SKIMA: Semantic Knowledge and Information Management

Héctor Pérez-Urbina
Universidad de las Américas, Puebla
Ex hacienda Sta. Catarina Mártir, 72820, Cholula, México
hectorm.perezua@udlap.mx

Gennaro Bruno
IMAG-LSR, University of Grenoble
BP 72, 38402, Saint Martin d'Hères, France
Gennaro.Bruno@imag.fr

Genoveva Vargas-Solar
IMAG-LSR, University of Grenoble
BP 72, 38402, Saint Martin d'Hères, France
Genoveva.Vargas-Solar@imag.fr

Abstract

This paper describes SKIMA, a mediation system that gives transparent access to heterogeneous and distributed sources considering their semantics and the semantics of application requirements. It is based on a pivot model that abstracts concepts and semantic relations based on the $\mathcal{SHIQ}(\mathcal{D})$ description logic [10]. We use this model to represent application domains and source contents. Our approach provides an integrated and global view over local sources and couples it to the description of an application domain using semantic correspondences. In order to do so, it applies on inference to reason about these correspondences and other metadata, and exploits this knowledge to perform intelligent query processing tasks. We apply on a first prototype of the ADEMS framework to validate our approach in the Computer Assisted Instruction context by configuring SKIMA for a programmed instruction system.

1. Context and motivation

Currently there is an amazing quantity of heterogeneous information distributed over a large number of sources. Retrieving information is becoming a difficult task in which regular users use their knowledge in terms of formats, query languages and data models to obtain acceptable results. Instead of having to use many tools to retrieve different kinds of information most users prefer having a single tool that allows managing heterogeneous information that comes from different sources. This tool is generally known as mediation system [15].

The objective of a mediation system is to allow exploitation of many data sources through one access point. Users make queries in terms of the global schema and then the system translates them into queries in terms of local sources, retrieves the results, integrates them and returns the answer. In order to do so, there are three main issues to be considered: sources integration, schema integration and data integration.

- Sources integration means resolving heterogeneity both at semantic and structural levels. Heterogeneity may be structural if differences concern data models, query languages or internal protocols. Semantic heterogeneity expresses a difference in the meaning or in the interpretation of the same data.
- Schema integration consists of building a global schema by merging partial schemas. Three generation techniques have been proposed for resolving these problems. The first concerns schema merging and it is based on association rules between hand-written concepts. The second introduces structural matching algorithms which permit the automatic resolution of association rules before the merging. The last generation adopts semantic knowledge organized as ontologies or conceptual graphs and stores it in electronic vocabularies for reducing human intervention.
- Data integration assumes that schema conflicts are resolved, but introduces new problems as multiple values of the same entity in different sources, non-observance of integrity constraints, non-conformity of the measuring units, different data formats, etc. Most of these problems are resolved by translation rules, but their definition is not easy. Many techniques are explored such as data mining (searching relations between attributes values) and ontologies (taking into account meaning in a context). Description logics [3] have often been adopted in the context of these rules.

Despite of these three main aspects, classical mediation systems [7,8,14] fail to take into consideration semantics to perform tasks such as mediator configuration, source integration and query processing. To cope with this limitation current efforts include the development of mediation systems which apply on knowledge to perform some of their tasks, such as query reformulation or metadata representation [1,4,11]. However none of these approaches exploits reasoning on knowledge (semantics) to both, provide intelligent query processing and mediation system configuration.

This paper presents a mediation system that completely applies on semantics in order to perform intelligent query processing. Similarly, we use semantics to easily and automatically configure the mediation system for it to be well adapted to application requirements regarding a specific domain. We have chosen Computer Assisted Instruction (CAI) as our experimental domain because we consider it is a highly distributed and constantly evolving environment that requires the use of a great variety of data.

The remainder of this paper is organized as follows. Section 2 introduces our approach, a mediation system based on semantics for providing transparent access to distributed sources for applications. Section 3 presents SKIMA, a mediation system that considers semantics to perform intelligent mediation tasks. Section 4 presents a CAI experimental context and describes our validation prototype. Finally, Section 5 concludes this paper, discusses the main results and the future work.

2. General approach

SKIMA is the mechanism we propose, a mediation system based on semantics that enables applications to have transparent access to a set of sources. The system manages data in three levels: a domain schema, a global schema and a set of local schemata (see Figure 1). The domain schema represents a specific application domain; the set of local schemata is the representation of local sources content, and the global schema is an integrated and global view over sources. The mediation system couples local sources to the application domain through the global schema using semantic correspondences called mappings.

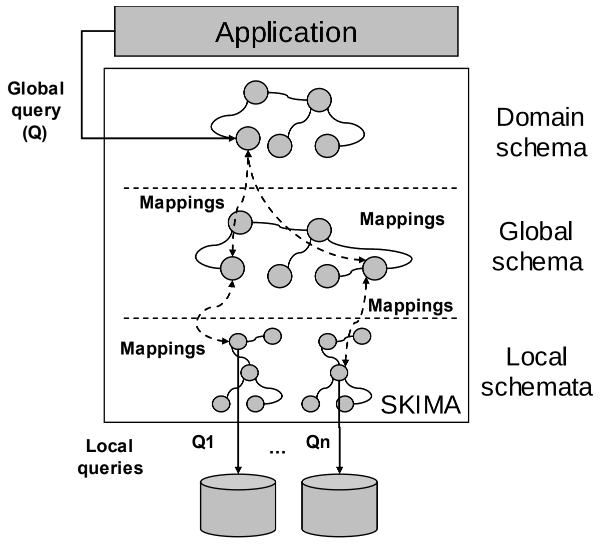


Figure 1. General approach.

Schemata are represented as sets of concepts and their semantic relations (ontologies) based on the $\mathcal{SHIQ}(\mathcal{D})$ description logic [10]. We use this high expressive language because it allows to model concepts and their relations as well as data types (attributes) and domain constraints (axioms). Schemata are loaded to an inference engine. Thanks to inference, our approach allows performing intelligent mediation tasks.

3. SKIMA

SKIMA is composed by four internal components: the parser, the reformulator, the rewriter and the evaluator. All four components interact with an inference engine that manages schemata and other metadata in order to perform their functions (see Figure 2). In a typical scenario an application uses the mediation system to query local sources in terms of its own domain. Before being executed, queries pass through a four-phase process.

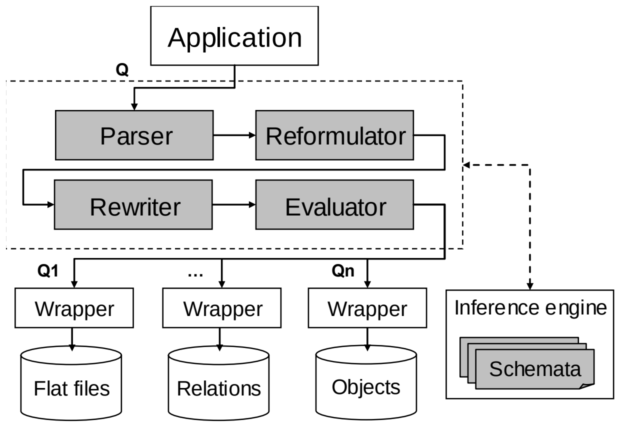


Figure 2. Architecture.

First, the parser verifies if the query is syntactically and semantically correct. If so, it is passed to the reformulator which gets a set of equivalent (or approximate) queries and passes them to the rewriter. The rewriter uses mappings in order to translate these queries expressed in terms of the domain schema into queries expressed in terms of the local schemata. Finally these queries and passed to the evaluator which sends them to a proper source in order to return results to the application. Further details on query processing are discussed in section 3.2.

3.1 Metadata

In order to perform query processing, SKIMA applies on metadata that, as schemata, are represented with ontologies and are loaded to the inference engine. We consider two types of metadata: source descriptions and mappings.

- Source descriptions. SKIMA is able to manage several sources; nevertheless, an application could have preferences regarding sources. For example some applications may prefer to access sources with a certain level of quality or cost. This is why our approach considers a set of source descriptions when processing queries. A source is described in terms of its quality, cost, availability and location. Each source has a source description stored in the source description ontology.

- Mappings. A mapping is a semantic relation between two concepts *a* and *b* from different schemata. There are three types of mappings: exact ($a \equiv b$), used when *a* and *b* are semantically equivalent; sound ($a \subseteq b$), used when *a* is subclass of *b*; and complete ($a \supseteq b$), used when *a* is superclass of *b*. Mappings are stored in the mapping ontology as axioms between concepts.

The source description ontology and the mapping ontology are managed in the inference engine.

3.2 Querying SKIMA

Queries are posed in terms of the domain schema. They correspond to a concept definition of the domain schema. For example, if there were a domain regarding family concepts, a query could be "women that have children", which is a concept definition of the domain schema. A query is represented as a concept tree and is attached to a set of source preferences that establishes which sources could be considered for evaluation.

A concept tree is a structure composed by a set of nodes that are part of the allowed vocabulary to define a concept. Allowed nodes include concepts and roles of the domain schema; logical operators (AND, OR and NOT); and universal (ALL) and existential (SOME) quantifiers. These nodes are used to build complex queries. There are building rules that describe the way a concept tree should be built for it to represent a concept definition. For instance leaves are reserved to concepts or roles, SOME nodes must have exactly two children (the first being a role while the second a concept), children of OR and AND nodes must be concepts, etc.

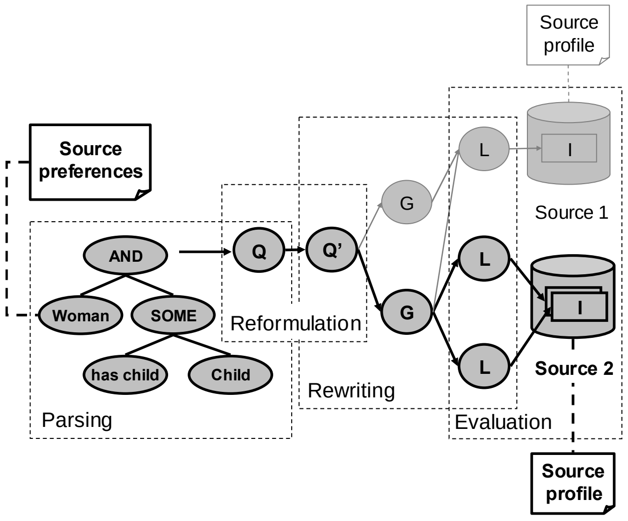


Figure 3. Query processing.

Consider once more the query "women that have children". In order to be executed, this query is represented with a concept tree and then given to the parser to start the query processing. Figure 3 shows the concept tree of this specific query and depicts the four phases of query processing: parsing, reformulation, rewriting and evaluation.

3.2.1 Parsing

Parsing involves verifying the query syntactically and semantically. The parser receives a concept tree and verifies if it represents a concept by checking if building rules are respected. If the concept tree is well formed *i.e.,* is syntactically correct, the parser builds an internal representation of it called query expression (Q) and verifies if it is a satisfiable concept in the domain schema (its definition respects all constraints). If so, Q is semantically correct and the parser passes it to the reformulator, otherwise the query is not executed.

We would like to point out the convenience and importance of verifying the semantics of a query. If an application posed the query "*people who are women and men"*, despite the query is syntactically correct, as it is not possible that a person is a woman and a man, the query would not be executed. Semantic verification is accomplished by simply asking the inference engine whether a concept (Q) is satisfiable in a specific ontology (the domain schema).

3.2.2 Reformulation

In a typical scenario, there are not mappings for all domain concepts; this is why it is important to reformulate Q in order to increase the possibility of success in the rewriting phase. Reformulating consists of getting a set of equivalent (or approximate) domain concepts of a given query expression for which at least one mapping exists.

Consider mappings shown in Figure 4. As can be seen, it is stated that "*women with children"* (domain schema) are equivalent to "*mothers"* (global schema). If an application posed the query "*women"*, reformulation would be necessary to retrieve any results given the fact the concept "*Woman*" has no associated mappings. The reformulator would discover that "*women*" has one approximate reformulation: "*women that have children*" and would continue the process letting the application know that there are only approximate results.

Reformulation is accomplished by reasoning on domain concepts and mappings. It is necessary to get equivalent or approximate concepts of Q and then, for each of these reformulations to verify that there is at least one mapping. Only the reformulations that have associated mappings are passed to the rewriter. The reformulator completely applies on subsumption reasoning tasks to perform the whole process.

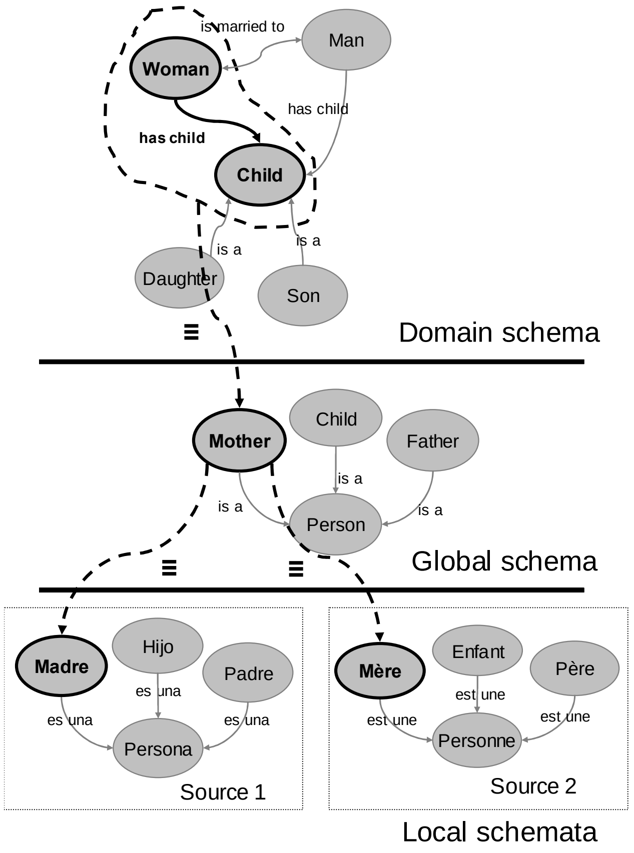


Figure 4. Mappings between schemata.

3.2.3 Rewriting

Rewriting consists in translating a set of reformulations to a set of local concepts. It is done in two phases; the first phase consists of translating reformulations to a set of global concepts while the second phase consists of translating the resultant global concepts to a set of local concepts.

Figure 4 shows an example of exact mappings between domain, global and local schemata. Continuing with the example on reformulations, if the only reformulation of the given query expression Q were "*women that have children*" (Q'), the rewriter would use mappings to discover that Q' corresponds to "*mothers*" in the global schema and to "*madres*" and "*mères*" in the local schemata, thus, it would pass the local concepts "*Madre@Source1*" and "*Mère@Source2*" to the evaluator.

Rewriting is accomplished by reasoning on the mapping ontology. In the first phase the rewriter asks the inference engine to retrieve the list of equivalent (or approximate) global concepts for each reformulation. Similarly, in the second phase the inference engine is asked to retrieve the list of equivalent (or approximate) local concepts for each global concept. The rewriter, as the reformulator, completely applies on subsumption reasoning tasks to perform the whole process.

3.2.4 Evaluation

Evaluation consists of returning results of a rewritten query *i.e.,* a set of local concepts, to the application. The evaluator receives a set of local concepts and sends them to the proper wrappers for results to be retrieved in local sources. Once the evaluator gets results from sources, it combines and integrates them, and finally it returns them to the application.

In order to choose proper wrappers, the evaluator reasons on the source description ontology. It asks the inference engine to retrieve what the source of each local concept is and then it gets the corresponding source profile. Only if this profile is compatible to the source preferences attached to the query, the local concept is sent to the corresponding wrapper.

3.2.5 Performance

As already said, SKIMA applies on inference to perform its specific tasks. One could guess there should be no problems (in terms of performance) while handling a small number of concepts and be more interested on performance with many concepts and/or axioms. To this matter we could say that performance depends, almost exclusively, on the used inference engine.

We decided to work with Racer [13] (Renamed ABox and Concept Expression Reasoner) because it can handle TBoxes with generalized concept inclusions, ABoxes (based on the unique name assumption) and concrete domains. It provides the means to verify concept consistency and concept subsumption w.r.t. a T-box, and to find inconsistent concepts and the parents and/or children of a concept, among other useful information.

In terms of performance, a first study was conducted that uses a very complex ontology derived from a real-world application with hundreds of named concepts and several general axioms. Racer can classify all concepts within seconds (for further details see [12]). We believe this study demonstrates that Racer is capable of handling very complex ontologies maintaining a good performance.

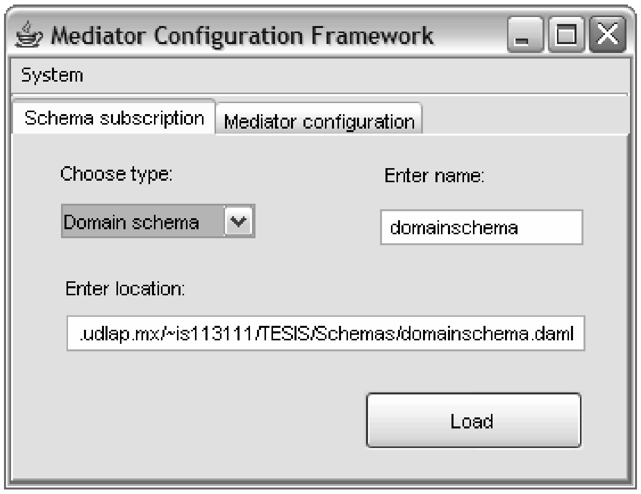
3.3 Configuring SKIMA

In order for an application to use SKIMA it is necessary to configure it *i.e.*, to provide schemata and metadata (mappings and source descriptions). In order to configure the mediation system we apply on the mediator configuration framework (MCF). MCF contains a collection of schemata and metadata, and it allows users to choose a set of schemata and metadata in order to configure their own SKIMA. MCF is composed by three components: the schema subscription module, the mediator configuration module and the schema base.

The schema subscription module is used to add schemata and metadata to a repository called schema base. The mediator configuration module provides the means to choose schemata and metadata from the schema base given a set of application preferences called schema profile. The schema profile is a set of preferred schema characteristics. It is used to specify a domain schema, a global schema and a set of local schemata with their related mappings from the schema base.

A schema profile is composed by two parts: the global schema and the domain schema description in terms of their names and the local schemata description as a set of preferred source characteristics in terms of quality, cost, availability and location. Schema profiles are modeled as ontologies as well and they are also stored in the schema base.

MCF has two functions: (1) schema and metadata subscription and (2) mediator configuration. The schema subscription module is used to subscribe schemata and metadata to the schema base. It receives a set of schemata and their related metadata and stores them in the schema base. The mediator configuration module is used to configure a specific SKIMA from a given schema profile.



Choose type: Domain schema

Enter name: domainschema

Enter location: .udlap.mx/~is113111/TESIS/Schemas/domainschema.daml

Figure 5. Mediator configuration framework.

Figure 5 shows MCF with its two sections: schema subscription and mediator configuration. To subscribe a schema means loading it into the schema base. In order to load a schema one must specify its type (domain, global, local, etc.), its name and its location. In order to configure a mediator, MCF retrieves the list of schema profiles contained in the schema base. Once a specific profile is selected, the framework searches the correspondent schemata and metadata and returns them to the user.

MCF is a prototype of ADEMS, a general purpose framework used to instantiate and to configure mediators taking into account application and source semantics [6]. MCF interacts with an inference engine in order to perform its functions based on reasoning. We will not discuss MCF details in this paper for lack of space.

4. Experimentation

Our experimentation context is CAI (Computer Assisted Instruction) because we consider it is a highly distributed and constantly evolving environment that requires the use of a great variety of data. CAI in general refers to the use of a computer as a tool within the educational process. CAI refers to practice activities, tutorials or simulations offered by computers as supplement of traditional education [5,9].

Our validation prototype is a very simple PI (Programmed Instruction) system. PI is a method of presenting new subject matter to students in a graded sequence of controlled steps [2]. It is based on the following algorithm:

```
1. Present initial unit
2. While there are units to present
3.     The student reads, assimilates and integrates the
       presented information
4.     The system asks the student something related to
       the unit that has just been presented
5.     If the unit is considered to be approved then
6.             Present next unit
```

Students work through the programmed material at their own speed. After each step they test their comprehension by answering an examination question(s). They are then immediately shown the correct answer or the following unit.

4.1 Schemata and metadata

The domain schema represents the PI domain (see Figure 6) and contains the following concepts:

- Course represents courses offered by the system. A course is composed by a set of sections or unities, it is identified by a name and it is related to a set of keywords. Every course is managed by an administrator and it is related to the set of learners (students) that take the course and to the facilitator (teacher) that is responsible for the course.
- Section represents unities that compose courses. A section is identified by a name and can be part of one or more courses in a certain order. Each section has a topic and it is related to a set of resources and to an exercise.
- Topic is used to represent a list of topics. These topics correspond to the topics of courses and sections of the system. More than one course or section may share the same topic.

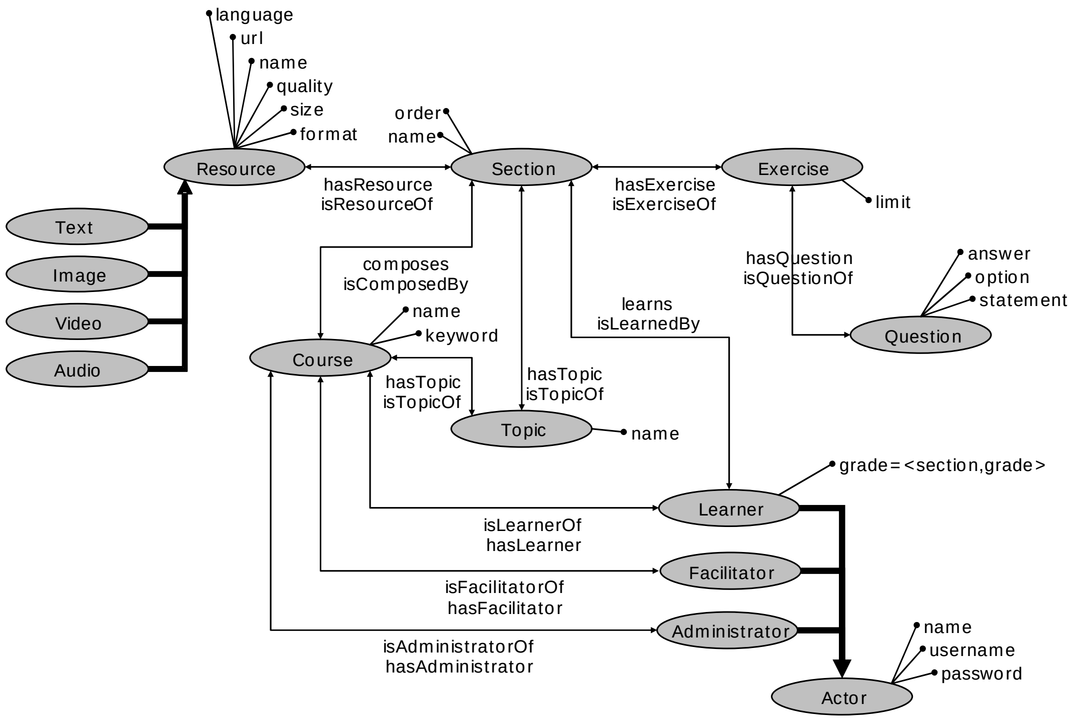


Figure 6. Programmed instruction domain.

- Exercise is used to represent evaluations. At the end of every section learners are given an evaluation to determine whether they continue with the following section. These evaluations consist of a set of questions and are represented by Exercise. Each exercise has a limit that determines the minimum needed correct answers for the learner to approve the associated section.
- Question represents the questions we use to compose exercises. Each question has a statement, a set of options and a correct answer.
- Resource represents a set of resources that are used by sections. We consider four kinds of resources: text, image, video and audio. Text represents plain and rich text documents, presentations, worksheets and Web pages. Image, Video and Audio represent images, videos and audios of any format. Resources are described in terms of their name, location, format, language, quality and size.
- Actor represents the actors of the system; every actor is identified by a name, a username and a password. There are three kinds of actors: learners, facilitators and administrators. Learners or students are related to the course they are taking, the section they are currently studying and the grades they got in previous sections. Facilitators or teachers are related to the course(s) they teach. Administrators are related to the course(s) they are responsible for.

We make the assumption that the global schema is a copy of the domain schema. Classes from the domain schema are related to their corresponding class from the global schema with an exact mapping. The local schema is composed by the representation of sources that store courses, sections, resources and actors.

We assume that all sources share a source profile that is compatible with a set of preferences to which all queries made by the PI system are attached. Source profiles are stored in the source description ontology while mappings between the local schema and the global schema are stored in the mapping ontology.

4.2 Querying local sources

We consider two concepts and two sources in order to illustrate query processing within the PI domain. Figure 7 shows the two concepts of the global schema we are interested on and corresponding pertinent local schemata. Source 1 contains text resources while Source 2 contains sections that have videos.

We consider the following mappings:

- Texts with format PDF are equivalent to Acrobat documents in Source 1: *(AND Text (EQUAL format “PDF”)) ≡ AcrobatDocument@Source1.*

- Sections that have resources which are videos are equivalent to sections in Source 2: *(AND Section (SOME hasResource Video)) ≡ Section@Source2*.

Imagine the following queries are made to SKIMA by the PI system:

- Q1. "Texts with format PDF": *(AND Text (EQUAL format "PDF"))*.
- Q2. "Sections that have videos whose format is MPEG": *(AND Section (SOME hasResource (AND Video (EQUAL format "MPEG"))))*.
- Q3. "All resources": *Resource*.

In order to execute these queries, SKIMA rewrites them based on existent mappings. Q1 is rewritten to *AcrobatDocument@Source1* and the mediator returns exact results. Q2 is rewritten to *Section@Source2* and complete results are returned because Source 2 contains sections that have videos with all kinds of formats, not only MPEG. Q3 is rewritten to *Documents@Source1* which has sound results because Source 1 only contains text resources, not all kinds of resources.

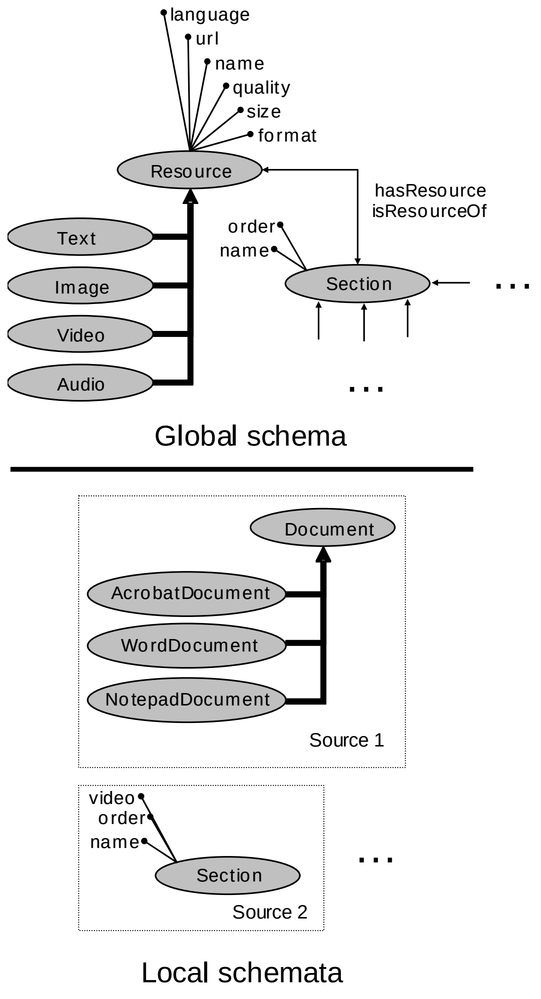
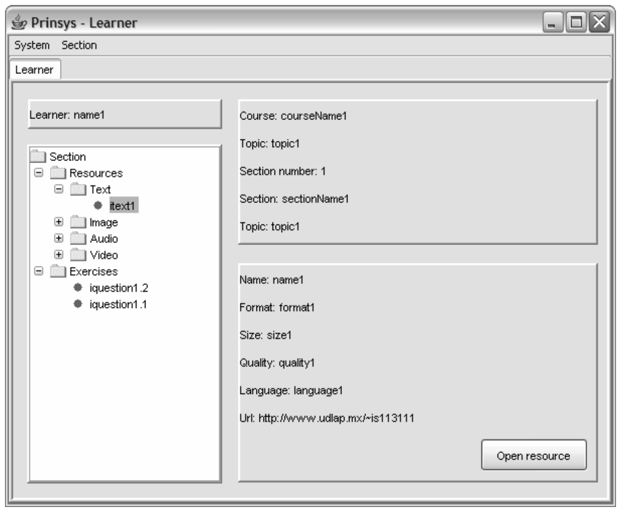


Figure 7. Schemata.

4.3 Using the PI system

In order to use the PI system the user *i.e.*, a learner, has to enter his/her username and password. If this information is valid the user logs in and the system presents the specific section he/she has to study of the course he/she is taking. The system shows the name of the user and information about the section and the correspondent course. It also shows the list of associated resources and exercises (see Figure 8).

When the user clicks on a resource, the system retrieves its metadata and allows the user to open it. In order to continue with the following section, the learner must take the section evaluation. When the user clicks on an exercise the system shows a question and a list of possible answers (Pressey's model [2]). If the user gives as many correct answers as defined in the domain schema he/she may continue with the following section (until there are not more sections to be taken), otherwise, he/she will have to recheck the resources and take the evaluation again. The PI system uses SKIMA to perform section evaluation and to retrieve all the information it shows to the user.



Learner: name1

Course: courseName1

Topic: topic1

Section number: 1

Section: sectionName1

Topic: topic1

Name: name1

Format: format1

Size: size1

Quality: quality1

Language: language1

Url: http://www.udlap.mx/~is113111

Figure 8. Programmed instruction system.

5. Current status

We have presented SKIMA as a mediation system that considers semantics to integrate heterogeneous local sources and to perform intelligent query processing. In our approach we use inference to both retrieve explicit knowledge and automatically discover implicit knowledge. It could be interesting for SKIMA to be validated by experts in a certain domain. Experts would configure the mediation system in order to decide whether it behaves as expected. It is important to consider that reasoning with application and source semantics can be computational expensive specially when handling large amounts of information. This situation motivates the interest of studying the impact that large amounts of information have on performance. We consider it is important studying and implementing query optimization techniques in order to make query processing as inexpensive as possible.

We have also introduced MCF, a framework that configures mediators in an intelligent way by reasoning on metadata. Perspectives related to MCF include its future improvement as a complete ADEMS framework [6]. ADEMS is currently under construction. The complete framework will allow building more complex mediation system architectures (centralized, hierarchical or peer-to-peer) by configuring reusable and cooperative mediators. Such mediators will provide interactive query processing which will enable users to drive the process through the application by choosing preferred reformulations or rewritings. Mediators will also enhance query expressiveness by taking into account query projections and filters on concepts and attributes.

6. References

[1] Arens, Y., Knoblock, C., Shen, W., *Query reformulation for dynamic information integration,* Journal of Intelligent Information Systems – Special Issue on Intelligent Information Integration, 6(2/3): 99-130, 1996.

[2] Ayala, G., *Educación Asistida por Computadora*, http://mail.udlap.mx/~ayalasan/ambientesDeAprendizaje/cap01.html, 2003.

[3] Baader, F., Nutt, W., *Basic description logics*, chapter 2, pages 47-100, Cambridge University Press, 2003.

[4] Baker., P., Brass, A., Bechhofer, S., Goble, C., Paton, N., Stevens, R., *TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources*, In Janice Glasgow, Tim Littlejohn, Francis Major, Richard Lathrop, David Sankoff, and Chistoph Sensen, editors, 6th International Conference on Intelligent Systems for Molecular Biology, pages 25-34, Montreal, Canada, 1998.

[5] Bangert-Drowns, R. L., *Meta-Analysis of Findings on Computer-Based Education with Precollege Students*, Annual Meeting of the American Educational Research Association, Chicago, IL, March – April, 1985.

[6] Bruno, G., Vargas-Solar, G., Collet, C., *ADEMS, an Adaptable and Extensible Mediation Framework: application to biological sources*, In Proceedings of the Workshop on Advances in Databases and Information Retrieval, ISBN: 970-36-0070-0*,* ENC'03, Tlaxcala, Mexico, September, 2003.

[7] Carey, M., Haas, L., *Towards Heterogeneous Multimedia Information Systems: The Garlic Project,* RIDE-COM, 1995.

[8] Chawathe, S., García-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., Widom, J., *The TSIMMIS project: Integration of heterogeneous information sources*, In 16th Meeting of the Information Processing Society of Japan, pages 7-18, Tokyo, Japan, 1994.

[9] Grimes, D. M., *Computers for Learning: The Uses of Computer Assisted Instruction (CAI) in California Public Schools*, Sacramento, CA: California State Department of Education, 1977.

[10] Horrocks, I., Sattler, U., Tobies, S., *Reasoning with Individuals for the Description Logic SHIQ*, In Proceedings of 17th International Conference on Automated Deduction, ISBN: 3-540-67664-3, 2000.

[11] Kirk, T., Levy, A., Sagiv, Y., Srivastava, D., *The Information Manifold,* In C. Knoblock and A. Levy, editors, Information Gathering from Heterogeneous, Distributed Environments, Stanford University, Stanford, California, 1995.

[12] Möller, R., Haarslev, V., *Description Logics Systems*, chapter 8, pages 282-305. Cambridge University Press, 2003.

[13] Möller, R., Haarslev, V., *Racer system description.* In International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy, 2001.

[14] Tomasic, A., Raschid, L., Valduriez, P., *Scaling Access to Heterogeneous Data Sources with DISCO*, Knowledge and Data Engineering, 10(5):808-823, 1998.

[15] Wiederhold, G., "*Mediators in the architecture of future information systems*", Computer, pages 38-49, 1992.