when loading from source to target, may or may not be natural. We can truncate strings, round off reals to integers, and choose an arbitrary frame from a video to create an image. The coerced value in the target represents an equivalence class of values in the source. A truncated string stands for the original string (but also for all other strings padded to the original length), the integer rounded from a real approximates the real (but also all others that round to the same integer), and an image from a video is a representative of the many possible frames that could have been chosen. Since a user may know a better way to choose a representative for an equivalence class, we provide the following IDS.

> **IDS 3** ***Issue****: The target type has weaker discriminating power than the source type.* ***Default****: The default coercion routines select some representative value from among the many possible values.* ***Suggestion****: You may wish to specify your own coercion routine.*

Case 4, $type(a) \not\sim type(b)$. Coercion for Case 4 may make no sense for an application; for example, converting from *Vector Drawing* to *Char*. Often, however, such a conversion may indeed be wanted. For example, we often convert a page of text stored as an *Image* to a *String* by means of optical-character recognition, or convert a *Char* representing a digit to an *Integer*.

For this case, we may be able to use the default coercion routines by converting a source value from $type(b)$ to a common ancestor of $type(a)$ and $type(b)$ and then converting from the common ancestor to a target value of $type(a)$. This, however, may not yield the desired result. The *Integer* 2, for example, converted in this way to *String(5)* might be "2.000" rather than the expected " 2".

Since a user will most likely either want to reject the pair $\langle a,\ b\rangle$ as a possible pair in the target-to-source mapping or provide or choose a tailor-made conversion routine, we issue the following IDS.

> **IDS 4** ***Issue****: There is a mismatch between the type of the target object set, <target object-set name>, whose type is <type(a)>, and the type of the source object set, <source object-set name>, whose type is <type(b)>.* ***Default****: If there is no common ancestor type in the type hierarchy, the pair is rejected; otherwise the system uses the default conversion routines from <source object-set name> to the common ancestor <common ancestor type> and then from <common ancestor type> to <target object-set name>.* ***Suggestion****: If you want to load <target object-set name> values from <source object-set name>, you may wish to specify a conversion routine; otherwise, reject the pair.*

To make the task easier, some common conversion routines can be provided, such as converting back and forth between length-bounded strings and numbers.

Nondisplayable Types.

For an ordered pair $\langle a,\ b\rangle$ in a target-to-source mapping with at least one nondisplayable object set, either a and b are both nondisplayable or one is and the other is not. The later two cases, where either a is displayable and b is not or a is not displayable and b is, may require derived object sets for their resolution (if indeed there is a resolution). We therefore discuss these cases, along with other cases requiring derivations, in Section 4.

When both a and b are nondisplayable, we satisfy Requirement 5 because $type(a) = type(b) = \text{OID}$. Nevertheless, we may still have to exercise some caution for a single target-to-source mapping, and we can encounter extremely difficult object-identity problems when we have multiple target-to-source mappings that include the pair $\langle a,\ b\rangle$. We do not address these multiple-source, object-identity problems, but do address OID conversion for a single source.

The OID's in the source and target may have different representations, but, because they are OID's, we can be sure that they are in a one-to-one correspondence with the objects they represent and that the specific values chosen have no particular meaning other than to stand for the objects they represent. Since we can always form a one-to-one correspondence using any reasonable chosen representation for target OID's, we can always convert source OID values to target OID values. (We are assuming, of course, a sufficiently large set of target OID values.) This conversion is useful even if the representations are the same because the target then has control not only over the choice of representation but also over the choice of value. We can thus satisfy the following requirement

> **Req. 6** *The agreed-on (possibly trivial) conversion function(s) that convert values for nondisplayable types must ensure that objects have the same OID only if they represent the same real-world object.*

We would like to be able to satisfy the stronger if-and-only-if requirement, but, as stated earlier, we do not resolve the object-identity problem here. We can satisfy the weaker requirement by assigning a different target OID to every encountered object unless we know that the objects are the same because they are identical object instances from the same source.

3.2. Relationship-Set Constraint Compatibility

Constraint requirements for relationship sets fall into two basic categories: (1) type requirements needed to satisfy referential-integrity constraints and (2) predicate-calculus constraints specified for relationship sets in the target model instance. We discuss each in turn.

Type Requirements

Requirement 4, for referential integrity, does not require matching relationship sets in a target-to-source mapping to have matching object sets; it only requires the object sets of a relationship set in the source to be *ISA* subsets of matching object sets. For nondisplayable object sets, translation is straightforward. For displayable types, however, the requirement leaves open the possibility of a type incompatibility and thus the possible necessity to coerce the values in some of the connecting object sets before loading the relations of a source relationship set into a target relationship set.

Let $\langle r,\ r'\rangle$ be an ordered pair of relationship sets in a target-to-source mapping f. Let a be an object set connected to r and a' be an object set connected to r'. Let $\langle a,\ a''\rangle$ be in f and assume, as stated in Requirement 4 that there are one or more *ISA*'s connecting a' and a'' so that $a' \subseteq a''$. The types of a, a', and a'' can all be different. As a concrete example, let $type(a)$ be *String*, $type(a')$ be *Integer*, and $type(a'')$ be *Real*. From our discussion in Section 3.1, we may assume that the type incompatibility between a and a'' has been resolved, e.g. that we have a routine to convert the reals in a'' to strings before loading them into a. If we naively load r from r', however, we will load integers where we are expecting strings. As a resolution, we should first convert the integers in a' to strings.

In general, since $a' \subseteq a''$, there is a default coercion (possibly a sequence of default coercions) from a' to a''. Thus, it is reasonable to assume that a composition of these coercions plus the final coercion that converts values of a'' to values of a provides a reasonable way to convert a' values to a values. Other alternatives, including in particular, a direct conversion are also possible. Thus, when the type of a differs from the type of a', we issue the following IDS.

IDS 5 ***Issue****: To load the target relationship set <target relationship set> from the source relationship set <source relationship set>, the type of source object set <connected source object set> must coerce to the type of target object set <connected target object set>.* ***Default****: The type of <connected source object set> will be coerced first to <type of superset object set>, the type of the superset to which <connected target object set> maps, and then to <type of the connected target object set>.* ***Suggestion****: If desired, you may specify a more direct coercion.*

Predicate-Calculus Constraints

Predicate-calculus constraints derived from OSM model instances, and more particularly OSM participation and FD constraints, commonly impose certain restrictions on the relations in a relationship set. For a pair of relationship sets in a target-to-source mapping $\langle a,\ b\rangle$, the constraints can impose four possible implication relationships. Let $constr(r)$ denote the constraints of r, which are closed predicated-calculus formulas[†] derived from OSM model instances[‡] as explained in Section 2. Then, the relationships[§] are: (1) $constr(a) \Leftrightarrow constr(b)$, (2) $(constr(a) \Leftarrow constr(b)) \wedge (constr(a) \not\Rightarrow constr(b))$, (3) $(constr(a) \not\Leftarrow constr(b)) \wedge (constr(a) \Rightarrow constr(b))$, and (4) $(constr(a) \not\Leftarrow constr(b)) \wedge (constr(a) \not\Rightarrow constr(b))$.

Case 1 $constr(a) \Leftrightarrow constr(b)$. Case 1 here causes similar problems to Case 1 for object sets. Even when the types match, the structure matches, and the constraints match, we do not know that the semantics match. We discuss semantic matching in Section 3.4.

Case 2 $(constr(a) \Leftarrow constr(b)) \wedge (constr(a) \not\Rightarrow constr(b))$. For this case the constraints on the target relationship set are less restrictive than the constraints on the source relationship set. Assume, as an example, that the optional constraint on the relationship set between *City* and *City Map* in Figure 5 were mandatory. Then the constraints on this relationship set would be stronger than the constraints on the relationship set between *City* and *City Map* in Figure 2. In this case, the user who designed the target may be expecting more information than the source can provide—maps that include more than one city, for our example. We thus issue the following IDS.

IDS 6 ***Issue****: The constraints on the target relationship set, <target relationship set>, which are less restrictive than on the source relationship set, <source relationship set>, suggest that the target may be expecting more facts than the source can provide.* ***Default****: All source relationships will be transformed to target relationships.* ***Suggestion****: Since you may have been expecting additional facts, which cannot be provided by the source, you may wish to find another source that can provide these facts.*

Case 3, $(constr(a) \not\Leftarrow constr(b)) \wedge (constr(a) \Rightarrow constr(b))$. For this case the constraints on the target relationship set are more restrictive than the constraints on the source relationship set. There are two possible consequences: (1) we may not be able to load all facts without violating a target constraint, and (2) even if we load all facts we may have insufficient data to satisfy a target constraint. An interaction between these two consequences is also possible so that if we fail to load all facts because we violate a constraint, we may have insufficient data to satisfy some other constraint.

[†]As a technical aside, we observe that although the notation here succinctly captures the essence of what we want to say, the constraints, predicates, and values are for two different populated model instances. Thus, we must in some way convert these model instances and their values into a common symbol system before investigating implications [43]. It is therefore implied that we match and convert source predicate symbols to target predicate symbols and that we convert source value symbols to target value symbols by running them through the coercion routines we have established before we apply the implications.

[‡]While it is easy to state these implication with respect to an unrestricted set of predicate-calculus statements, we are really only interested in the predicate-calculus statements implied by OSM. Indeed, we are even more restrictive since we do not allow all the constraints available in OSM. (See [24, 22] for a complete description of constraints implied by OSM model instances.) In particular, we restrict ourselves to referential integrity constraints, subset constraints, intersection and union constraints, min-max participation constraints, and functional dependencies.

[§]Observe, by the way, that these four relationships have interesting correspondences to the four cases discussed for displayable types.

For the first consequence, consider as an example the functional correspondence between *Country* and *Kind of Money* in the target in Figure 2 versus the many-many correspondence between *Country* and *Currency* in the source in Figure 4. (Assume for our example here that there are no optional constraints on the relationship set between *Currency* and *Country* in Figure 4.) The user who designed the target may want just one currency for each country, but the source may provide several. For example, the source may have "German Marks", "Deutsche Mark", "DM", and "Euro" as currency names for Germany, but the target wants only one, e.g. the English name of the currency use as cash in souvenir shops ("German Marks"). In general, for this first consequence of Case 3, we cannot load all source facts, so the question becomes how to select the desired subset. We thus issue the following IDS.

> **IDS 7** ***Issue****: The constraints on the target relationship set, <target relationship set>, may not allow all facts from the source relationship set, <source relationship set>, to be loaded.* ***Default****: The system will load as many facts from the source as possible (in a convenient system-chosen order) discarding any fact that violates a target constraint.* ***Suggestion****: You may wish to specify a particular way to select the relationships from the source. Alternatively, you may wish to loosen the constraints for the <target relationship set> relationship set.*

For the second consequence, consider as an example the optional participation constraint on *Country* for the *Country Photo* relationship set in Figure 4. When matched with the corresponding mandatory participation constraint of *Travel Photo is for Country* in Figure 2, we see that the target constraint is stronger than the source constraint. The target is insisting that every country have a travel video, but the source may have none to supply. It may thus be impossible to satisfy the target constraint. In this case the user should either find a different source or loosen a target constraint. We thus issue the following IDS.

> **IDS 8** ***Issue****: The constraints on the target relationship set, <target relationship set>, may require additional facts that the source relationship set, <source relationship set>, cannot supply or that are not loaded if IDS 7 applies.* ***Default****: The system will recursively discard facts (in a convenient system-chosen order) from the populated target model instance, namely those that demand additional facts, until all constraints are satisfied.* ***Suggestion****: If this is not what you want, you may wish to find a different source that can supply the required facts. Alternatively, you may wish to alter the participation constraints for <target relationship set> in the target model instance to allow one or more mandatory participation constraints to be optional.*

To illustrate the point about recursively discarding facts, consider the following possible way the system might proceed in a particular case. Suppose the system loads all the target facts it can from the source in Figure 4 into the model instance in Figure 2. Suppose the system then discards all empty object and relationship sets, e.g. those such as *Country Map* and *Country has Location* for which the source has no facts, and also discards all constraints pertaining to these discarded object and relationship sets. Now, to satisfy the constraints for the populated object and relationship sets, suppose the system repeatedly checks constraints. If the system detects a violation, then it can also identify those facts that cause that violation. Accordingly it can discard those facts until no constraints are in violation. For example, if there is no travel photo for a country c, since country participation with a travel photo is mandatory, we discard c; discarding c, however, may violate referential integrity for the other relationship sets attached to *Country*, *Country has Population*, for example; and thus we discard the violating relationship set in *Country has Population*, which in turn causes a violation of the mandatory participation constraint on *Population*, so we discard the population value. We continue checking and propagating in this way until all constraints are satisfied[†]. If the system is clever, it discards as few facts as possible, but guaranteeing that it always discards as few as possible is likely to be exponential in runtime complexity.

[†]Discarding facts is safe as long as all constraints under consideration are downwards monotonic, i.e. each subset of a valid interpretation is again a valid interpretation. However, OSM allows for requirements that are not downwards monotonic (minimum participation, referential integrity, subset constraint, union constraint). Thus

To compound the problems further, we observe that we may simultaneously have both the first and the second consequence for Case 3. Consider, as an example, the relationship sets between *Country* and *Country Map* in Figures 2 and 5. The participation of *Country* is optional in Figure 5 but mandatory in Figure 2, and the relationship set in Figure 2 is one-one, rather than merely functional in one direction. In this case, we issue both IDS 7 and IDS 8.

Case 4, $(constr(a) \not\Leftarrow constr(b)) \wedge (constr(a) \not\Rightarrow constr(b))$. For this case, both target and source have at least one more restrictive constraint and at least one less restrictive constraint. The relationship sets between *City* and *City Map* in Figures 2 and 5 are an example. The optional constraint prevents the implication from the relationship set in Figure 5 to the relationship set in Figure 2, and the FD prevents the implication in the other direction.

Since Case 4 is a combination of Cases 2 and 3, we use the same IDS's. We issue IDS 6 and either one or both of IDS 7 and IDS 8. We can use the same reasoning as just explained to decide whether just one or the other or both of IDS 7 and IDS 8 should be issued. For the *City-City_Map* example we issue IDS 6 (to warn the user that there are no maps with more than one city) and IDS 8 (to warn the user that some city may not be loaded because it has no map).

3.3. ISA Constraint Compatibility

ISA constraints in OSM require specialization object sets to be subsets of generalization object sets and may also require that a generalization object set be a union of the specialization object sets, or that the specialization object sets be disjoint, or both. If the types of all the object sets in an ISA hierarchy are identical, we can easily satisfy these conditions. If the types differ, however, we must be more precise about exactly what these conditions mean. Suppose object set a is a specialization of object set b, but that $type(a) \neq type(b)$. Since a is a specialization of b, OSM requires that we must semantically have $a \subseteq b$, that is, each object in a must be an object in b. When the types are different, this merely means that the same object is represented differently in the two object sets. Hence, to check the constraint there must be a way to match objects in the two sets. OSM satisfies this by requiring "semantically correct" injective type conversions[†] both from a specialization to a generalization and from a generalization, appropriately restricted, to a specialization. When we check ISA-hierarchy constraints such as subset, union, or mutual-exclusion, we are always checking with respect to the semantics—type conversions are always implicit and used when needed.

With this in mind, we state the following requirements about ISA hierarchies for our target-to-source mapping function f.

> **Req. 7** *Let f be a target-to-source mapping. Then the agreed-on (possibly trivial) conversion functions for the object sets in the range of f must be "compatible" with the ISA constraints in the source. That is, if a' and b' are in the range of f and if the ISA constraints in the source imply $a' \subseteq b'$, then:*
>
> 1. *The conversion for the specialization, $c_{a'}$, and the conversion for the generalization, $c_{b'}$, coincide on common values.*
> 2. *The conversion for the generalization, $c_{b'}$, never equates different values appearing in different specializations.*

> **Req. 8** *Let f be a target-to-source mapping, and let $\langle a, a' \rangle$ and $\langle b, b' \rangle$ be object-set ordered pairs in f. If the ISA constraints in the target imply $a \subseteq b$, then the ISA constraints in the source must imply $a' \subseteq b'$.*

discarding facts may result in a violation of a constraint that had already been satisfied. In order to adjust, we can proceed by recursively discarding more facts with the consequence, unfortunately, of removing otherwise useful data. We observe, however, that the recursion is always guaranteed to terminate with a valid interpretation, which might be the empty population in the worst case. This claim follows from the implicational form of the first-order logic formalization of all OSM constraints under consideration.

[†]By "semantically correct" type conversions we simply mean that the conversions correspond to the user-intended semantics (e.g. the integers are a subset of the reals with natural coercions in both directions that are injective when appropriately restricted). If default conversions are not available or not semantically appropriate for a particular situation, a user must provide the needed type conversions.

Req. 9 *Let f be a target-to-source mapping, and let $\langle a,\ a' \rangle$ and $\langle b,\ b' \rangle$ be object-set ordered pairs in f. If the constraints of the target imply $\forall x(x \in a \Rightarrow x \notin b)$ and $\forall x(x \in b \Rightarrow x \notin a)$, then the constraints in the source must imply $\forall x(x \in a' \Rightarrow x \notin b')$ and $\forall x(x \in b' \Rightarrow x \notin a')$.*

Req. 10 *Let f be a target-to-source mapping, and let $\langle a,\ a' \rangle$ and $\langle b_1,\ b'_1 \rangle$, ..., $\langle b_n,\ b'_n \rangle$ be ordered pairs in f. If a is the generalization of a union-constrained ISA in the target such that the ISA constraints in the target imply $a = b_1 \cup ... \cup b_n$, then the ISA constraints in the source imply $a' = b'_1 \cup ... \cup b'_n$.*

Requirements 7 through 10 are intended to ensure that a populated ISA hierarchy in the target model will satisfy the OSM constraints on ISA hierarchies. Requirement 8 together with Requirement 7(1) and the OSM requirements for types in ISA hierarchies assure us that all subset constraints in the target hold. Similarly, with the OSM ISA-hierarchy constraints implicitly in place, Requirement 9 together with Requirement 7 assure us that any mutual-exclusion constraints hold, and Requirement 10 together with Requirement 7(1) assure us that any union constraints hold.

3.4. Context Keywords and Data Values

Using context keywords and data values, we now address the problem of semantics. We attempt to eliminate from consideration nonsensical matches, such as mapping *Country* in Figure 2 to *Prominent Person* in Figure 4, *Population* to *Description*, and *Country has Population* to *Description ProminentPerson*. In these examples, all types and constraints are fully compatible, but we can easily see that the semantics are completely different.

We address semantic issues in two ways. We look for the presence of expected context keywords, and we look for the presence of expected objects (values) in object sets and expected relationships (tuples of values) in relationship sets. We declare what we expect, as explained in Section 2, in the textual representation of the target model instance (see Figure 3). We check for the presence of keywords in the names of source object and relationship sets and in any type names or comments in the source pertaining to object sets or relationship sets, and we check for the presence of values and tuples of values in the populated model instance derived from the source database.

For our supposed match of prominent persons and their descriptions with countries and their populations, there is very little chance of any keyword or value match. Neither *Population* nor *population*, the default keywords, match *Description*, nor would they be found in any reasonable type declaration or comment about *Description*. Moreover, there is no chance that the *Population* regular expressions would match any reasonable *Description* string. Similar statements can be made about the two relationship sets involved and, for the most part, also about the other object sets. It is reasonable to imagine, however, that a comment about *Prominent Person* may include "country". A comment like "– contains the name of a well-known person in a country" is certainly reasonable for *Prominent Person*.

The mere presence of a single keyword or value match would not usually provide enough evidence to convince us that the there is a semantic match, but many matches of both keywords and values for a relationship set and its connected object sets would provide evidence that the semantics match. For example, when we map *Country* in Figure 2 to *Country* in Figure 4, map *Population* to *Population*, and map *Country has Population* to *Country Population*, we find expected keywords. Further, when values for both *Country* and *Population* are checked in Figure 7, we find expected tuples in *Country*, *Population*, and *Country Population*.

If we have type mismatches, observe that we should first coerce source values to target types before applying regular expressions. If the *Population* values in Figure 4, for example, were of type *Integer*, we would first need to convert them to *String*. Observe for this example, that the typical default conversion of just extracting and concatenating the digits, would not work for the simple set of regular expressions in Figure 3. Here, our conversion from *Integer* to *String* would need to have commas separating the thousands and millions. Alternatively, the regular expressions could

be significantly improved—for example, to allow optional commas, to allow for periods or spaces in place of commas, to allow for values rounded to millions or to millions with a decimal point.

Sorting out what are sometimes subtle differences in meanings, as opposed to sorting out gross differences as in our country-population/prominent-person example, can be challenging, even for users. We do not expect a system to be able to resolve subtleties without user input. The mapping generator can, however, discard gross differences as possibilities and isolate relevant information for more subtle cases.

One interesting case with likely semantic subtleties is the possibility that multiple relationship sets can span the same object sets. In Figure 6, for example, both *Airport is located in City* and *Airport serves City* connect the object sets *Airport* and *City*. It is not likely that either context keywords or values would be sufficient to positively declare that the relationship set *Airport is for City* in Figure 2 matches one but not the other city-airport relationship set in Figure 6. When a target relationship set may match with any one of several source relationship sets that span the same object sets and when keyword matches do not uniquely select one of the source relationship sets, we issue the following IDS.

> **IDS 9** ***Issue****: The target relationship set <target relationship set> can map to only one of the multiple relationship sets in the source that span the object sets, <list of object sets>.* ***Default****: The system will choose the one with the highest confidence value or will choose arbitrarily among two or more with the same highest confidence value.* ***Suggestion****: You may instead wish to specify which one to choose.*

The semantic checks we are proposing do not provide absolute assurances, but they do provide evidence for or against mapping pairs. Used in conjunction with techniques proposed in [52, 14, 29, 50, 46, 36, 13, 49, 48, 6], however, they can increase or diminish our confidence in proposed results. They also help us deal with the potential geometric explosion of possible matches based on syntax alone. As we generate target-to-source mappings, we avoid the geometric explosion by relying on the presence of some semantic matches, and we record the evidence obtained by these matches to support or refute possible matching pairs. The user has the final say, of course, but the mapping generator does its part by removing the tedium of checking all the possibilities, by isolating and questioning identifiable semantic subtleties, and by finding and pointing out the evidence to support its proposed mappings.

4. MATCHING RULES—DERIVED OBJECT AND RELATIONSHIP SETS

Although a source may not have object and relationship sets that directly correspond to a target's declared object and relationship sets, target facts may nevertheless be derivable from source facts. We can, for example, form a derived relationship set by joining relationship sets along a path in a model instance, or form a specialization object set by selecting only those objects in an object set that satisfy some criteria. In general, we can specify these object- and relationship-set derivations as queries. Once specified, we can consider these derived object and relationship sets to be part of the source. We can then generate target-to-source mapping pairs and use the inverses of these mapping pairs to populate object and relationship sets in a target.

Since we can specify derived object and relationship sets by queries, and since the number of queries over a model instance is typically unbounded, we are selective in the kinds of queries our target-to-source generator recognizes and supports. We do not, for example, support query transformations involving aggregate operators (e.g. derive the average population of countries for each continent) or query transformations over image content (e.g. derive the subset of topological maps that have more green than any other color), or a host of other possible query transformations. The categories of query transformations we do consider are: (1) generalizations and specializations of object sets, (2) string decompositions and compositions, (3) derivations of nondisplayable object sets for matching displayable/nondisplayable object sets, and (4) path queries including queries over degenerate paths, consisting of only one edge. For each of these transformations we must (1) recognize that we need the transformation, (2) formulate the transformation query, and (3) derive the constraints for the view generated by the transformation query.

4.1. Generalization/Specialization Derivations

For a target object set, a particular source may have (1) none of the set of desired values, (2) a proper subset of the desired values, (3) exactly the set of desired values, or (4) a proper superset of the desired values. We need not consider the first case, except to say that there should be no source match for the target object set. For the second case, we can either reject any potential target-to-source matches for the object set (if the user is not satisfied with only a subset) or we can treat the second case as if it were the third case. The fourth case may have a resolution in terms of roles, in which case we resolve it as we explain beginning in the next paragraph. Otherwise, either the extra values in the proper superset may be acceptable to a user, in which case, we can treat the fourth case as if it were the third case, or if not, the the values need to be filtered with a user-supplied selection criterion. Since our approach to recognition in this paper is based on simple context-keyword and sample-value matches, there is not a good way to recognize the need for a selection criteria. Semantic hypernymy techniques, such as those discussed in [6] and [13], can help with this recognition problem, but these are beyond the scope of this paper. We therefore do not further discuss cases that required user-supplied selection criteria.

Roles

Suppose a target has a role r for an object set s in relationship set q and q has a potential match[†] with a source relationship set q'. Suppose also that q' has an object set s' that has a potential match with with s or is a specialization of a source object set s'' that has a potential match with s, and further that s' has no role for q'. Then we can generate a role for the source and match it with r. We can give the new source role the name r, the same name as the target role name, adding a numeric suffix if needed to make the name unique in the source. Assuming its name is r, we can then populate it with the values computed by the expression $\rho_{s' \leftarrow r}\pi_{s'}q'$.

As an example, consider the role *Video With City Scene* in the target in Figure 2. The target relationship set *City appears in Video With City Scene* has a potential match with the source relationship set *City Clip*, and the target object set *Travel Video* has a potential match with the source object set *Clip*. We thus add *Video With City Scene* as a role on *Clip* in the relationship set *City Clip* in Figure 4 and populate it by the expression $\rho_{Clip \leftarrow VideoWithCityScene}\ \pi_{Clip}City\_Clip$.

Missing Generalizations

When a source contains the desired set of objects or a user-acceptable subset or superset, there are two cases of interest. (1) The source contains the desired set of objects in a single object set. (2) The source contains the desired set of objects in a set of object sets. We can resolve the first case of these two cases by a direct match, as discussed in Section 3.3. For the second case, there is a missing source generalization. We discuss this second case here.

The basic idea is to create a new generalization in the source to which the generalization in the target can map. This new generalization has the same name as the target generalization to which it corresponds (with a numeric suffix if needed to make the name unique in the source). We then make the new generalization a union generalization of the set of object sets that holds the objects of interest. If we can also prove from source constraints that the specializations that form the union generalization are mutually exclusive, we can strengthen the union constraint to be a partition constraint. In addition to creating the union and providing the proper constraints, we may also need to coerce values so that they have the proper types for the created source ISA hierarchy.

As an example, the object set *Map* in Figure 2 does not correspond to any object set in the source in Figure 5, but its target specializations, *Country Map* and *City Map* in Figure 2 do correspond respectively to the source object sets with the same names in Figure 5. Hence, we can generate a virtual source object set for Figure 5 that is the union generalization of *Country Map* and *City Map*.

[†]When we say "potential match," we mean that we are considering an ordered pair in the target-to-source mapping we are building, not that the ordered pair is in the target-to-source mapping. As we build, the set of matches we consider need not satisfy requirements for target-to-source mappings such as being injective. It is only at the end of the process that we restrict the set of potentially matching ordered pairs to a set that satisfies the requirements.

Length	Nr Hours	Nr Minutes
1 hr 15 min	1 hr	15 min
15 min	⊥	15 min
1 hr 30 min	1 hr	30 min
2 hrs	2 hrs	⊥

Fig. 9: Generated *Length_NrHours_NrMinutes* relationship set.

4.2. Composite Strings

Conceptual modelers do not always choose to represent values at the same level of atomicity. In our sample application, for example, the target (Figure 2) has the length of a travel video decomposed into the number of hours and the number of minutes, whereas a source (Figure 4) models it only as length. There are two cases to consider: (1) the composite is in the source and (2) the composite is in the target.

Composite in Source.

We can consider a source object set to be composite if regular expressions for two or more target object-set values decompose source values. We can obtain further supporting evidence for the decomposition if context keywords for the source object set are present in all the target object sets whose regular expressions decompose source values. The first and third values in the source object set *Length* in Figure 7, for example, can be decomposed by the regular expressions in *Nr Hours* and *Nr Minutes* in Figure 3, Lines 58–67. Further, the context keyword *Length* in these target object sets matches the source object set name.

If we have regular expressions that partition each of the string values in the source, the system may proceed without user intervention. Otherwise, we issue the following IDS.

> **IDS 10** ***Issue****: The regular expressions provided for the target object sets, <list of involved target object sets>, do not appear to partition the source values in <composite object set>.* ***Default****: The system will extract the subcomponents it recognizes, discard any remaining string components, and fill in null strings for missing substrings.* ***Suggestion****: You may wish to adjust the regular expressions.*

Figure 9 shows the values generated for our sample application. The ⊥ denotes a null string.

Figure 10 shows the resulting transformation of the source composite *Length* and how it connects to the model instance in Figure 4. We add an object set to the source for each of the n participating target object sets. We use the name of the target object sets for the source (with a numeric suffix if necessary to make the name unique in the source). For our example, we add *Nr Hours* and *Nr Minutes* to the source. We next add an $n+1$-ary relationship set connecting the source object set to be decomposed and the n new source object sets. The participation constraints on this relationship set are 1 for the source object set to be decomposed and 1:* for the new source object sets[†].

We can populate this view with the query $\epsilon_{Length,\ NrHours,\ NrMinutes}Length$, whose result is the relation in Figure 9. Here, we introduce the value-extraction operator ϵ. The ϵ operator has the form $\epsilon_{A_1,\ldots,A_n}r$, where r is a single-attribute relation, and A_1 through A_n are a set of attribute names. The result is a relation whose scheme is $A_1 \ldots A_n$ and whose tuples are formed by extracting substrings from each value v of r. For each i ($1 \leq i \leq n$), if A_i names the attribute of r, the A_i component is v, and if A_i names an attribute with an associated routine that extracts a substring a_i from v, the A_i component of the tuple is a_i, otherwise the A_i component is ⊥.

Composite in Target.

[†]It is reasonable to consider adding an FD whose composite left-hand-side consists of all the new object sets and whose right-hand-side is the original source object set, but only if we can guarantee that the regular expressions are such that they never discard any string components. Since this may be difficult (potentially impossible) to prove, we do not add the FD.

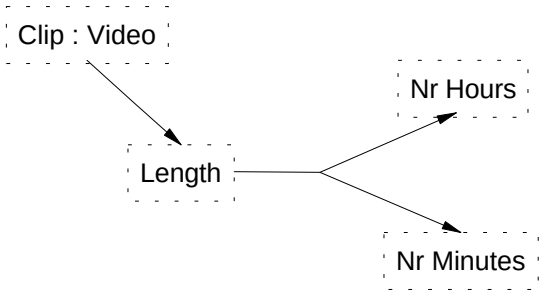

Fig. 10: Generated *Nr Hours* and *Nr Minutes* for *Length* for the source in Figure 4.

We can consider a target object set to be composite[‡] if the regular expression for its value recognizes a concatenation of values from two or more source object sets. We can obtain further supporting evidence for the composition if keywords declared in the composite target object set appear in some or all of the source object sets to be concatenated. As an example, if *Location* in Figure 2 were displayable and *Latitude* and *Longitude* were not present, then *Location* would be a composite of *Latitude* and *Longitude* in Figure 5.

Once a composite target object set is recognized and its constituent source object sets have been identified, the system faces the problem of how to construct the composite values. A priori, it does not know which of the values in the object sets to concatenate together, in which order to concatenate the values, and whether the concatenated values require a separator. We therefore issue the following IDS.

> **IDS 11** ***Issue****: Each value in <target object set> is to be constructed by concatenating one value from each of the following object sets: <list of source object sets>. The system can only guess which values should be concatenated together, in which order they should be concatenated, and what characters (if any) should separate the values.* ***Default****: As its best guess, the system finds any minimal path connecting all the identified source object sets (or groups of disjoint minimal paths connecting all the identified source object sets in case the object sets are in disconnected components), joins over the (possibly degenerate) path, and projects on the object sets to form a relation. Then, using a convenient order the system concatenates values in each tuple in the generated relation with a space between each value.* ***Suggestion****: If there are multiple minimal paths or if the minimal path does not properly join values to be concatenated, specify the path. Unless it does not matter, specify the order for concatenation. If a string other than a blank character should separate the concatenated values, specify it.*

For our *Location* example, we would want to join over the relationship sets *Country Latitude* and *Country Longitude* and project on *Latitude* and *Longitude* to obtain the pairs to concatenate, and we would want to order the pairs with *Longitude* first, with a space separating the two values. Since the path length between *Longitude* and *Latitude* is just as short through *City* and since the system might concatenate the pairs with *Latitude* first, we should specify our choices. The separating space provided by the default is acceptable.

Given the list of source object sets to be concatenated, a join path among them, the order for concatenation, and a separator character or string, we can generate the source view we need. For our example, Figure 11 shows the generated view and how it connects to the source model instance in Figure 5. In general, we (1) create a new object set whose name is the name of target object set (with a numeric suffix attached to make the name unique within the source, if necessary), (2) create a new $n + 1$-ary relationship set connecting the n source object sets and the new object set, (3) give the participation constraint 1 to the connection between the new relationship set and

[‡]Note that since the target is under the control of the user, the target composite object sets can always be decomposed. We are assuming, however, that the target model instance is fixed and that the user does not wish to decompose object sets that are composites of source object sets.

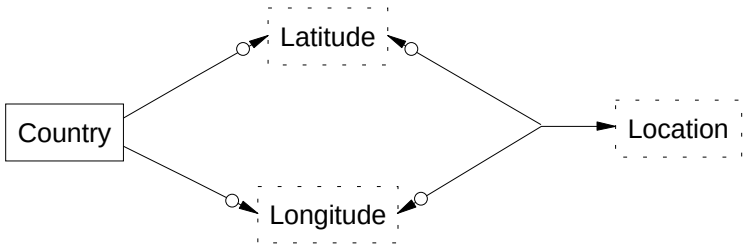

Fig. 11: Generated *Location* view for source in Figure 5.

the new object set, (4) give the participation constraint 0..* to all other connections (or tighter constraints if we can prove that they hold), and (5) add a functional constraint to the relationship set whose left-hand-side consists of all the source object sets participating in the concatenation and whose right-hand-side is the newly derived object set.

We can populate this view in Figure 11 with the query $\gamma_{Location\ :=\ Longitude+" "+Latitude}\ \pi_{Longitude\ Latitude}\ (Country\_Longitude \bowtie Country\_Latitude)$. Here, we introduce the concatenation operator γ. The γ operator has the form $\gamma_{B\ :=\ A_1+\ldots+A_n} r$ where B is a new attribute not among the attributes of the relation r and each A_i, $1 \leq i \leq n$, is either an attribute of r or is a string. The result of the γ operator is r with an additional attribute B, where each B value on row k is a concatenation of the given strings and the specified attribute values from row k.

4.3. Displayable/Nondisplayable Object-Set Matches

Earlier, in Section 3.1, we discussed object-set pairs in target-to-source mapping in which object sets in the pair were either both displayable or both were nondisplayable. Here, we consider the displayable/nondisplayable mismatches. There are two cases to consider: (1) the target has a displayable object set that corresponds to a nondisplayable source object set and (2) the target has a nondisplayable object set that either corresponds to a displayable source object set or corresponds to a source relationship set or derivable source relationship set.

Nondisplayable Object Set in Source

Based on context keywords, our sample application has several target displayable object sets that potentially match with nondisplayable object sets in the sources. The target in Figure 2 includes the displayable object set *Airport*, which matches the nondisplayable object set *Airport* in the source in Figure 6, and the displayable object sets *Country* and *City* match nondisplayable object sets with the same names in Figure 5.

In these cases, we look for a displayable source object set whose values are in a one-to-one correspondence with the OID's of the nondisplayable source object set under consideration. We call displayable object sets whose values are in a one-to-one correspondence with the OID's of a nondisplayable object set *key object sets*. *Airport Code*, for example, is a key for *Airport* in Figure 6, and *Country Name* is a key for *Country* in Figure 5. Note, however, that *City* in Figure 5 has no key. For these possibilities there are three cases to consider.

Case 1 – one key. If a target displayable object set s has a potential match with a nondisplayable object set n and the constraints of the source guarantee that a displayable object set d is a key for n, then, if there is only one such displayable object set d, we can discard the potential match between s and n and add, if not already present, the potential match between s and d. The potential mapping from the displayable object set *Airport* in Figure 2 to the nondisplayable object set *Airport* in Figure 6 can be redirected to the displayable object set *Airport Code*.

Case 2 – no key. If a target displayable object set s has a potential match with a nondisplayable object set n but there is no key for n, we can do nothing. In this case, we reject the potential match of s and n—it makes no sense to load the displayable target object set with (arbitrary) OID values.

Case 3 – multiple keys. If a target displayable object set s has more than one object-set key or composite object-set key, the system may need the user's help. We thus issue the following IDS

statement.

> **IDS 12** ***Issue****: The target displayable object set <target displayable object set> appears to match the source nondisplayable object set <source nondisplayable object set>, but there are several possible keys, <list of keys>.* ***Default****: The system will select one, giving preference to keys that have potential matches with <target displayable object set>, but otherwise will select one arbitrarily.* ***Suggestion****: If this is not satisfactory, designate a match between the target displayable object set and one of the keys.*

Nondisplayable Object Set in Target

Suppose we have determined (e.g. through context keyword matching) that a nondisplayable target object set corresponds to a displayable source object set. Given this correspondence, we can always generate a nondisplayable source object set, populate it with OID's that are unique within the source, and place the OID's in a one-to-one correspondence with the values in the matching displayable source object set. We can then match the nondisplayable target object set with this generated nondisplayable source object set instead of with the displayable source object set. We will have then reduced the nondisplayable/displayable mismatch to a nondisplayable/nondisplayable match and we can proceed as explained in Section 3.1.

As an example consider the nondisplayable object set *Location* in the target in Figure 2 and assume that the source is as in Figure 11, which has a displayable object set *Location*. To resolve this nondisplayable/displayable conflict, we generate a new nondisplayable object set for the source in Figure 11. We give the new object set the name of the target object set to which we want to establish a correspondence (appended with a numeric suffix if necessary to make it unique). For our example, we need the numeric suffix and would generate $Location_2$ to make its name different from the displayable object set *Location*, which already exists in Figure 11. We then generate a new relationship set connecting the two source object sets and supply the relationship set with the participation constraint 1 for both connections to force the values in the two object sets to be in a one-to-one correspondence. Finally, we populate the generated nondisplayable object set with as many source-unique OID's as values in the displayable source object set and populate the generated relationship set to satisfy the one-to-one correspondence.

When a nondisplayable target object set corresponds to a source relationship set or derivable source relationship set, we generate a nondisplayable source object set that represents the relationship set. In particular, we populate a new nondisplayable source object set with unique OID's in equal number to the number of relationships in the relationship set. We then generate new source binary relationship sets that connect the new nondisplayable object set to the object sets of the relationship set. These new binary relationship sets all have a participation constraint of 1 on the side of the new nondisplayable object set. Participation constraints for the connecting object sets are derivable. If for a participating object set an FD is derivable using standard FD theory, the maximum participation constraint is 1 and is otherwise *. If a value in a participating object can be dangling in the join (degenerate join for only one relationship set), the minimum participation constraint is 0 and is otherwise 1. We can determine if the object can be dangling in the join, by considering the optional/non-optional participation constraints along the join path. We must also add an equality constraint declaring that the set of tuples in the original relationship set is identical to the set of tuples in the join of all the new binary relationship sets connected to the new nondisplayable object set with a projection on all the connected object sets (i.e. with the new object set projected out).

The system recognizes that a nondisplayable target object set corresponds to a source relationship set when the following conditions hold. (1) The nondisplayable target object set has no corresponding source object set. (2) Each target object set related to the nondisplayable target object set has a potential match with a source object set. (3) There is a path in the source connecting these source object sets. If more than one path is possible, we issue the following IDS.

> **IDS 13** ***Issue****: The nondisplayable target object set <target object set> can map to at most one of the following possible paths in the source: <list of paths>.* ***Default****: The default action is to choose arbitrarily among the shortest paths.* ***Suggestion****: You may wish instead to specify which one it should choose, or reject them all.*

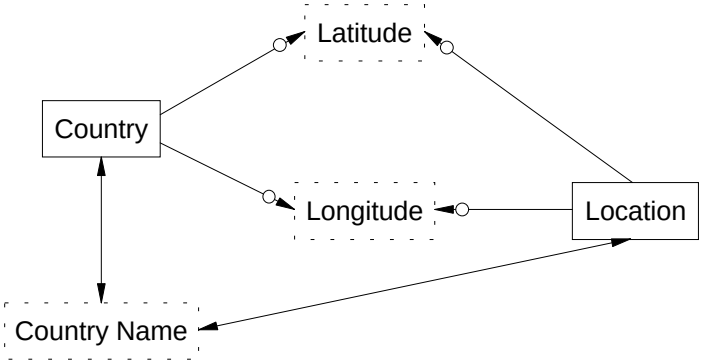

Fig. 12: Generated *Location* derivation for source in Figure 5.

Location in our target model instance in Figure 2 is nondisplayable and does not have a potential match with any object set in the source in Figure 5. However, the related target object sets *Country*, *Latitude*, and *Longitude* all have potential matches with source object sets. Let us assume that *Country* matches with *Country Name*, not *Country*, as a result of the displayable/nondisplayable match resolution discussed earlier. Thus, the default path, *CountryName Country*, *Country Latitude*, *Country Longitude* is the matching path we want. Figure 12 shows the transformation that produces *Location* as a source object set and its connection to the object sets in Figure 5. The equality expression, which must be added, is

$$\begin{aligned}
&\forall y \forall z \forall w (\exists x (Country_CountryName(x,\ y) \\
&\qquad \wedge\ Country_Latitude(x,\ z)\ \wedge\ Country_Longitude(x,\ w)) \\
&\Leftrightarrow \\
&\quad \exists v (CountryName_Location(y,\ v) \\
&\qquad \wedge\ Latitude_Location(z,\ v)\ \wedge\ Location_Longitude(v,\ w))).
\end{aligned}$$

4.4. Source Paths

In the same sense that views may differ with respect to atomicity of object-set values, views may also differ with respect to atomicity of relationships. One view may have a direct relationship between values in two object sets, while another view may model the same relationship indirectly with intermediate values. A common example is a grandparent relationship, which may directly relate a grandchild and a grandparent in one view but may have a parent as an intermediate value in another view. In general, we may have a single relationship set in one view that semantically corresponds to a path in another view. We consider only the case in which the path is in the source. (The other case in which the path is in the target fails for our application because the target has no data and thus cannot supply the intermediate values.)

We recognize a possible correspondence between a target relationship set and a source path when the following conditions hold. A target relationship set r has no potential match to a source relationship set, but all of r's object sets have potential matches (either to existing or generated object sets). The matched source object sets must also be connected by one or more relationship sets. We can be particularly confident about a target relationship set matching a source path if the regular expressions for target tuples match a relation formed by joining over one (or more) of these paths and projecting on the matched source object sets.

Our sample application has several examples. The source relationship set *City is in Country* in Figure 2 does not directly correspond to any relationship set in any of the sources (Figures 4, 5, and 6). Consider, however, the path consisting of the relationship sets *Country Photo* and *City CityPhoto* in Figure 4. *Country* and *City* in the target (Figure 2) potentially match with *Country* and *City* in the source (Figure 4), and a join/project over the source path would likely yield a match with the *City is in Country* regular expressions provided in the target ontology (Figure 3),

e.g. would likely match `< Berlin , Germany >`, or `< New\s*York , (USA | United\s*States) >`, or `< Paris , France >`. To illustrate matching with virtual paths and object sets, consider the target path *NrHours TravelVideo* in Figure 2 which matches the virtual source path *Clip Length*, *Length NrHours NrMinutes* in Figure 10. To illustrate a degenerate path (a path with only one relationship set), consider the source relationship set *USExchangeRate is for KindOfMoney* in Figure 2 which potentially matches the (degenerate) path *Currency ForeignCurrency Rate* in Figure 4.

Sometimes a target relationship set may correspond to several paths in a source. The target path *City is in Country* in Figure 2, for example, corresponds not only to *Country Photo*, *City CityPhoto* in Figure 4 as just mentioned, but also to *Country Clip*, *City Clip*. Which path, if any, corresponds semantically with the target relationship set must be determined. When there is only one choice, the target-to-source mapping generator records it as a possibility, but when there are multiple possible paths, the system issues the following IDS.

> **IDS 14** ***Issue****: The target relationship set <target relationship set> can map to at most one of the following possible paths in the source: <list of paths>.* ***Default****: The default action is to choose arbitrarily among the shortest paths.* ***Suggestion****: You should instead specify which one it should choose, or reject them all.*

As a default the system produces a virtual source relationship set by joining along a source path and projecting on the matched source object sets. Sometimes we may need to use a different query to generate the virtual relationship set. We therefore issue the following IDS.

> **IDS 15** ***Issue****: A derived relationship set in a source can be produced by any query.* ***Default****: The default query joins over the path and projects on the object sets for the relationship set. It also automatically provides constraints for this derived relationship set.* ***Suggestion****: If you want a different query, you must specify it. Further, if you provide a query and want the constraints to be tighter than 0..* participation constraints, you must also specify the constraints.*

Consider, for example, the target relationship set *USExchangeRate is for KindOfMoney* in Figure 2, which matches with the source relationship set *Currency ForeignCurrency Rate* in Figure 4. Here, we must first respond to the IDS 14 and choose to match *Kind Of Money* with *Currency* rather than with *ForeignCurrency*. But simply projecting on *Currency* and *Rate* does not give us the result we want. Since we need the exchange rate to be only for US dollars, we should provide the query $\pi_{Currency\ Rate}\ \sigma_{ForeignCurrency\ =\ "US\$"}\ (Currency\_ForeignCurrency\_Real)$ as a replacement for the default query.

The system can compute constraints for the generated virtual source relationship set for the default query, but for arbitrarily specified user queries, constraint computation may not be possible. We therefore ask the user to provide constraints if they should be tighter than 0..* participation constraints for every connection. For the default, we can use the functional constraints on the path to obtain a reduced set of FD's among the source object sets according to standard functional-dependency theory and specify these either as one-maximum participation constraints when the left-hand-side of an FD is a single object set or otherwise as FD constraints on the relationship set with *-maximum participation constraints. To specify the minimum participation constraints, we consider the mandatory/optional specifications of the path in the source. If the mandatory/optional constraints along the path demand the participation of the objects in a matched object set, the minimum is 1; otherwise the minimum is 0.

5. TARGET-TO-SOURCE MAPPING GENERATOR

We present our proposed target-to-source mapping generator as an algorithm that fills in a table as it executes. The filled-in table includes the justification for the generated target-to-source mapping, as well as information about alternative mapping pairs that do not become part of the

selected mapping function. The table also includes information about which IDS statements are issued as well as the user's response to these statements.

Algorithm *Generate Target-to-Source Mapping* fills a table with four sections: (1) *Recognition Criteria*, which contains information about how the algorithm recognizes a proposed matching pair, (2) *Special Considerations*, which contains information about what issues arise for a proposed matching pair and how these issues are resolved, (3) *Confidence*, which contains a confidence value for a proposed matching pair, and (4) *Selected Pair*, which is marked only if the proposed matching pair becomes part of the resulting generated functional mapping. For rows in the first two sections without IDS's, the generator fills in "x" for "yes" and leaves a blank for "no", and for rows with IDS's it fills in "d" for "yes, default", "u" for "yes, user-specified", and "r" for "no, user-rejected".

Algorithm 1 (Generate Target-to-Source Mapping)

Input:
  Target: an OSM model instance (textual representation)
  Source: a populated OSM model instance
Available Resources
  Type Hierarchy and Default Coercion Routines
  Unit Conversion Routines
  Confidence-Value Criteria
Output:
  A selected ‘‘best’’ target-to-source mapping
Procedure
  record direct-match object-set information
  record direct-match relationship-set information
  generate derived object sets and record information as follows:
    generate missing source roles
    generate missing source generalizations
    resolve composite source object sets
    resolve composite target object sets
    resolve target-displayable/source-nondisplayable object sets
    resolve target-nondisplayable/source-displayable object sets
  generate derived relationship sets and record information
  for each column in the table
    if the confidence value is not -1, ‘‘considered but rejected’’
    then set the confidence value as specified in the
      confidence-value criteria
  select the ‘‘best’’ functional mapping that satisfies
    Requirements 1 through 10

Tables 1 through 6 show the filled-in tables for our sample application. The generator algorithm fills in one table for each source, Figure 4 about countries, Figure 5 about maps, and Figure 6 about airports. The tables for both the countries source and the maps source, however, are too large to be shown as a single table. We break the countries source table into three tables (Tables 1, 2, and 3) and further reduce these three by omitting rows with no entries in the *Special Considerations* section of the table, and we break the maps source table into two tables and also further reduce the first by omitting most of the rows in the *Special Considerations* section that have no entries.

Algorithm *Generate Target-to-Source Mapping* takes as its input (1) a target OSM model instance like the one shown in Figure 3 and (2) a source populated OSM model instance like the (partial) one shown in Figure 7. It also takes three auxiliary inputs: (1) a type hierarchy like the one shown in Figure 8 as well as associated default coercion routines that, as a minimum, provide coercions in both directions for every ISA in the hierarchy, (2) unit-conversion routines, and (3) a user-specified criteria for confidence values, in which the confidence values are nonnegative numbers. For illustration here, the confidence criteria we select is computed for each proposed

	Matched Pairs															
Criteria	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
Recognition Criteria																
keyword	x	x					x	x	x	x	x	x	x	x	x	x
value	x	x	x	x	x	x				x		x				
structure	x	x	x	x	x	x	x	x	x	x			x	x		
Special Considerations																
units (IDS 1)			x			x										
$=$	x	x		x	x	x	x		x	x		x	x	x	x	x
$\supset$ (IDS 2)								d								
$\subset$ (IDS 3)			d													
$\not\sim$ (IDS 4)											r					
Confidence	8	8	5	6	6	6	6	5	6	8	-1	6	6	6	-1	-1
Selected Pair (Req. 1–10)	x	x		x		x	x	x	x	x	x	x	x			

01. ⟨*Country*, *Country*⟩
02. ⟨*Population*, *Population*⟩
03. ⟨*Size*, *Population*⟩
04. ⟨*Kind Of Money*, *Currency*⟩
05. ⟨*Kind Of Money*, *Foreign Currency*⟩
06. ⟨*US Exchange Rate*, *Rate*⟩
07. ⟨*Travel Photo*, *Photo*⟩
08. ⟨*Caption*, *Caption*⟩
09. ⟨*City Photo*, *City Photo*⟩
10. ⟨*City*, *City*⟩
11. ⟨*City*, *City Photo*⟩
12. ⟨*Airport*, *International Airport*⟩
13. ⟨*Travel Video*, *Clip*⟩
14. ⟨*Topic*, *Subject*⟩
15. ⟨*Nr Hours*, *Length*⟩
16. ⟨*Nr Minutes*, *Length*⟩

Table 1: Direct object-set matches for mapping Figure 2 to Figure 4

	Matched Pairs									
Criteria	01	02	03	04	05	06	07	08	09	10
Recognition Criteria										
keyword										
value	x									
structure	x	x	x	x	x	x	x	x	x	x
Special Considerations										
$\Leftrightarrow$	x	x			x			x	x	x
$\Leftarrow \not\Rightarrow$ (IDS 6)						d				
$\not\Leftarrow \Rightarrow$ (IDS 7)			d				d			
$\not\Leftarrow \Rightarrow$ (IDS 8)			d	d			d			
Confidence	6	4	4	3	4	3	4	4	4	4
Selected Pair (Req. 1–10)	x		x	x	x	x	x	x		

01. ⟨*Country has Population*, *Country Population*⟩
02. ⟨*Country has Size*, *Country Population*⟩
03. ⟨*Country has Kind Of Money*, *Country Currency*⟩
04. ⟨*Travel Photo is for Country*, *Country Photo*⟩
05. ⟨*Caption describes Travel Photo*, *Caption Photo*⟩
06. ⟨*City Photo is for City*, *CityPhoto City*⟩
07. ⟨*Travel Video is for Country*, *Clip Country*⟩
08. ⟨*Topic classifies Travel Video*, *Clip Subject*⟩
09. ⟨*Nr Hours is for Travel Video*, *Clip Length*⟩
10. ⟨*Nr Minutes is for Travel Video*, *Clip Length*⟩

Table 2: Direct relationship-set matches for mapping Figure 2 to Figure 4

match pair by $2 \times x + y$, where x is the number of recognition criteria marked plus 1 if either $=$ is marked for an object-set pair or $\Leftrightarrow$ is marked for a relationship-set pair and y is the number of type and constraint resolutions (i.e. the number of "u" and "d" marks in rows for IDS's 2-8)[†].

Algorithm *Generate Target-to-Source Mapping* proceeds by first doing direct matches for existing object and relationship sets and then doing derived matches. Although the algorithm does not iterate, later steps may alter the information recorded in previous steps. If for example, the algorithm decomposes a source object set (e.g. *Length* in Figure 4) and produces matching virtual object sets (e.g. *Nr Hours* and *Nr Minutes* in Figure 10), then as part of filling the column in the table for the decomposed derived source object sets, the algorithm rejects any proposed match from the target object sets used for naming the new source object sets to the decomposed source object set (e.g. rejects ⟨Nr Hours, Length⟩ and ⟨Nr Minutes, Length⟩). The algorithm rejects an existing column by making the confidence value -1, "considered but rejected."

We discussed and illustrated the recognition criteria[†] we use for Algorithm *Generate Target-to-Source Mapping* as we explained each case in Sections 3 and 4. The recognition criteria for an existing object set s, for example, consists of (1) checking for a source keyword in an object-set name or object-set type name or a comment associated with the object set, (2) checking for a source value that matches a target regular expression for s, and (3) checking to see whether any target relationship set connected to s has a potential match with a source relationship set according to the criteria for a relationship set. (This last recognition criterion depends on potential object set matches and is filled in as potential relationship matches, both existing and derived, are checked.) For criteria (2), the value criteria, we have filled-in the tables assuming reasonable database values,

[†]The study of how to generate confidence values is itself a major research project. Sophisticate techniques such as the use of the Dempster-Shafer theory of evidence as suggested in [46], for example, may be appropriate. We leave this research for future work.

[†]We also mentioned in the introduction that additional recognition criteria proposed by others can be used. It should be clear that we can expand the *Recognition Criteria* section of our table by adding a row for each criteria. It should also be clear that we can make our recognition criteria more fine-grained (by decomposing criteria) or more gross-grained (by aggregating) criteria.

	Matched Pairs										
Criteria	01	02	03	04	05	06	07	08	09	10	11
Recognition Criteria											
keyword	x		x	x							
value			x	x	x	x	x	x	x	x	x
structure	x	x			x	x	x	x	x	x	x
Special Considerations											
units (IDS 1)			x	x							
$=$	x		x	x							
$\Leftrightarrow$		x			x	x					
$\Leftarrow \not\Rightarrow$ (IDS 6)											d
$\not\Leftarrow \Rightarrow$ (IDS 7)							d	d	d		
$\not\Leftarrow \not\Rightarrow$ (IDS 6)										d	
$\not\Leftarrow \not\Rightarrow$ (IDS 8)										d	
partitioning (IDS 10)	d										
rel-set paths (IDS 14)								u	r	r	r
query (IDS 15)					d	d	u	d	d	d	d
Confidence	6	4	6	6	6	6	5	5	-1	-1	-1
Selected Pair (Req. 1–10)	x	x	x	x	x	x	x	x			

01. $\langle$*Video With City Scene*, $\rho_{Clip \leftarrow VideoWithCityScene}\pi_{Clip}City\_Clip\rangle$
02. $\langle$*City appears in Video With City Scene*, $\rho_{Clip \leftarrow VideoWithCityScene}City\_Clip\rangle$
03. $\langle$*Nr Hours*, $\pi_{NrHours}\epsilon_{Length\_NrHours\_NrMinutes}Length\rangle$
04. $\langle$*Nr Minutes*, $\pi_{NrMinutes}\epsilon_{Length\_NrHours\_NrMinutes}Length\rangle$
05. $\langle$*Nr Hours is for Travel Video*,
$\pi_{Clip\ NrHours}(Clip\_Length \bowtie \epsilon_{Length\_NrHours\_NrMinutes}Length)\rangle$
06. $\langle$*Nr Minutes is for Travel Video*,
$\pi_{Clip\ NrMinutes}(Clip\_Length \bowtie \epsilon_{Length\_NrHours\_NrMinutes}Length)\rangle$
07. $\langle$*US Exchange Rate is for Kind Of Money*,
$\pi_{Currency\ Rate}\sigma_{ForeignCurrency = \text{"US\$"}}Currency\_ForeignCurrency\_Rate\rangle$
08. $\langle$*City is in Country*, $\pi_{City\ Country}(\rho_{CityPhoto \leftarrow Photo}City\_CityPhoto \bowtie CountryPhoto\rangle$
09. $\langle$*City is in Country*,
$\pi_{City\ Country}(\rho_{VideoWithCityScene \leftarrow Clip}City\_VideoWithCityScene \bowtie Clip\_Country)\rangle$
10. $\langle$*Airport is for City*,
$\pi_{City\ InternationalAirport}(\rho_{CityPhoto \leftarrow Photo}City\_CityPhoto$
$\bowtie CountryPhoto \bowtie Country\_InternationalAirport\rangle$
11. $\langle$*Airport is for City*,
$\pi_{City\ InternationalAirport}(\rho_{VideoWithCityScene \leftarrow Clip}City\_VideoWithCityScene$
$\bowtie Clip\_Country \bowtie Country\_InternationalAirport)\rangle$

Table 3: Derived object-set and relationship-set matches for mapping Figure 2 to Figure 4

	Matched Pairs										
Criteria	01	02	03	04	05	06	07	08	09	10	11
Recognition Criteria											
keyword	x	x	x	x	x	x	x	x	x	x	x
value							x		x		x
structure	x	x			x	x	x		x		x
Special Considerations											
units (IDS 1)											x
$=$	x	x	x	x	x	x	x		x		x
$\supset$ (IDS 2)											
$\subset$ (IDS 3)											
$\not\sim$ (IDS 4)								r		r	
Confidence	6	6	4	4	6	6	8	-1	8	-1	8
Selected Pair (Req. 1–10)	x	x			x	x	x		x		x

01. ⟨*Country Map*, *Country Map*⟩
02. ⟨*City Map*, *City Map*⟩
03. ⟨*Map*, *Country Map*⟩
04. ⟨*Map*, *City Map*⟩
05. ⟨*Latitude*, *Latitude*⟩
06. ⟨*Longitude*, *Longitude*⟩
07. ⟨*Country*, *Country Name*⟩
08. ⟨*Country*, *Country Map*⟩
09. ⟨*City*, *City Name*⟩
10. ⟨*City*, *City Map*⟩
11. ⟨*Size*, *Nr Sq Km*⟩

Table 4: Direct matches for mapping Figure 2 to Figure 5

only some of which are given (i.e. in Figure 3).

In Sections 3 and 4 we also discussed and illustrated the special considerations we use for Algorithm *Generate Target-to-Source Mapping*. As part of this discussion, we enumerated all the Issue/Default/Suggestion (IDS) statements we use in our algorithm. We use this enumeration to reference specific IDS statements in the *Special Considerations* section of our tables.

We can select the "best" functional mapping that satisfies Requirements 1 through 10 by a backtracking algorithm that enumerates all maximal mappings (i.e. mappings to which no more pairs can be added without violating the requirements). Our backtracking algorithm ignores rejected pairs, but considers all others as possible components of maximal mappings. We then choose the "best" maximal mapping by summing the confidence values for each pair to get a confidence value for the mapping and taking the mapping with the largest total confidence value. If several have the largest confidence value, we can select arbitrarily, and in this case, we should also draw this circumstance to the attention of the user.

For our example, the marks in the *Selected Pair* row designate the set of ordered pairs for each of the three source-to-target mappings. In Table 4, for example, we do not include the pair in Column 03 (⟨*Map*, *Country Map*⟩) because Requirement 1 constrains the mapping to be injective and we have in Column 01 (⟨*Country Map*, *Country Map*⟩) with a higher confidence value. Similarly, we do not include the pair in Column 04 because of the higher confidence pair in Column 02. Two of the pairs in Table 4 have been rejected, Columns 08 and 09, and two of the pairs in Table 5 have also been rejected, Columns 03 and 04. We include all the rest to form our target-to-source mapping from the target about travel (Figure 2) to the source about maps (Figure 5).

	Matched Pairs							
Criteria	01	02	03	04	05	06	07	08
Recognition Criteria								
keyword	x							
value								
structure	x	x	x	x	x	x	x	x
Special Considerations								
units (IDS 1)								
$=$	x							
$\supset$ (IDS 2)								
$\subset$ (IDS 3)								
$\not\sim$ (IDS 4)								
subset connections (IDS 5)								
$\Leftrightarrow$							x	
$\Leftarrow$ $\not\Rightarrow$ (IDS 6)								
$\not\Leftarrow$ $\Rightarrow$ (IDS 7)					d			d
$\not\Leftarrow$ $\Rightarrow$ (IDS 8)					d			d
$\not\Leftarrow$ $\not\Rightarrow$ (IDS 6)						d		
$\not\Leftarrow$ $\not\Rightarrow$ (IDS 7)								
$\not\Leftarrow$ $\not\Rightarrow$ (IDS 8)						d		
multiple rel sets (IDS 9)								
partitioning (IDS 10)								
concatenating (IDS 11)								
multiple keys (IDS 12)								
obj-set paths (IDS 13)		d	r	r				
rel-set paths (IDS 14)								
query (IDS 15)					d	d	d	d
Confidence	6	2	-1	-1	4	4	4	4
Selected Pair (Req. 1–10)	x	x			x	x	x	x

01. $\langle$*Map*, $\rho_{CountryMap \leftarrow Map}CountryMap \cup \rho_{CityMap \leftarrow Map}CityMap\rangle$
02. $\langle$*Location*, $\pi_{Location}\alpha_{Location}(Country\_CountryName \bowtie Country\_Latitude \bowtie Country\_Longitude)\rangle$
03. $\langle$*Location*, $\pi_{Location}\alpha_{Location}(Country\_CountryName \bowtie Country\_Latitude \bowtie City\_Latitude \bowtie City\_Longitude)\rangle$
04. $\langle$*Location*, $\pi_{Location}\alpha_{Location}(Country\_CountryName \bowtie Country\_Longitude \bowtie City\_Longitude \bowtie City\_Latitude)\rangle$
05. $\langle$*Country Map is for Country*, $\pi_{CountryName\ CountryMap}(Country\_CountryMap \bowtie Country\_CountryName)\rangle$
06. $\langle$*City Map is for City*, $\pi_{CityMap\ CityName}(City\_CityMap \bowtie City\_CityName)$
07. $\langle$*Country Location*, $\pi_{CountryName\ Location}\alpha_{Location}(Country\_CountryName \bowtie Country\_Latitude \bowtie Country\_Longitude)\rangle$
08. $\langle$*Latitude Location Longitude*, $\pi_{Latitude\ Location\ Longitude}\alpha_{Location}(Country\_CountryName \bowtie Country\_Latitude \bowtie Country\_Longitude)\rangle$

Table 5: Derived matches for mapping Figure 2 to Figure 5

Criteria	*Matched Pairs* 01	02	03	04
Recognition Criteria				
keyword	x	x		
value	x	x		
structure	x	x	x	x
Special Considerations				
units (IDS 1)				
$=$	x	x		
$\supset$ (IDS 2)				
$\subset$ (IDS 3)				
$\not\sim$ (IDS 4)				
subset connections (IDS 5)				
$\Leftrightarrow$			x	
$\Leftarrow$ $\not\Rightarrow$ (IDS 6)				
$\not\Leftarrow$ $\Rightarrow$ (IDS 7)				
$\not\Leftarrow$ $\Rightarrow$ (IDS 8)				d
$\not\Leftarrow$ $\not\Rightarrow$ (IDS 6)				
$\not\Leftarrow$ $\not\Rightarrow$ (IDS 7)				
$\not\Leftarrow$ $\not\Rightarrow$ (IDS 8)				
multiple rel sets (IDS 9)				
partitioning (IDS 10)				
concatenating (IDS 11)				
multiple keys (IDS 12)				
obj-set paths (IDS 13)				
rel-set paths (IDS 14)			u	r
query (IDS 15)			d	d
Confidence	8	8	4	-1
Selected Pair (Req. 1–10)	x	x	x	

01. $\langle$*City*, *City*$\rangle$
02. $\langle$*Airport*, *Airport Code*$\rangle$
03. $\langle$*Airport is for City*, $\pi_{AirportCode\ City}(Airport\_AirportCode \bowtie Airport\_serves\_City)\rangle$
04. $\langle$*Airport is for City*, $\pi_{AirportCode\ City}(Airport\_AirportCode \bowtie Airport\_is\_located\_in\_City)\rangle$

Table 6: Matches for mapping Figure 2 to Figure 6

6. FORMAL PROPERTIES OF TARGET-TO-SOURCE MAPPINGS

In this section we discuss our stated goal of producing a valid interpretation for the target model given that the source interpretation for a target-to-source mapping is valid. Having specified the mapping algorithm in Section 5, we can now evaluate to what extent this goal has indeed been achieved.

In the following we assume that the source input to the algorithm consists of a populated OSM model instance that satisfies all declared integrity constraints. We then consider the mapping f obtained as output from our algorithm that generates target-to-source mappings, along with the corresponding generated target population. To check whether this target population satisfies the integrity constraints declared in the target input, we proceed in four steps:

- First, we provide a list of all integrity constraints used in the restricted subset of OSM as introduced in Section 2 (see Column 1 of Table 7). The items of this list are grouped into three classes. The first class contains the local constraints that refer to one individual object set. The second class contains the local constraints that refer to one individual relationship set. Finally, the third class contains the global ISA constraints.

- Second, we identify the OSM submodel instance in the target that includes only those object and relationship sets that map to the source. We argue that the object and relationship sets that map to a source constitute a proper OSM submodel instance as follows. (1) Discarding any relationship set always yields a proper OSM submodel instance. (2) Discarding any object set that has no attached relationship set and no specialization(s) in an ISA hierarchy also always yields a proper OSM submodel instance. Observe that once we have discarded an object set in an ISA hierarchy that has no specializations (and no connected relationship sets), its parent may then also have no specializations and no connected relationship sets and may also be discarded. Thus, we can discard object sets in an ISA hierarchy recursively, starting from the bottom until we (a) wish to stop, (b) discard the entire ISA hierarchy, or (c) encounter an object set that has a connecting relationship set. Although we may apply the second rule recursively, we claim that these are the only rules we use to form the OSM submodel instance whose items all match, and thus the OSM submodel instances we use are proper. Our claim follows from two observations. (A) By Rule (1), we can immediately discard all unmatched relationship sets. (B) An unmatched object set can have no matched connecting relationship sets (Req. 4) and can have no descendant specializations (direct or indirect) that are matched. For suppose that a descendent specialization is matched, then by the discussion in Section 4.1 we generate a match for all missing generalizations. The implication of Observation (B) is that we can (recursively) discard all unmatched object sets. Hence, once unmatched relationship sets have been discarded, we can discard any object set that stands alone (not in an ISA hierarchy), and we can recursively discard unmatched object sets bottom up until we either discard the entire ISA hierarchy or until we encounter a matched object set. If we encounter a matched object set in an ISA hierarchy, all its ancestor generalizations (direct and indirect) are guaranteed to have a match. Thus, we are left with only matched object sets and matched relationship sets.

- Third, we inspect each integrity constraint of the reduced target. Each constraint refers to a well-determined set of items (object sets or relationship sets) in the target model (see Column 2 of Table 7). If item a is in the domain of f, then a has an image b in the source which is different from all other images under f (Req. 1) and of the same sort (Req. 2). In this case a is populated exactly from b. If all target items referred to by a target constraint are mapped to some source item (see Column 3 of Table 7), then we can show that the satisfaction of the target constraint is implied by the assumed satisfaction of the source constraints. Here we have to distinguish two subcases: either an image b is directly declared in the source model (see Column 4 of Table 7), as discussed in Section 3, or b has been derived (see Column 5 of Table 7), as discussed in Section 4.

- Fourth, we consider the interaction of constraints. Usually, such an interaction will not occur.

kind of constraint	target items	source items	justification for existing source items	justification for derived source items
local constraints for one object set				derived by a query whose result has a single attribute
displayable type	a	b	Req. 5 (type compatibility) with IDS 1–4 (type compatibility and coercions)	for derived role (Section 4.1): Req. 4 with IDS 5 yield the required coercion; for derived generalization (Section 4.1): coercion is appropriately constructible; for derived component of composite in source (Section 4.2): by extraction routine with IDS 10, assuming that the target type allows nulls; for derived composite corresponding to composite in target (Section 4.2): by concatenation with IDS 11; for derived composite key object set corresponding to nondisplayable source object set (Section 4.3): possibly with IDS 12, similar to derived composite (Section 4.2)
nondisplayable type	a	b	Req. 5 (type compatibility); (Req. 6 is useful too but does not effect the formal properties)	for derived role (Section 4.1): Req. 4 with IDS 5 yield the required coercion; for derived generalization (Section 4.1): coercion appropriately constructible; for a derived nondisplayable source object set corresponding to a displayable source object set (Section 4.3): by generating unique OID's for displayable objects; for a derived nondisplayable source object set corresponding to a source relationship set (Section 4.3): possibly with IDS 13, by generating unique OID's for relationships
local constraints for one relationship set				derived by a query whose result has two or more attributes which must be identifiers of object sets
arity	a	b	Req. 3 (same arity)	in all cases: obvious by construction
referential integrity	a	b	Req. 4 (appropriate subset constraints for related object sets) with IDS 5	for associated relationship of a derived role (Section 4.1): inherited from assumed relationship; for derived relationship using a source path (Section 4.4) with IDS 14 and IDS 15: object sets of target relationship are already matched

minimum participation (upwards monotonic)	a	b	IDS 6 (source constraint equivalent, Case 1, or more restrictive, Case 2) or IDS 8 (source constraint less restrictive, Case 3; discard violating objects): may have an impact on referential integrity for another relationship, or for subset and union constraints that are adjusted by recursion	for associated relationship of a derived role (Section 4.1): inherited from assumed relationship, and by the definition of roles; for derived relationship set using a source path (Section 4.4) with IDS 14 and IDS 15: constraints for view appropriately computed
maximum participation (downwards monotonic)	a	b	IDS 6 (source constraint equivalent Case 1, or more restrictive, Case 2) or IDS 7 (source constraint less restrictive, Case 3; discard some relationships): may have an impact on minimum participation for an object set involved in the same relationship (IDS 8) that is adjusted by recursion	for associated relationship of a derived role (Section 4.1): inherited from assumed relationship; for derived relationship set using a source path (Section 4.4) with IDS 14 and IDS 15: constraints for view appropriately computed
functional dependency (downwards monotonic)	a	b	IDS 6 (source constraint equivalent Case 1, or more restrictive, Case 2) or IDS 7 (source constraint less restrictive, Case 3; discard some relationships): may have an impact on minimum participation for an object set involved in the same relationship (IDS 8) that is adjusted by recursion	for associated relationship of a derived role (Section 4.1): inherited from assumed relationship; for derived relationship set using a source path (Section 4.4) with IDS 14 and IDS 15: constraints for view appropriately computed
global ISA constraints				
subset constraint (downwards monotonic for a; upwards monotonic for b)	$a \subseteq b$	$a' \subseteq b'$	Req. 7(1) (ISA-compatible coercions) and Req. 8 (corresponding subset constraint in source)	arise only in the context of a union constraint
mutual exclusion constraint (downwards monotonic for a and b)	$a_1 \cap a_2 = \emptyset$	$a'_1 \cap a'_2 = \emptyset$	Req. 7 (ISA-compatible coercions) and Req. 9 (corresponding mutual exclusion constraint in source)	for derived partition (Section 4.1): population generated by the union constraint, and mutual exclusion is guaranteed by Req. 9 with Req. 7
union constraint	$a = a_1 \cup \ldots \cup a_n$	$a' = a'_1 \cup \ldots \cup a'_n$	Req. 7(1) (ISA-compatible coercions) and Req. 10 (corresponding union constraint in source)	for derived union (Section 4.1): population generated by the union constraint

Table 7: Justification for valid-interpretation claim

But IDS 7 and IDS 8 may require us to discard some part of the target population tentatively generated from the source population (or to change a target constraint). The default actions are designed to recursively adjust any violation of a constraint, and they are guaranteed to terminate successfully.†

Theorem 1 *Let t be a target OSM model instance and s be a source OSM model instance. Let f be a target-to-source mapping from t to s generated by Algorithm* Generate Target-to-Source Mapping*, and assume that t is populated from s according to f in accordance with the default rules in the IDS statements (or in accordance with user-supplied rules that are consistent with the default rules). Let t' be the OSM submodel instance whose object sets and relationship sets all map to s. Then, the generated population of t' is a valid interpretation.*

Proof. Case Analysis. For each case Table 7 summarizes the reasoning for the validity of the interpretation of the populated OSM submodel instance of the target that has matching source object and relationship sets. The entries in Table 7 refer to pertinent discussions in Sections 2 through 5, which we do not repeated here. Relying on this previous discussion, Steps one through three above outline the required soundness and completeness arguments necessary to complete the proof. □

7. MERGING TARGET-TO-SOURCE MAPPINGS

In Section 5 we summarized our approach for finding a mapping from a given target model to just one source model, and in Section 6 we showed that this mapping produces a valid interpretation for the OSM submodel instance populated from the single source. Let us now consider the case that more than one populated source model is available. We only briefly sketch the challenges and the options to resolve them. The basic challenge is to determine how the source models and their populations relate to each other. With respect to the various items under consideration, do they complement or do they overlap or are they conflicting? These questions about integrating sources have already been studied under several points of view—a rather general approach and its relationship to other work is reported in [41].

In general, all mutual relationships among the sources could have an impact on the final result of integration. Within the framework presented in this paper, however, we are assuming a much more focused situation: our clear emphasis is given by the target model, which can be used to direct the search for a good integration. More specifically, it appears reasonable to proceed in two steps:

- In a first step, we apply Algorithm *Generate Target-to-Source Mapping* to map the target model individually to each of n source models, and we then generate the corresponding target populations I_i, separately for each of the sources, $i = 1, \ldots, n$.
- In a second step, we somehow merge these populations. We can therefore take advantage of the results of the first step: any individual target population I_i is guaranteed to satisfy the constraints of the target model, and also has a Criteria-Consideration-Confidence Table generated by our algorithm that can be exploited.

Clearly, in the second step there are still some options left for deciding the basic questions about complementation, overlap, and conflict of items. For each of the crucial decisions we should propose an appropriate IDS, requesting user insights, clarifications, or qualifications, while providing a default. In general, we must deal with the following issues: (1) identify and process semantic equalities that are hidden by different syntactic representations, (2) take the union $I_1 \cup \ldots \cup I_n$ of all available data, and (3) discard some of the data if constraints cannot be satisfied.

semantic equalities: Identifying semantic equalities is necessary in order to detect overlaps between sources that are not already removed by the effects of duplicate removal when taking

†This loss of data may be unacceptable. But for the proof, we are assuming that when losing data is unacceptable, the user chooses to change the target constraints.

the union. Discovering semantically identical objects in different sources by using only syntactic material, however, can be difficult. Unique identifying lexical values may or may not exist, and even when such values do exist, they may have different representations and may be prone to error. Without human intervention, we may only be able to assert with a certain probability that objects from different sources are identical.

union: Taking the (set theoretical) union of available data is reasonable in order to get a best achievable covering of the items under consideration, thereby capturing all possibilities that one source is complemented by the others. Since the union depends on object identity, we can only produce a union that is as good as our ability to discover semantically identical objects. Further, in our framework we are taking the union of populated OSM submodel instances, which may not all be the same. We must therefore use the full target, from which each submodel instance is derived, to guide the formation of the union.

constraints: Once we form the union, we may or may not satisfy the constraints of the target model instance. Discarding data in the case of a conflict (i.e. if some constraint of the target model is violated by the equality-reduced union) appears to be indispensable in order to achieve constraint satisfaction purely automatically. Otherwise, given no additional semantic input, the algorithm would have to arbitrarily invent spurious data. Furthermore, some kinds of constraints can never be satisfied by adding data, for example a 1-maximum cardinality constraint. On the other hand, within our two step procedure we can always reach satisfaction by discarding data from the union, because any part of the union that originates from exactly one source does satisfy all constraints in the OSM submodel instance populated from that source. The hard problem, however, is to discover how to minimally remove individual facts such that the constraints are satisfied.

We leave the resolution of these issues within the context of our framework to future research.

8. SOURCE MODELING

Until now, we have considered the sources as given populated OSM model instances. In this section we briefly explain how to convert populated source data repositories to populated OSM model instances. The basic idea is to model the source directly as it stands when the source is a structured data repository, and to extract the data into an OSM model instance when the source is a semistructured or an unstructured data repository. Elsewhere, we have provided a detailed explanation for this conversion for relational-database sources [25] and for unstructured, data-rich sources [23]. Here, we sketch the idea for object-oriented databases.

Given an ODL (Object Definition Language) specification for ODMG [15], such as the one in Figure 13, we model it with an OSM model instance according to the following rules.

- Every attribute becomes an object set. If attributes in different interfaces have the same name and the same type, there is only one object set; otherwise there is an object set for each, with a distinguishing subscript appended for the second object set, the third object set, and so forth.
- The ODL type provides the type for an object set.
- The interface name becomes a nondisplayable object set except in the special case when there is only one attribute.
- An interface name that has become a nondisplayable object set functionally determines each attribute in its interface.
- Keys in an interface functionally determine the nondisplayable object set derived from the interface name.
- Relationships in an interface become relationship sets.

```
interface Country (
     extent Countries
     keys Country_Name ) : persistent
{
     attribute String Country_Name;
     attribute Integer Nr_Sq_Km;
     attribute String Latitude;
     attribute String Longitude;
     relationship Set¡Country_Map¿ has_Country_Map
          inverse Country_Map::is_for_Country;
};

interface City (
     extent Cities ) : persistent
     – Several cities may
     – have the same name.
{
     attribute String City_Name;
     attribute String Latitude;
     attribute String Longitude;
     attribute Set¡Image¿ City_Map;
     relationship Set¡City_Map¿ has_City_Map
          inverse City_Map::is_for_City;
};

interface Map_For_Country (
     extent Maps_For_Country ) : persistent
{
     attribute Image Country_Map;
     relationship Country is_for_Country
          inverse Country::has_Country_Map;
};

interface Map_For_City (
     extent Maps_For_City ) : persistent
{
     attribute Image City_Map;
     relationship City is_for_City
          inverse City::has_City_Map;
};
```

Fig. 13: Source ODL for ODMG database.

- In relationships, a bulk type (e.g. *Set*) specifies a "many" constraint, and the absence of a bulk type specifies a "one" constraint—these cardinalities determine the FD's for the relationship set.
- Optional constraints are added to generated relationship sets if (1) the connection is for an ODL attribute that appears in more than one interface or (2) the connection is for a relationship, unless it yields a relationship set that turns out to be the only relationship set connected to an object set.
- Comments are included as written.

Applying these rules to Figure 13 yields the OSM model instance in Figure 5[†]. Obtaining the data for the valid interpretation is straightforward. Nondisplayable object sets need system-generated OID's, one for each object. Other object sets, including those of type *Image*, can be obtained by projection.

[†]A more sophisticated set of rules can model more complex ODMG ODL, but this set is sufficient to illustrate the idea.

9. CONCLUSION

We have presented a framework for addressing the problems encountered in extracting information from heterogeneous information sources using ontologically specified target views. In this framework, we assume that a user wishes to obtain information with respect to a particular world view, the target view. This target view is specified independently of any particular source, and is thus a view that can be used for dynamically changing sources and for future (as yet nonexisting) sources.

As a fundamental feature of the proposed framework, we model both source views and the target view using the same conceptual model. This reduces the heterogeneity problem to same-model mappings and provides for a solid theoretical foundation. The model we use is OSM, which has a direct correspondence to first-order predicate calculus. The target-to-source mapping we produce maps target object and relationship sets to source object and relationship sets or derived source object and relationship sets. The requirements we place on this mapping (Req. 1 through 10) ensure that it has the properties needed so that we can properly load a target model instance from a populated source model instance. We proved that if a source has a valid interpretation, then the generated mapping produces a valid interpretation for the part of the target loaded from the source.

The target-to-source mapping generator can operate entirely automatically. The results, however, may not be satisfactory because the mapping generator makes default decisions about issues that arise. When these issues arise, the mapping generator interacts with a user through IDS (Issue/Default/Suggestion) statements. These IDS statements explain the issue involved, say what default action the system will take if the user does not intervene, and provide suggestions about what the user should do. The issues addressed include units (IDS 1), type compatibility (IDS 2 through 4), subset connections for relationship sets (IDS 5), stronger and weaker relationship constraints (IDS 6 through 8), multiple (existing) relationship sets between the same (existing) object sets (IDS 9), aggregate-value decomposition (IDS 10) and composition (IDS 11), keys for nondisplayable object sets (IDS 12), nondisplayable object sets matching existing or derived relationship sets (IDS 13), multiple relationship-paths for derived relationship sets (IDS 14), and user-specified queries for derived relationship sets (IDS 15).

The ontological description for a target OSM model instance enables target-to-source object-set and relationship-set matching through sample values and expected keywords. We express these sample object and relationship values and these expected keywords by regular expressions. As explained in our discussion, we take this approach to show how values and keywords can be a valuable asset in matching target and source object sets and relationship sets and to show how the matching results can be used to express confidence in the mapping. We believe, however, that techniques using thesauri for synonyms, hypernyms, and hyponyms as well as probabilistic structural matching can and should also be used in matching target and source object and relationship sets. Instead of repeating this work here, we provide a place for it in our framework and expect to use it, as well as value and keyword matching, in our work.

In general, our framework shows how to make the process of generating target-to-source mappings synergistic in the sense that the system (1) does all it can to provide solutions, (2) requests only specific information from the user, (3) records reasons for the decisions it makes, and (4) provides a measure of confidence in its results. We provide the system with the ontological knowledge that enables this level of synergy.

Besides this work, which we discussed in detail, we also briefly explored both upstream and downstream activities. On the upstream side, we discussed the process of source modeling, and we illustrated the process for ODMG object-oriented databases. On the downstream side, we discussed the process of merging target-to-source mappings for several sources. We pointed out that merging valid model instances for the same target model instance gives us a solid foundation from which to address the issues involved.

As for future work, we intend to materialize our framework in a prototype implementation and explore, in depth, both the upstream modeling activities and the downstream merging activities. As is often the case for significant research issues, even though progress has been made, there is

still much more to do.

Acknowledgements — Research for this paper was done while David W. Embley was on a sabbatical leave at Dortmund University.

REFERENCES

[1] S. Abiteboul, P. Buneman, T. Milo, D. Suciu, and J. Widom, editors. *Proceedings of the Workshop on Management of Semistructured Data*, Tucson, Arizona (1997).

[2] Y. Arens, C.Y. Chee, C.-N. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, **2**(2):127–159 (1993).

[3] P. Atzeni, G. Ausiello, C. Batini, and M. Moscarini. Inclusion and equivalence between relational database schemata. *Theoretical Computer Science*, **19**:267–285 (1982).

[4] P. Atzeni and V. DeAntonellis. *Relational Database Theory*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, California (1993).

[5] C. Batini, M. Lenzerini, and S.B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, **18**(4):323–364 (1986).

[6] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, **28**(1):54–59 (1999).

[7] J. Biskup. Achievements of relational database schema design theory revisited. In B. Thalheim and L.Libkin, editors, *Semantics in Databases*, volume LCNS 1358, pp. 29–54. Springer Verlag (1998).

[8] M. Bouzeghoub and I. Comyn-Wattiau. View integration by semantic unification and transformation of data structures. In *Proceedings of the 9th International Conference on the Entity-Relationship Approach (ER'90)*, Lausanne, Switzerland (1990).

[9] M.A. Bunge. *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World*. Reidel, Boston (1977).

[10] M.A. Bunge. *Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems*. Reidel, Boston (1979).

[11] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Source integration in data warehousing. In *Proceedings of the 9th International Workshop on Database and Expert Systems Applications (DEXA-98)*, pp. 192–197. IEEE Computer Society Press (1998).

[12] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. A principled approach to data integration and reconciliation in data warehousing. In *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)*, Heidelberg, Germany. CEUR Electronic Workshop Proceedings http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/ (1999).

[13] S. Castano and V. De Antonellis. Semantic dictionary design for database interoperability. In *Proceedings of 1997 IEEE International Conference on Data Engineering (ICDE'97)*, pp. 43–54, Birmingham, United Kingdom (1997).

[14] M. Castellanos. A methodology for semantically enriching interoperable databases. In *Proceedings of the 11th British National Conference on Databases (BNCOD-11)*, volume 696 of *Lecture Notes in Computer Science*, pp. 58–75, Keele, UK. Springer-Verlag (1993).

[15] R.G.G. Cattell. *The Object Database Standard: ODMG-93, Release 1.2*. Morgan Kaufmann publishers, Inc., San Francisco (1996).

[16] S. Cluet and T. Milo, editors. *Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB'99)*, Philadelphia, Pennsylvania (1999).

[17] W.W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of 1998 ACM SIGMOD International Conference on Management of Data*, pp. 201–212, Seattle, Washington (1998).

[18] L.M.L. Delcambre, D. Maier, R. Reddy, and L. Anderson. Structured maps: Modeling explicit semantics over a universe of information. *International Journal on Digital Libraries*, **1**(1):20–35 (1997).

[19] R.B. Doorenbos, O. Etzioni, and D.S. Weld. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agents*, pp. 39–48, Marina Del Rey, California (1997).

[20] O.M. Duschka and M.R. Genesereth. Infomaster—an information integration tool. In *International Workshop on Intelligent Information Integration*, Freiburg, Germany (1997).

[21] A.K. Elmagarmid and C. Pu. Introduction to the special issue on heterogeneous databases. *ACM Computing Surveys*, **22**(3):175–178 (1990).

[22] D.W. Embley. *Object Database Development: Concepts and Principles*. Addison-Wesley, Reading, Massachusetts (1998).

[23] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, **31**(3):227–251 (1999).

[24] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object-oriented Systems Analysis: A Model-Driven Approach*. Prentice Hall, Englewood Cliffs, New Jersey (1992).

[25] D.W. Embley and M. Xu. Relational database reverse engineering: A model-centric, transformational, interactive approach formalized in model theory. In *DEXA'97 Workshop Proceedings*, pp. 372–377, Toulouse, France. IEEE Computer Society Press (1997).

[26] J. Fowler, B. Perry, M. Nodine, and B. Bargmeyer. Agent-based semantic interoperability in InfoSleuth. *SIGMOD Record*, **28**(1):60–67 (1999).

[27] A. Gal. Semantic interoperability in information services: Experience with coopware. *SIGMOD Record*, **28**(1):68–75 (1999).

[28] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An extensible framework for data cleaning. Technical Report RR-3742, INRIA (1999).

[29] M. Garcia-Solaco, M. Castellanos, and F. Slator. Discovering interdatabase resemblance of classes for interoperable databases. In *Proceedings of RIDE-IMS'93 Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, pp. 26–33, Vienna, Austria (1993).

[30] M. Garcia-Solaco, F. Slator, and M. Castellanos. A structure based schema integration methodology. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pp. 505–512, Taipei, Taiwan (1995).

[31] M.R. Genesereth, A.M. Keller, and O.M. Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD International Conference on Management of Data*, pp. 539–542, Tucson, Arizona (1997).

[32] P.B. Golgher, A.H.F. Laender, A.S. da Silva, and Ribeiro-Neto. An example-based environment for wrapper generation. In S.W. Liddle, H.C. Mayr, and B. Thalheim, editors, *Proceedings of the 2nd International Conference on the World-Wide Web and Conceptual Modeling*, Lecture Notes in Computer Science, 1921, pp. 152–164, Salt Lake City, Utah (2000).

[33] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semistructured information from the Web. In *Proceedings of the Workshop on Management of Semistructured Data*, Tucson, Arizona (1997).

[34] S. Huffman and C. Baudin. Toward structured retrieval in semi-structured information spaces. In *Proceedings of the 1997 International Joint Conference on Artificial Intelligence*, pp. 751–756 (1997).

[35] R. Hull. Relative information capacity of simple relational database schemata. *SIAM Journal on Computing*, **15**(3):856–886 (1986).

[36] V. Kashyap and A. Sheth. Semantic and schematic similarities between database objects: A context-based approach. *The VLDB Journal*, **5**:276–304 (1996).

[37] T. Kirk, A.Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments* (1995).

[38] C. Knoblock, S. Minton, J. Ambite, N. Ashish, J. Margulis, J. Modi, I. Muslea, A. Philpot, and S. Tejada. Modeling web sources for information integration. In *Proceedings of AAAI 1998* (1998).

[39] J. Larson, S. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, **15**(4) (1989).

[40] A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Data Bases*, Mumbai (Bombay), India (1996).

[41] J. Lin and A.O. Mendelzon. Merging databases under constraints. *International Journal of Cooperative Information Systems*, **7**(1):55–76 (1998).

[42] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, **22**(3):267–293 (1990).

[43] J.A. Makowsky and E.V. Ravve. Dependency preserving refinements and the fundamental problem of database design. *Data and Knowledge Engineering*, **24**(3):277–312 (1998).

[44] I. Muslea, S. Minton, and C. Knoblock. STALKER: Learning extraction rules for semistructured, Web-based information sources. In *Proceedings of AAAI'98: Workshop on AI and Information Integration*, Madison, Wisconsin (1998).

[45] A. Ouksel and A. Iqbal. Ontologies are not the panacea in data integration: A flexible coordinator to mediate context construction. *Distributed and Parallel Databases*, **7**:1–29 (1999).

[46] A.M. Ouksel and C.F. Naiman. Coordinating context building in heterogeneous information systems. *Journal of Intelligent Information Systems*, **3**(2):151–183 (1994).

[47] A.M. Ouksel and A. Sheth. Semantic interoperability in global information systems—a brief introduction to the research area and the special section. *SIGMOD Record*, **28**(1):5–12 (1999).

[48] L. Palopoli, D. Saccà, and D. Ursino. Automatic derivation of terminological properties from database schemes. In *Database and Expert Systems Applications—9th International Conference—DEXA'98*, pp. 90–99, Vienna, Austria (1998).

[49] L. Palopoli, D. Saccà, and D. Ursino. An automatic technique for detecting type conflicts in database schemes. In *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management*, pp. 306–313, Bethesda, Maryland (1998).

[50] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous database systems. *ACM Transactions on Database Systems*, **19**(2):254–290 (1994).

[51] A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, **22**(3):183–236 (1990).

[52] M. Siegel and S. Madnick. A metadata approach to resolving semantic conflicts. In *Proceedings of the Seventeenth International Conference on Very Large Databases* (1991).

[53] K. Smith and L. Obrst. Unpacking the semantics of source and usage to perform semantic reconciliation in large-scale information systems. *SIGMOD Record*, **28**(1):26–31 (1999).

[54] S. Spaccapietra and C. Parent. View integration: A step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering*, **6**(2):258–274 (1994).

[55] D. Suciu and G. Vossen, editors. *Proceedings of the Third International Workshop on the Web and Databases (WebDB 2000)*, Dallas, Texas (2000).

[56] Jeffrey D. Ullman. Information integration using logical views. In Foto N. Afrati and Phokion Kolaitis, editors, *Proceedings of the 6th International Conference on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pp. 19–40, Delphi, Greece. Springer-Verlag (1997).