Flexible Data Integration and Ontology-Based Data Access to Medical Records

Lucas Zamboulis and Alexandra Poulovassilis and George Roussos

***Abstract*— The ASSIST project aims to facilitate cervical cancer research by integrating medical records containing both phenotypic and genotypic data, and residing in different medical centres or hospitals. The goal of ASSIST is to enable the evaluation of medical hypotheses and the conduct of association studies in an intuitive manner, thereby allowing medical researchers to identify risk factors that can then be used at the point of care to identify women who are at high risk of developing cervical cancer.**

This paper presents the current status of the ASSIST medical knowledgebase. In particular, we discuss the challenges faced in constructing the ASSIST integrated resource and in enabling query processing through a domain ontology, and the solutions provided using the AutoMed heterogeneous data integration system. We focus on data cleansing issues, on data integration issues related to integrating relational medical data sources into an independent domain ontology and also on query processing. Of particular interest is the challenge of providing an easily maintainable integrated resource in a setting where the data sources and the domain ontology are developed independently and are therefore both highly likely to evolve over time.

I. INTRODUCTION

The rapid development of biomedical informatics has lead to a wealth of information made available to researchers and practitioners. However, the lack of standards [8] has lead to an interoperability problem between many nodes containing overlapping and/or complementary information. For this reason, [19], [10] advocate the use of data integration in the biomedical domain. First, data integration leads to more reliable experiments, since it provides access to more data. Second, it allows a wider range of studies due to the increased breadth of information available. Third, in contrast with approaches that merely provide common access to data sources, it facilitates studies from a wide range of data resources by providing a single schema to the user.

Data integration approaches often provide a structural instead of a conceptual schema as the interface to the integrated resource, e.g. as in [23]. Although accurate, this schema is unfamiliar to users, and also the schema can capture only a small portion of the knowledge provided by domain experts.

[18] advocates *ontology-based data access*, i.e. a setting where an ontology provides access to one or more data sources, as a means for describing a domain at a high-level of abstraction, separating users' knowledge of the domain from the data layer. This setting also has the advantage of allowing domain knowledge to be expressed as logical formalisms, allowing inference mechanisms and ultimately converting a (possibly integrated) data source to a knowledgebase.

One of the systems that pioneered the interoperation of schema-based data sources with ontologies is TAMBIS [20]. This idea has been recently extended by [16], which also discusses the multiple roles of an ontology in such systems, i.e. a data source, a mediated schema and a system ontology.

In terms of ontology-based data access for relational data sources, OntoGrate [6] performs relational database integration and provides ontology-based data access with reasonable performance using query answering [15] by translating database schemas to ontologies and providing first order logic mappings between these ontologies and a global ontology. [18] follows a different integration strategy, by first federating all relational databases, then providing GLAV mappings between the federated schema and the global ontology. Given a query posed on the ontology, the GLAV mappings are used to generate SQL sub-queries that can be submitted to and evaluated by the relational data sources.

We provide an approach that combines schema- and ontology-based data integration. In particular, we use the AutoMed heterogeneous data integration system to integrate the medical data sources into a virtual relational integrated schema, and then semi-automatically transform that schema to a given domain ontology. Compared with [18], our approach leverages the existing schema-based data integration and query processing capabilities of AutoMed, while still allowing medical knowledge to be expressed within our system and be used to enrich user queries using query rewriting techniques such as those of [18], which will then be evaluated using AutoMed.

In this paper, we focus on the construction of the integrated resource using AutoMed as a first step in producing the ASSIST medical knowledgebase. Section II gives an overview of AutoMed to the level of detail necessary for this paper. Section III describes a number of aspects of integrating the relational data sources and exposing the integrated resource via an ontology, including the integration strategy followed, data cleansing issues and solutions, exposing the integrated resource using a domain ontology, and query processing. Section IV gives our conclusions and plans for future work.

II. OVERVIEW OF THE AUTOMED SYSTEM

AutoMed (`www.doc.ic.ac.uk/automed`) is a heterogeneous data transformation and integration system which offers the capability to handle virtual, materialised, and indeed hybrid data integration across multiple data models. It supports a low-level *hypergraph-based data model (HDM)*

Manuscript received June 11, 2008. This research has been partly funded by the ASSIST E.U. FP6 project. All authors are with the School of Computer Science and Information Systems, Birkbeck, University of London, WC1E 7HX, London, U.K., {`lucas,ap,gr`}`@dcs.bbk.ac.uk`. L. Zamboulis is also with Benchmark Performance Ltd., CO16 9PT, U.K.

and provides facilities for specifying higher-level modelling languages in terms of this HDM. An HDM schema consists of a set of nodes, edges and constraints, and each modelling construct of a higher-level modelling language is specified as some combination of HDM nodes, edges and constraints (see [11]). For any modelling language, $\mathcal{M}$, specified in this way (via the API of AutoMed's Model Definitions Repository — MDR), AutoMed provides a set of primitive schema transformations that can be applied to schema constructs expressed in $\mathcal{M}$. In particular, for every construct of $\mathcal{M}$ there is an `add` and a `delete` primitive transformation which add to/delete from a schema an instance of that construct. For those constructs of $\mathcal{M}$ which have textual names, there is also a `rename` primitive transformation.

Instances of modelling constructs within a particular schema are identified by means of their *scheme* enclosed within double chevrons $\langle\langle \ldots \rangle\rangle$ AutoMed schemas can be incrementally transformed by applying to them a sequence of primitive transformations, each adding, deleting or renaming just one schema construct (thus, in general, AutoMed schemas may contain constructs of more than one modelling language). A sequence of primitive transformations from one schema S_1 to another schema S_2 is termed a *pathway* from S_1 to S_2. All source, intermediate, and integrated schemas, and the pathways between them, are stored in AutoMed's Schemas & Transformations Repository (STR).

Each `add` and `delete` transformation is accompanied by a query specifying the extent of the added or deleted construct in terms of the rest of the constructs in the schema. This query is expressed in a comprehensions-based functional query language, IQL[1]. Also available are `extend` and `contract` primitive transformations which behave in the same way as `add` and `delete` except that they state that the extent of the new/removed construct cannot be precisely derived from the other constructs present in the schema. More specifically, each `extend` and `contract` transformation takes a pair of queries that specify a lower and an upper bound on the extent of the construct. The lower bound may be `Void` and the upper bound may be `Any`, which respectively indicate no known information about the lower or upper bound of the extent of the new construct.

The queries supplied with primitive transformations can be used to generate GAV, LAV or GLAV mappings, and to translate queries and data along a transformation pathway (see [12], [13], [14]). The queries supplied with primitive transformations also provide the necessary information for these transformations to be automatically *reversible*, in that each `add`/`extend` transformation is reversed by a `delete`/`contract` transformation with the same arguments, while each `rename` is reversed by a `rename` with the two arguments swapped.

[1]Such languages subsume query languages such as SQL-92 and OQL in expressiveness [3]. IQL also provides a common query language that queries written in various high level query languages, such as SQL, can be translated into and out of. Further details are given in [24].

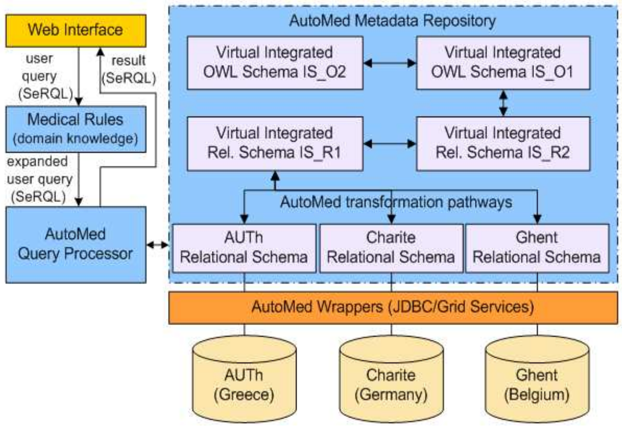

Fig. 1. The ASSIST Architecture.

III. THE ASSIST INTEGRATED RESOURCE

ASSIST (`assist.iti.gr`) aims to facilitate cervical cancer research through a system that integrates medical records residing in different medical centres or hospitals and containing various phenotypic and genotypic data. The derived integrated resource is enriched with medical knowledge encoded within an ontology produced by domain experts. The resulting medical knowledgebase allows researchers and physicians to evaluate hypotheses and conduct association studies in an intuitive way. The goal of ASSIST is to identify risk factors that can then be used at the point of care to identify women in high risk of developing cervical cancer.

Figure 1 shows the architecture of the medical knowledgebase. AutoMed is used to integrate three relational databases into the virtual[2] integrated relational schema IS_{R2}, and then transform this schema into the domain ontology IS_{O2}. Users query ASSIST through a web interface that helps them formulate queries using the ontology. The user query, internally expressed in SeRQL, is first expanded using medical rules expressed in the $\mathcal{EL}++$ description logics language [1], and is then submitted to the AutoMed query processor for evaluation.

The next sections focus on deriving the ASSIST integrated resource using AutoMed, and on query processing in our setting. In particular, Section III-A presents the ASSIST data sources, while Section III-B discusses the strategy followed to integrate them into the domain ontology. Section III-C focuses on the integration of the data sources into the virtual integrated relational schema IS_{R2} and Section III-D focuses on data cleansing. Section III-E describes the transformation of IS_{R2} into the domain ontology IS_{O2}. Section III-F discusses query processing in our setting.

A. *Exposing Medical Records in ASSIST*

ASSIST currently contains three relational data sources using different DBMSs. AutoMed can access these via JDBC, or, to circumvent security and firewall issues associated

[2]Both materialised and virtual integration are being explored in the ASSIST project and this paper is concerned with the latter scenario.

with JDBC access, using OGSA-DAI [3], which requires the installation of a thin Grid service layer at each data source.

As illustrated in Figure 1, each data source schema is imported in AutoMed via an AutoMed wrapper. These schemas are then incrementally transformed and integrated into one or more integrated schemas, using the API of AutoMed's STR to issue transformations and to create intermediate and final virtual schemas within this repository. AutoMed wrappers are also responsible for translating between IQL and the data source query language, in this case SQL.

Prior to exposing the data sources with JDBC or OGSA-DAI, sensitive data that could lead to patient identification was removed and/or anonymised. In particular, directly identifying data such as name and address was removed, while other data that could indirectly identify patients such as birth and visit dates were anonymised.

B. Integration Strategy

The ASSIST ontology aims to be a domain ontology, rather just an ontology interface to the current data sources, since these may be extended and their number may increase. As a result, the ontology and the data sources are assumed to be independent, and therefore differ not only in their data model, but also in the information they contain. For example, the current data sources model patients' visits only, while the ontology also contains the concept of cases, i.e. a case is defined to contain all visits made by a patient within a certain timeframe. Further, the data sources contain data that are not to be included in the integrated resource, and so the ontology does not represent the whole of the data stored in the data sources.

To accommodate these requirements, we created two virtual relational schemas (see Figure 1). IS_{R1} is used to integrate the whole of the information of the data sources, while IS_{R2} contains all information present in the domain ontology IS_{O2}, but not information from IS_{R1} that is not to be used in the ontology. As a result, IS_{O2} and IS_{R2} contain the same information, but are structurally quite different. We then defined transformation pathways between the data sources and IS_{R1}, between IS_{R1} and IS_{R2}, and finally between IS_{R2} and the domain ontology IS_{O2}. Note that pathway $IS_{R2} \leftrightarrow IS_{O1}$ is automatically generated, as discussed in Section III-E.

This solution presents two desirable properties. First, since IS_{R2} is a relational schema that contains the same information as the domain ontology, the creation of the overall transformation pathways from the data sources to this ontology is a two-step process and therefore more straightforward, compared to integrating each data source schema directly into the ontology. This is due to the difficulty of mapping between a relational schema and an ontology, caused by the fundamental differences in modelling decisions made when designing each one. For example, ontology classes are often modelled as data values in the relational model (e.g. `Caucasian` is a subclass of `Race` in our ontology, but a data value in the data sources).

Second, since IS_{R1} is modelled based on the data sources, mappings maintenance is much simpler, considering the possible evolution of either the ontology or of the data sources. In case one of the data sources LS_i evolves, only the pathway $LS_i \leftrightarrow IS_{R1} \leftrightarrow IS_{R2}$ is affected (if LS_i was extended with information not present in IS_{R1}), or even just $LS_i \leftrightarrow IS_{R1}$ (in all other cases). In case the ontology evolves, only the pathway $IS_{R1} \leftrightarrow IS_{R2} \leftrightarrow IS_{O2}$ is affected (if the ontology was extended with information not present in IS_{R2}), or oven just $IS_{R2} \leftrightarrow IS_{O2}$ (in all other cases).

This solution illustrates how mapping composition [2] for BAV is the straightforward process of appending transformation pathways, in contrast with the GAV/LAV/GLAV approaches. Furthermore, BAV readily supports the evolution of both integrated and data source schemas [7].

C. Relational Data Integration

The first step in producing the ASSIST integrated resource is to integrate the data source schemas into IS_{R1}. For each one of the data source schemas, LS_i, we issue AutoMed primitive transformations, producing schema LS'_i. This schema is now identical to IS_{R1} and so an id AutoMed transformation is issued between LS'_i and IS_{R1}, denoting that the two schemas contain the exact same constructs. As an example of this process, transformations 1–4 in Table I are some of the transformations used to integrate LS_{AUTh} with IS_{R1}. Note that, as shown in the transformations, the primary key of a relation in IS_{R1} is a set of tuples with arity 2, where the first component of each tuple is a string that identifies the provenance of the tuple (see also Section III-D).

The second step is to produce pathway $IS_{R1} \rightarrow IS_{R2}$. Transformations 8–10 in Table I are some of the transformations in this pathway, and show the relationship between relation $\langle\langle \mathsf{case} \rangle\rangle$ and relation $\langle\langle \mathsf{visits} \rangle\rangle$ in IS_{R2}. Currently, each patient visit is considered as a separate case, however this is going to change in the near future, and a case will be defined as the set of visits of a particular patient over a certain timeframe. Adapting the pathway for this is straightforward, e.g. if the timeframe is a single year, the IQL query for transformation 8 will become: $[\mathsf{genCaseId}\ \mathsf{v}|\{\mathsf{y},\mathsf{v}\} \leftarrow (\mathsf{group}\ [\{\mathsf{getYear}\ \mathsf{d},\mathsf{v}\}|\{\mathsf{v},\mathsf{d}\} \leftarrow \langle\langle \mathsf{visits},\mathsf{date}\rangle\rangle])]$, where function getYear expects a datetime input and returns the year, function group expects a collection of pairs, groups them on their first component, and returns a collection of collections of visit identifiers, and function genCaseId generates a case identifier for each collection of visits.

D. Data Cleansing

Data cleansing is the process of identifying inconsistent data within one or more data sources and taking measures to rectify or remove them. This includes erroneous, inaccurate and incomplete data that exist due to errors in constructing the data sources or due to merely putting together data

[3]OGSA-DAI (`www.ogsadai.org.uk`) is an open-source, extendable middleware product exposing resources on Grids via web services. OGSA-DAI supports relational, XML and text data sources and a variety of operations for them, including querying and updating.

TABLE I

AUTOMED TRANSFORMATIONS FOR SECTIONS III-C, III-D AND III-E

Namespace r2owl corresponds to `http://assist.dcs.bbk.ac.uk`, assist to `http://assist.iti.gr/assist_ontology.owl` and rdfs to `http://www.w3.org/2000/01/rdf-schema`, while authLSID stands for URN:LSID:assist.auth.gr. Functions sk1 and sk2 are Skolem functions, as discussed in Section III-E.

Some Transformations in Pathway $LS_{AUTh} \leftrightarrow IS_{R1}$
① $add(\langle\langle visits\rangle\rangle, [\{'authLSID.tblvisit', t\} \mid t \leftarrow \langle\langle tblvisit\rangle\rangle])$
② $add(\langle\langle visits, visit\_id\rangle\rangle, [\{\{'authLSID.tblvisit', t\}, \{'authLSID.tblvisit', t\}\} \mid t \leftarrow \langle\langle tblvisit\rangle\rangle])$
③ $add(\langle\langle visits, date\rangle\rangle, [\{\{'authLSID.tblvisit', t\}, v\} \mid \{t, v\} \leftarrow \langle\langle tblvisit, VisitDate\rangle\rangle])$
④ $add(\langle\langle visits, patient\_id\rangle\rangle, [\{\{'authLSID.tblvisit', t\}, \{'authLSID.tblpersonalinfo', p\}\} \mid \{t, p\} \leftarrow \langle\langle tblvisit, PID\rangle\rangle])$

Some Data Cleansing Transformations in Pathway $LS_{AUTh} \leftrightarrow IS_{R1}$
⑤ $add(\langle\langle patients\rangle\rangle, [\{'authLSID.patients', t\} \mid t \leftarrow \langle\langle tblpersonalinfo\rangle\rangle])$
⑥ $add(\langle\langle patients, patient\_id\rangle\rangle, [\{\{'authLSID.patients', t\}, \{'authLSID.patients', t\}\} \mid t \leftarrow \langle\langle tblpersonalinfo\rangle\rangle])$
⑦ $add(\langle\langle lifestyle, cigarettes\_per\_day\rangle\rangle, [\{\{'authLSID.tblpatientprofile', t\}, authSmokingMapping\ c\} \mid \{t, c\} \leftarrow \langle\langle tblpatientprofile, CigsPerDay\rangle\rangle; c >= 0; c <= 3])$

Some Transformations in Pathway $IS_{R1} \leftrightarrow IS_{R2}$
⑧ $add(\langle\langle case\rangle\rangle, \langle\langle visits\rangle\rangle)$ ⑨ $add(\langle\langle case, case\_id\rangle\rangle, \langle\langle visits, visits\_id\rangle\rangle)$ ⑩ $add(\langle\langle case, patient\_id\rangle\rangle, \langle\langle visits, patient\_id\rangle\rangle)$

Some Transformations in Pathway $IS_{R2} \leftrightarrow IS_{O1}$
⑪ $add(\langle\langle assist : case\rangle\rangle, [sk1\ c \mid c \leftarrow \langle\langle case\rangle\rangle])$
⑫ $add(\langle\langle assist : case\_id, assist : case, rdfs : Literal\rangle\rangle, [\{sk1\ c, cid\} \mid \{c, cid\} \leftarrow \langle\langle case, case\_id\rangle\rangle])$
⑬ $add(\langle\langle assist : patient\_id, assist : case, rdfs : Literal\rangle\rangle, [\{sk1\ c, pid\} \mid \{c, pid\} \leftarrow \langle\langle case, patient\_id\rangle\rangle])$
⑭ $add(\langle\langle assist : lifestyle\_id, assist : case, rdfs : Literal\rangle\rangle, [\{sk1\ c, lid\} \mid \{c, lid\} \leftarrow \langle\langle case, lifestyle\_id\rangle\rangle])$
⑮ $add(\langle\langle assist : case\_pk\rangle\rangle, [sk2\ c \mid c \leftarrow \langle\langle case\rangle\rangle])$ ⑯ $add(\langle\langle pk, assist : case, assist : case\_pk\rangle\rangle, [\{sk1\ c, sk2\ c\} \mid c \leftarrow \langle\langle case\rangle\rangle])$

Some Transformations in Pathway $IS_{O1} \leftrightarrow IS_{O2}$
⑰ $add(\langle\langle assist : Person\rangle\rangle, [pidlit \mid \{pid, pidlit\} \leftarrow \langle\langle r2owl : patient\_id, r2owl : patients, rdfs : Literal\rangle\rangle])$
⑱ $add(\langle\langle assist : Case\rangle\rangle, \langle\langle r2owl : Case\rangle\rangle)$
⑲ $add(\langle\langle assist : hasCase, assist : Person, assist : Case\rangle\rangle, [\{y, x\} \mid \{x, y\} \leftarrow \langle\langle r2owl : patient\_id, r2owl : case, rdfs : Literal\rangle\rangle])$

TABLE II

EXAMPLE USER QUERY q ON IS_{O2}.

Note that authLSID stands for URN:LSID:assist.auth.gr and chariteLSID for URN:LSID:assist.charite.de.

$$q = [\{p, v\} \mid \{p, c\} \leftarrow \langle\langle assist : hasCase, assist : Person, assist : Case\rangle\rangle; \{c, v\} \leftarrow \langle\langle assist : caseHasPart, assist : Case, assist : Visit\rangle\rangle]$$

$$\begin{aligned} q' = [&\{\{'authLSID : tblvisit', t1\}, \{'authLSID : tblpatientprofile', v\}, authSmokingMapping\ c1\} \mid \\ &\{t1, v\} \leftarrow authLSID : \langle\langle tblvisit, VID\rangle\rangle; \{t2, c1\} \leftarrow authLSID : \langle\langle tblpatientprofile, CigsPerDay\rangle\rangle; c1 >= 0; c <= 3; \\ &\{'authLSID : tblpatientprofile', v\} = \{'authLSID : tblpatientprofile', t2\}; (authSmokingMapping\ c1) =' More\_than\_20'] \\ ++\ [&\{\{'chariteLSID : patient', t3\}, \{'chariteLSID : tabPatientGeneralInfo', t3\}, c2\} \mid \\ &t3 \leftarrow chariteLSID : \langle\langle tabPatientGeneralInfo\rangle\rangle; \{t4, c2\} \leftarrow chariteLSID : \langle\langle tabPatientGeneralInfo, Cigs\_per\_day\rangle\rangle; \\ &\{'chariteLSID : tabPatientGeneralInfo', t3\} = \{'chariteLSID : tabPatientGeneralInfo', t4\}; c2 =' More\_than\_20'] \end{aligned}$$

sources that are individually consistent but that exhibit inconsistencies as a whole. In the following, we describe three data cleansing operations that are necessary in our setting.

1) Globally unique identifiers: Patients in ASSIST are assumed to be disjoint across databases and therefore so are all patient-related data. Since each data source uses locally unique identifiers, we need to ensure that no two records from two different resources are construed as equivalent. For this reason, we use to life science identifiers, or LSIDs [5], a Uniform Resource Name (URN) specification that provides a standardised naming scheme for life sciences entities. For example, the LSID `URN:LSID:assist.auth.gr.patients:126` refers to the row with primary key value 126 in table patients of the AUTh database — in this case the LSID issuing authority is `assist.auth.gr`.

Transformations ⑤ and ⑥ in Table I show how this is achieved for LS_{AUTh}. Note that, since in this case we are performing data cleansing for a data integration setting, each of these transformations is adding a construct of schema IS_{R1} (data integration aspect) and populating it by first performing data cleansing on the construct of LS_{AUTh}, which is deleted later in the pathway. If only data cleansing was required, e.g. for providing an export schema for a data source, we would instead replace these constructs.

This data cleansing operation has two benefits. First, it provides globally unique identifiers for primary key data and so joins on primary key attributes across data sources will not result in 'false positives', i.e. results that erroneously associate patient information across databases. Second, the use of URNs means that there is explicit provenance information within results of queries on the integrated schema, which may be of particular interest to users in certain cases.

2) Data values formats: The ASSIST ontology defines a set of permitted data values for each medical entity, which, in general, is different from the set of values permitted in each of the data sources. For example, the `Smoking` class in the ontology allows four different values, `No_Smoking`, `Less_than_10`, `10_to_20` and `More_than_20`, whereas the set of permitted values in the corresponding attribute in the AUTh data source, `tblpatientprofile.cigarettes_per_day`, is $[0, 1, 2, 3]$. To address this data values incompatibility, an IQL function is required to provide the association between the set of values used in the data source and the one used in the ontology. In our example, transformation 7 in Table I shows how function $\mathsf{authSmokingMapping}(\mathsf{Integer} \rightarrow \mathsf{String})$ maps data values about smoking between LS_{AUTh} and the ontology. Note that it is also possible to perform n-1, 1-m and $n-m$ data value associations using a suitable IQL function.

3) De-identification: Sensitive information that could lead to patient identification was removed from the ASSIST data sources prior to the integration effort. Since this may not always be the case, AutoMed is able to use techniques similar to the above to ensure data privacy. In particular, patient names and other sensitive information can be anonymised using an IQL function that, e.g. implements a salted MD5 hash function, commonly used in medical informatics [22]. Regarding the use of external de-identification services, it is possible to invoke such a service through an IQL function in order to keep this process external to AutoMed.

E. Ontology-based Data Access

Ontology-based data access to the integrated relational resource is achieved by a two-step process that forms the pathway $IS_{R2} \leftrightarrow IS_{O1} \leftrightarrow IS_{O2}$. First, we automatically translate schema IS_{R2} to an OWL-DL schema, IS_{O1}, and then we manually transform this schema to the ASSIST ontology IS_{O2}. Apart from generating part of the pathway automatically, this two-step approach has the advantage that the rest of the pathway, $IS_{O1} \leftrightarrow IS_{O2}$ is easier to create as there is no modelling language heterogeneity.

1) Relational-to-OWL-DL Translation: The relational-to-OWL-DL algorithm is based on [9], which describes the representation of relational databases in RDF. Similarly to [9], our representation of relational databases in OWL-DL can support single-column primary and foreign keys as well as composite ones. We now briefly present the algorithm and refer the reader to [25] for further details.

Given an input relational schema, in this case IS_{R2}, the translation algorithm has three parts. First, the algorithm translates the relations of the input schema. In our example, the algorithm first adds to IS_{R2} a class $\langle\langle \mathsf{C} \rangle\rangle$ for every relation $\langle\langle \mathsf{R} \rangle\rangle$ of IS_{R2} and a property $\langle\langle \mathsf{a}, \mathsf{c}, \mathsf{rdfs} : \mathsf{Literal} \rangle\rangle$ for each attribute $\langle\langle \mathsf{R}, \mathsf{a} \rangle\rangle$ of $\langle\langle \mathsf{R} \rangle\rangle$. The algorithm also adds a class $\langle\langle \mathsf{C_{pk}} \rangle\rangle$ and a property $\langle\langle \mathsf{pk}, \mathsf{C}, \mathsf{C_{pk}} \rangle\rangle$ for each relation. The extent of $\langle\langle \mathsf{C} \rangle\rangle$ is generated by skolemising the extent of the corresponding construct $\langle\langle \mathsf{R} \rangle\rangle$, i.e. the projection of the relation onto its primary key attributes, since all individuals in an ontology must be unique. The extent of $\langle\langle \mathsf{a}, \mathsf{c}, \mathsf{rdfs} : \mathsf{Literal} \rangle\rangle$ is generated similarly, i.e. by first projecting the relation onto the primary key attributes and the attribute itself, and then skolemising the primary key. The extent of $\langle\langle \mathsf{C_{pk}} \rangle\rangle$ is generated similarly to that of $\langle\langle \mathsf{C} \rangle\rangle$ but using a different Skolem function, while $\langle\langle \mathsf{pk}, \mathsf{C}, \mathsf{C_{pk}} \rangle\rangle$ is populated by pairs of instances from $\langle\langle \mathsf{C} \rangle\rangle$ and $\langle\langle \mathsf{C_{pk}} \rangle\rangle$, where the first item in each pair is equivalent to the second item.

The second part of the algorithm iterates through the foreign key constraints of the input relational schema and for each one that describes the referencing of attributes $\mathsf{a_i}$ of relation $\langle\langle \mathsf{R} \rangle\rangle$ from attributes $\mathsf{b_i}$ of relation $\langle\langle \mathsf{S} \rangle\rangle$, the algorithm creates classes $\langle\langle \mathsf{C_{R_{fk}}} \rangle\rangle$ and $\langle\langle \mathsf{C_{S_{fk}}} \rangle\rangle$ and properties $\langle\langle \mathsf{fk}, \mathsf{C_R}, \mathsf{C_{R_{fk}}} \rangle\rangle$, $\langle\langle \mathsf{fk}, \mathsf{C_S}, \mathsf{C_{S_{fk}}} \rangle\rangle$ and $\langle\langle \mathsf{fk}, \mathsf{C_{S_{fk}}}, \mathsf{C_{R_{fk}}} \rangle\rangle$, where fk is the name of the foreign key constraint. Classes $\langle\langle \mathsf{C_{R_{fk}}} \rangle\rangle$ and $\langle\langle \mathsf{C_{S_{fk}}} \rangle\rangle$ are populated by projecting the referenced and referencing attributes of $\langle\langle \mathsf{R} \rangle\rangle$ and $\langle\langle \mathsf{S} \rangle\rangle$, respectively, and skolemising them appropriately. Properties $\langle\langle \mathsf{fk}, \mathsf{C_R}, \mathsf{C_{R_{fk}}} \rangle\rangle$ and $\langle\langle \mathsf{fk}, \mathsf{C_S}, \mathsf{C_{S_{fk}}} \rangle\rangle$ are populated by projecting the corresponding relation onto the primary key attributes and the referenced/referencing attributes, respectively. The extent of property $\langle\langle \mathsf{fk}, \mathsf{C_{S_{fk}}}, \mathsf{C_{R_{fk}}} \rangle\rangle$ is populated by pairs where the first and second items are equivalent, and each comes from the skolemisation of the projection of the referencing attributes.

The third part then removes all relational constructs, using the newly created ontology constructs to specify their extents.

Transformations 11-16 show some of the transformations produced by the algorithm, and in particular the generation of the ontology constructs that correspond to relation $\langle\langle \mathsf{case} \rangle\rangle$.

2) OWL-DL Data Transformation: It is clear from the above that schema IS_{O1} bears a strong resemblance to the relational schema IS_{R2} from which it was generated, but is not necessarily identical to the domain ontology IS_{O2}. For this reason, we need to manually create pathway $IS_{O1} \leftrightarrow IS_{O2}$, and transformations 17-19 in Table I show some of the necessary transformations.

F. Query Processing

After constructing the integrated resource, a user can formulate a query against the ASSIST domain ontology via the web interface and submit it to ASSIST. This query, internally expressed in SeRQL, is expanded using the domain knowledge encoded within the ontology, and is then submitted to the AutoMed Query Processor (AQP) for evaluation. The AQP first translates this query into an equivalent IQL query Q. The AQP reformulates Q, which only contains terms from the ontology, into a query Q_{ref}, which contains terms from the data source schemas LS_i, using the transformation pathways $IS_{O2} \rightarrow IS_{O1} \rightarrow IS_{R2} \rightarrow IS_{R1} \rightarrow LS_i$. It then optimises Q_{ref}, produces the query plan and evaluates it against the data sources (see [24] for details on the query processing pipeline of the AQP).

As an example, consider an IQL query q on IS_{O2} (see Table II), which retrieves the cases in which patients smoke more than 20 cigarettes per day. This query is reformulated using pathways $IS_{O2} \rightarrow IS_{O1} \rightarrow IS_{R2} \rightarrow IS_{R1} \rightarrow LS_i$ (e.g. the first generator of q can be fully reformulated using transformations 19, 13, 10 and 4) and is then optimised into query q' as shown in Table II. This query contains two comprehensions, each of which can be evaluated using a single data source, LS_{AUTh} and $LS_{Charite}$, respectively. The second comprehension can be translated into SQL in full, however the first one cannot, since it is not possible to translate IQL function authSmokingMapping. As a result, the query plan produced for q' will contain 3 subqueries. These are submitted for evaluation to the data sources, the SQL results are translated back into IQL, and the AQP post-processes the IQL results to produce the result of Q.

IV. CONCLUSIONS AND FUTURE WORK

We have presented a flexible, scalable and easily maintainable solution to the problem of integrating relational data sources into a domain ontology, in a setting where both the data sources and the domain ontology are likely to evolve and where the number of data sources may increase. From a medical perspective, ASSIST will provide researchers with the ability to evaluate medical hypotheses and conduct association studies that were previously prohibitively difficult or even impossible, since such tasks require the integration of medical repositories and not mere access to them.

Note that additional ontologies may be created as resources holding information relevant to, but disjoint from, the ASSIST ontology. To enable querying across this set of integrated resources, a 'super-ontology' could then be created, which will integrate the ontologies of the integrated resources. This prospect, together with the ability to handle the evolution of both integrated and data source schemas, and the mapping composition strategy used to integrate the ASSIST data sources, exemplifies the flexibility and scalability of AutoMed's transformation-based approach.

We are currently working on enhancing the query processing capabilities of the ASSIST integration architecture. We plan to investigate combining ontology-based query answering techniques together with AutoMed's query processing capabilities over virtual integrated database schemas. We will also work on evaluating the integrated resource and the overall system in terms of query processing performance.

The current ASSIST data sources are relational, but this need not be the case in general and may indeed not be the case in ASSIST in the future. Medical records, stored locally or on the Web, frequently use unstructured or semi-structured formats, and the ability to integrate such resources using the same framework as structured resources is desirable. AutoMed can support the integration of such heterogeneous resources, as discussed in [17], [21].

Finally, further to extending the number and the type of data sources accessible through ASSIST, we also plan to investigate the interoperation of ASSIST with other biomedical systems. One such example is caBIG [4], which provides an international Grid on cancer-related research.

V. ACKNOWLEDGMENTS

We would like to thank Dikaios Papadogkonas for his insight into the semantics of the ASSIST data sources.

REFERENCES

[1] F. Baader, S. Brandt and C. Lutz. Pushing the EL envelope. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pp. 364-369, 2005.

[2] P.A. Bernstein, T.J. Green, S. Melnik and A. Nash. Implementing mapping composition. In *Proc. Int. Conference on Very Large Databases (VLDB'06)*, pp. 55–66, 2006.

[3] P. Buneman, L. Libkin, D. Suciu, V. Tannen, and L. Wong. Comprehension syntax. *SIGMOD Record*, 23(1):87–96, 1994.

[4] caBIG Strategic Planning Workspace. The Cancer Biomedical Informatics Grid (caBIG): infrastructure and applications for a worldwide research community. *Medinfo*, 12(Pt 1):330–334, 2007.

[5] T. Clark, S. Martin, T. Liefeld. Globally distributed object identification for biological knowledgebases. *Briefings in Bioinformatics*, 5(1):59–70, 2004.

[6] D. Dou et al. Integrating Databases into the Semantic Web through an Ontology-Based Framework. In *Proc. Int. Workshop on Semantic Web and Databases (SWDB'06 at ICDE'06)*, pp. 54–63, 2006.

[7] H. Fan and A. Poulovassilis. Schema evolution in data warehousing environments — a schema transformation-based approach. In *Proc. Int. Conference on Conceptual Modeling (ER'04)*, pp. 639–653, 2004.

[8] I.S. Kohane. Bioinformatics and medical informatics: the imperative to collaborate. *J. Am. Med. Infor. Assoc.*, 7(5):512–516, 2000.

[9] G. Lausen. Relational databases in RDF. In *Proc. Joint ODBIS & SWDB Workshop on Semantic Web, Ontologies, Databases*, 2007.

[10] B. Louie, P. Mork, F. Martin-Sanchez, A. Halevy and P. Tarczy-Hornoch. Data integration and genomic medicine. *J. Biomed. Inform.*, 40(1):5–16, 2007.

[11] P. McBrien and A. Poulovassilis. A uniform approach to inter-model transformations. In *Proc. Int. Conf. on Advanced Information Systems Engineering (CAiSE'99)*, pp. 333–348, 1999.

[12] P. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proc. International Conference on Data Engineering (ICDE'03)*, pp. 227–238, 2003.

[13] P. McBrien and A. Poulovassilis. Defining Peer-to-Peer Data Integration using Both as View Rules. In *Proc. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'03 at VLDB'03)*, pp. 91–107, 2003.

[14] P. McBrien and A. Poulovassilis. P2P query reformulation over Both-as-View data transformation rules. In *Proc. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'06 at VLDB'06)*, pp. 310–322, 2006.

[15] B. Motik, U. Sattler and R. Studer. Query Answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.

[16] P. Mork, R. Shaker and P. Tarczy-Hornoch. The Multiple Roles of Ontologies in the BioMediator Data Integration System. In *Proc. Data Integration for the Life Sciences (DILS'05)*, pp. 96–104, 2005.

[17] M. Maibaum, L. Zamboulis, G. Rimon, N. Martin, and A. Poulovassilis. Cluster based integration of heterogeneous biological databases using the AutoMed toolkit. In *Proc. Data Integration for the Life Sciences (DILS'05)*, pp. 191–207, 2005.

[18] A. Poggi et al. Linking data to ontologies. *J. Data Semantics*, X:133–173, 2008.

[19] W. Sujansky. Heterogeneous database integration in biomedicine. *J. Biomed. Inform.*, 34(4):285–298, 2001.

[20] R. Stevens et al. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. *Bioinformatics*, 16(2):184–186, 2000.

[21] D. Williams and A. Poulovassilis. Combining information extraction and data integration in the ESTEST system. In *Proc. Int Conf. on Software and Data Technologies (ICSOFT'06)*, pp. 13–21, 2006.

[22] D.H. Wyllie, T.E.A. Peto and D. Crook. MRSA bacteraemia in patients on arrival in hospital: a cohort study in Oxfordshire 1997-2003. *BMJ*, 331(7523):992–995, 2005.

[23] L. Zamboulis, H. Fan, K. Belhajjame, J. Siepen, A. Jones, N. Martin, A. Poulovassilis, S. Hubbard, S. M. Embury, and N. W. Paton. Data access and integration in the ISPIDER proteomics Grid. In *Proc. Data Integration in the Life Sciences (DILS'06)*, pp. 3–18, 2006.

[24] L. Zamboulis, S. Mittal, E. Jasper, H. Fan and A. Poulovassilis. Processing IQL queries in the AutoMed toolkit Version 1.2. AutoMed Technical Report 35, 2008.

[25] L. Zamboulis and A. Poulovassilis and J. Wang. Ontology-Assisted Data Transformation and Integration. In *Proc. Workshop on Ontologies-based Techniques for DataBases in Information Systems and Knowledge Systems (ODBIS'08 at VLDB'08)*, pp. TBC, 2008.