# An Integration Method for the Specification of Rule–Oriented Mediators[*]

H. Wache Th. Scholz H. Stieghahn
Center for Computing Technologies
University of Bremen
28334 Bremen
Germany
[wache|chuzakh|hoschi]@tzi.de

B. König–Ries
Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504
USA
koenig-ries@acm.org

22nd December 1999

## Abstract

Mediators configured by rules allow access to heterogeneous information sources. However, the acquisition (and maintenance) of the rules is left to the user. In this paper, we present an integration method to support the specification of mediators. During this method the semantic descriptions of each source, i.e. their application ontologies, are acquired. A three–step approach is proposed: First, the concepts used in different sources are annotated with labels describing their semantics. A global terminology provides the primitive terms of a domain which are combined with operators known from description logics in order to build complex labels. Second, concepts from different information sources are compared if and how they semantically correspond. In the third step, based on the semantic correspondence determined in the second step, mediator rules can be generated. Furthermore, the integration method can be supported by a number of assistants, which semi–automatically integrate the sources.

## 1 Introduction and Overview

As more and more information sources are connected to networks, most notably the Internet, the desire to gain transparent and integrated access to these information sources has evolved. This desire has triggered the development of mediator architectures [24]. A mediator (e.g. [10, 25, 21, 26]) is a component that is able to overcome semantic heterogeneity by translating between different conceptualizations of the universe of discourse. Mediators may be specified by rules (e.g. [18]).

But the finding of mediator specifications is a difficult and labor-intensive task that is left to the user without sufficient support by the system. Furthermore, for an appropriate support of an integration method by a system an explicit semantic description of each source is required. Such integration methods rely on a common domain model (i.e. an ontology [9]) onto which the different conceptualizations are mapped (cf. [1, 15]). But the two major problems that ontology–based integration methods encounter are: First, the development of such a domain model from scratch is expensive and difficult. Second, the scalability of such a fixed and static common domain model is low, because the kind of information sources which can be integrated in the future is limited.

In this paper, we present an approach to supporting the specification of mediators. The integration method needs a semantic description (i.e. an *application ontology* [22]) for each information source. In contrast to other ontology–based approaches,

[*]This work has been partially supported by the German Ministry for Education, Research and Technology under the grant number 01 IS 606 B.

application ontologies are not predefined but are generated and maintained *during* the integration process. The dynamic character of the application ontology allows the revision and extension with respect to newly added sources. A benefit of the method is that it can start with a very simple domain terminology which gradually gets refined during the verification and annotation process. This does not only reduce the start-up costs but also allows to capture very subtle differences in the underlying information sources. Furthermore, application ontologies do not need a global agreement of all sources, which also simplifies their development. In order to make the application ontologies comparable, they use a common global vocabulary organized in a terminology.

The integration method itself is based on a three–step approach: Assuming that we want to build a mediator that translates queries posed in terms of one information source (e.g. the one a user is working with) into terms used by other information sources, the first step is to acquire the semantics of each source. Each information source is described separately and without the consideration of other sources. The relationships to other sources are considered in the next step. For the semantic description, each concept from different information sources is labeled with a term from a common domain terminology acquiring implicitly its application ontology. In the second step, concepts from different sources are semantically compared. Two concepts are related with so–called semantic inter–correspondences estimating their semantic difference. Note that these two kinds of semantics, the domain semantics and the semantic difference, correlate. Because they are acquired separately in both steps, they can be verified against each other. In the third step, mediator rules are derived from the semantic inter–correspondences established in Step two.

Our method does not only guide the user through the different stages of the integration process, but also offers the help of software assistants in all three steps. For instance, assistants can suggest possible labels for concepts and can compare equally labeled concepts. The two assistants which will be presented here use different techniques like case–based reasoning for the label determination and abduction for the detection of the semantic inter–correspondences.

In the following section, we present the characteristics of an mediator–based integration and compare existing integration approaches with these requirements. In the third section we describe our method in more detail including the development of the domain terminology and some assistants. In the last section we give a short summary.

## 2 Requirements and State–of–the–Art

A number of approaches in different research areas was developed for the integration of several heterogeneous information sources (see for an overview e.g. [4, 8]). But most do not satisfy all requirements of a mediator–based integration, i.e. the easy integration and removing of complete information sources (e.g. depending on their availability), selecting the desired information from a large set of heterogeneous sources and an easy "ad-hoc" integration task executed by the user [24].

A rule–based mediator [18, 6] provides a sufficient base for the desired flexibility in the set of integrated information and the "ad-hoc" integration. A rule head contains a query against the conceptualization of one information system and the rule tail describes an equivalent query against another conceptualization. According to the modification in the set of sources the set of rules has to be adapted. But for the rules an adequate acquisition method (i.e. integration method) is required which corresponds to their flexibility. Moreover, for the selection of the desired information from a large set of sources the semantics must be considered. An explicitly represented semantic description supports both the selection of the information and the acquisition of the rules.

Some integration methods are specialized for the rule–based mediator specification. König–Ries [13] presents an approach for the semi–automated generation of mediator specifications (i.e. rules for the TSIMMIS mediator [18]). The information sources are described by graphs. Through comparisons of the graphs, similarities between the sources are estimated. But the comparison is based more on the syntax than on the semantics. Furthermore, König–Ries provides no clear strategy for the integration process. Milo [17] also describes information sources by graphs and provides support for the comparison of objects (i.e. graphs). This approach also lacks a clear strategy in guiding a user and the comparison is reduced to the syntax.

Another group of integration approaches considers an explicit specification of the semantics. In [1] a global domain ontology describes the semantics of the domain. Another example is presented in [7] where the knowledge base CYC [14] is used as global domain model. In the global domain model of these approaches all terms of a domain are arranged in a complex structure. Each information source is related to the terms of the global ontology (e.g. with articulation axioms [7]). But a complex structure like an ontology can imply frequent and expensive modifications if e.g. a further information source has to be integrated. These expensive modifications obstruct the desired flexibility and the "ad-hoc" integration requirement of the mediator.

In OBSERVER [15] it is assumed, that a predefined ontology exists for each information source. Consequently, new information sources can easily be added and removed. But the comparison of the heterogeneous ontologies leads to many homonym, synonym, etc problems, because the ontologies use their own vocabulary.

Another drawback of current ontology–based integration methods is that they assume predefined ontologies. But in reality a ontology must often be acquired from scratch before the integration task can be started. Furthermore, the development of a global ontology is very costly because of the required global agreement among all information sources.

The integration approach presented here is characterized by the inclusion of the development of the semantic source descriptions. In order to provide the flexibility required, each information source is described by its own application ontology but with a common vocabulary globally available avoiding the homonym and synonym problems.

## 3 MESA — Mediator Specification Assistant

### 3.1 Step one: Acquiring source semantics

An information source (IS) can be represented on three layers (cf. [12]). The first layer consists of the data and the syntactical structure of the IS. On the third layer, primitive terms of the domain are provided. The primitive terms are the vocabulary which is used to describe the semantics of the concepts of the IS. The second layer contains the syntactic and semantic description of the IS, i.e. the application ontology. It combines structural information from the first layer and semantic terms from the terminology in the third layer into concepts; the primitive terms of the terminology are annotated as *labels* to the concepts of the sources. Figure 1 gives an overview.

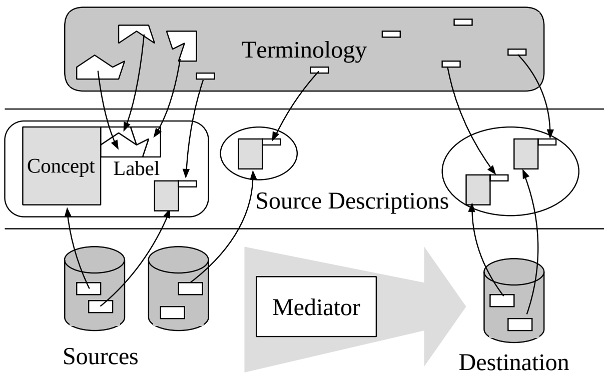

Figure 1: The three levels

The structure of the description is as follows: Each information item is an instance of a concept. Each concept consists at least of a name, a label, a type and a descriptor for the IS. Atomic concepts are represented as:

```
<name,label,type>@source
```

Complex concepts are similar, but additionally include the list of the attribute–names and their concept–descriptions:

```
<name,label,'complex',
   {attribute->concept,...}>@source
```

The terminology consists of a small set of primitive terms only. These terms are arranged in a subsumption taxonomy. Further mechanisms, however, e.g. for aggregating primitive terms, are not supported on the terminology layer (but on the second layer). The absence of more combination operators simplifies the development and maintenance of the terminology, because a user is freed from the decision of how the terms of the terminology should be arranged. As opposed to an ontology, no global agreement is needed about the structure of the terminology.

More powerful combination operators are provided for the construction of the labels on the second layer. A label specifies the semantics of a concept. Like other meta-data approaches (see e.g. [20, 11]) a label can consist of a primitive term from the terminology. But in contrast to existing approaches a label can also be composed from the primitive terms using combination operators in order to build more complex labels. With complex labels a complex semantics can be attached to the concepts.

Because of the composition character of the semantics, the primitive terms in the terminology can be reduced to a very small set consisting only of the essential basic terms of a domain. The terminology provides the vocabulary for the semantic description, but the labels combine the terms according to the structure and needs of an information source. The labels define the application ontology of an IS.

For the construction of the labels the following five operations are supported: conjunction $l_1 \sqcap l_2$, disjunction $l_1 \sqcup l_2$, negation $\neg l_1$, specialization $l_1 \rhd l_2$, and aggregation $l_1 \oplus l_2$, where $l_1, l_2$ are labels (see also the leftmost column of Table 1). The semantics of the operators is specified through a projection $\Pi$ to a variant of a description logic (see e.g. [3, 5, 2]). The description logic (DL) is restricted to concepts with only attributes (1:1 roles) and contains the following construction operators $C \sqcap D$, $C \sqcup D$, $\neg C$, and $\exists f.C$, where $C, D$ are concept terms and $f$ is the name of an attribute. Valid terminological axioms in the $\mathcal{T}$-Box are $C := D$ and $C \sqsubset D$. Then the projection function $\Pi$ is recursively defined: For all simple labels consisting of a primitive term from the terminology there exists a corresponding primitive concept term in the DL. All complex labels are projected according to the third column of table 1.

The terminology is implicitly projected by the function $\Pi$. The projection function can easily be extended for concepts. The result of extended function $\Pi^*$ is the pure semantic description of a concept; the complete set of concepts of an IS constitutes its application ontology in description logic statements. $\Pi^*$ directly describes a basic concept $O = \langle n, L(O), t \rangle @ s$ by its label $\Pi^*(O) = n := \Pi(L(O))$. A complex concept is a correlation of its label with the labels of the concepts related by attributes: let $O = \langle n, L(O), \textit{'complex'}, \{a_1 \rightarrow O_1, ..., a_m \rightarrow O_m\} \rangle @ s$ be a complex concept. Then its corresponding axiom in the application ontology is $\Pi^*(O) = \Pi(L(O)) := \exists a_1.\Pi(L(O_1)) \sqcap ... \sqcap \exists a_m.\Pi(L(O_m))$.

Moreover, with the help of the projection $\Pi$ labels can be compared and verified. Useful tests are for example: a label $L(O)$ of a concept $O$ is valid ($:\Leftrightarrow \Pi(L(O))$ *is valid*), the label of concept $O_1$ subsumes the label of concept $O_2$ (i.e. $L(O_2) \sqsubseteq L(O_1) :\Leftrightarrow \Pi(L(O_2)) \sqsubseteq \Pi(L(O_1))$), and two labels are equivalent (i.e. $L(O_1) = L(O_2) :\Leftrightarrow \Pi(L(O_2)) = \Pi(L(O_1))$).

| Complex Label ($l$) | ASCII | $\Pi(l) \mapsto$ Concept Term | Interpretation $(\Pi(l))^I$ |
|---|---|---|---|
| $l_1 \sqcap l_2$ | (AND $l_1$ $l_2$) | $l_1 \sqcap l_2$ | $l_1^I \cap l_2^I$ |
| $l_1 \sqcup l_2$ | (OR $l_1$ $l_2$) | $l_1 \sqcup l_2$ | $l_1^I \cup l_2^I$ |
| $\neg l_1$ | (NOT $l_1$) | $\neg l_1$ | $U \setminus l_1^I$ |
| $l_1 \rhd l_2$ | ($l_1$ OF $l_2$) | $l_1 \sqcap \exists to.l_2$ | $l_1^I \cap \{k^I \mid \exists k^I : (k^I, l_2^I) \in to^I\}$ |
| $l_1 \oplus l_2$ | (COMP $l_1$ $l_2$) | $\exists \textit{has-part}.l_1 \sqcap \exists \textit{has-part}.l_2$ | $\{k_1^I \mid \exists k_1^I : (k_1^I, l_1^I) \in \textit{has-part}^I\} \cap$ $\{k_2^I \mid \exists k_2^I : (k_2^I, l_2^I) \in \textit{has-part}^I\}$ |

Table 1: Label terms and their corresponding terms in description logics

The following simple example from the stock–market domain illustrates how ISs are represented within the three layers. Three ISs are described: *New_York* and *Frankfurt*, holding information about the stock–markets of New York and Frankfurt, respectively. The third information–source *Broker* represents a London broker, retrieving information from *New_York* and *Frankfurt* and mediating between them.

Both stock markets store the information in a (relational) database. In *New_York*, each security is stored by a ticker[1], its name, its kind of investment (i.e. `'S'` for stock and `'B'` for bond), a date, and its price. In *Frankfurt* the information is distributed among two tables. The first table stores only the *Wertpapierkennnummer (WKN)*, which is semantically similar to the ticker but represented as an integer, and the name of the security. In the second table, the date and price of a security are stored. Both tables are connected by the WKN. So, in *Frankfurt* nearly the same data as in *New_York* is represented.

Figure 2 provides an example for concepts in the information source *Frankfurt*. The data is distributed over two complex concepts that contain a common identifier which links both objects, equal to the key that is used in the data source itself. Since we haven't assigned any semantics to the concepts at this point, all labels are @nil[2]. Our domain terminology consists of the following primitive terms that are taken from the domain *stock–markets*[3]: `NAME`, `VALUE`, `DATE`, `KEY`, `KIND`, and `SECURITY`. With those primitive terms and the operators we can compose complex semantic information, for example a name for an investment (`NAME OF SECURITY`) or a price of an investment (`VALUE OF SECURITY`).

In the first step of the integration approach the concepts in all ISs must be labeled. Consider the `name` concept of the Frankfurt IS. The best semantic description our domain terminology offers is a specification of the primitive term `NAME` with respect to the investment `SECURITY` resulting in the label (`NAME OF SECURITY`). This label is attached to the `name` concept of the Frankfurt information source. Figure 3 shows the complete labeling for all information sources.

Up to now, we haven't dealt with the problem how to find correct labels for concepts. But finding correct labels requires profound knowledge of both the information source and the domain terminology. Normally a user does not have such knowledge and needs support in this task. A number of software assistants must help the user.

In general, these assistants do a heuristic classification task. They have to infer from the semantics of a concept from its syntax (i.e. its name and its type and — in case of a complex concept — its attribute structure). Of course, this inference is absolutely impossible in general. But an assistant can implement some heuristics and can suggest the user hypotheses. Then the user can de-

[1]an alphanumeric identifier for stocks

[2]Unassigned

[3]Only a subset of the actual terminology is shown as an example

```
<frankfurt-market, @nil, complex,
    { wkn -> <wkn, @nil, integer>@frankfurt,
      name -> <name, @nil, string>@frankfurt}>@frankfurt,
<frankfurt-prices, @nil, complex,
    { wkn -> <wkn, @nil, integer>@frankfurt,
      price -> <price, @nil, integer>@frankfurt,
      date -> <date, @nil, string>@frankfurt}>@frankfurt
```

Figure 2: Concepts in the IS *frankfurt*

```
<New_York-market, (SECURITY), complex,
    { tckr -> <ticker, (KEY OF SECURITY), string>@New_York,
      nm -> <name, (NAME OF SECURITY), string>@New_York,
      knd -> <kind, (SECURITY), char, {'S', 'B'}>@New_York,
      dt -> <date, (DATE), string>@New_York,
      prc -> <price, (VALUE OF SECURITY), real>@New_York
    }>@New_York

<Frankfurt-market, (SECURITY), complex,
    { wkn -> <wkn, (KEY OF SECURITY), integer>@frankfurt,
      name -> <name, (NAME OF SECURITY), string>@frankfurt
    }>@frankfurt,
<Frankfurt-prices, (SECURITY), complex,
    { wkn -> <wkn, (KEY OF SECURITY), integer>@frankfurt,
      price -> <price, (VALUE OF SECURITY), integer>@frankfurt,
      date -> <date, (DATE), string>@frankfurt
    }>@frankfurt

<Broker-market, (SECURITY), complex,
    { name -> <name, (NAME OF SECURITY), string>@broker,
      price -> <price, (VALUE OF SECURITY), integer>@broker,
      date -> <date, (DATE), string>@broker,
    }>@broker
```

Figure 3: Labeled concepts of all information sources

cide if he wants to accept a hypothesis or wants to reject it.

For example, a heuristic can try to conclude from the semantics of a concept from its name (and type). Obviously, an approach which uses only this kind of heuristic does not promise good results. Another assistant can use a knowledge base which contains knowledge about labeling of concepts. But who acquires and maintains such a knowledge base?

The most promising approach seems to be assistants which use case-based reasoning techniques because different heuristics can be realized and easily combined. In case-based reasoning the concept a label is to be determined for is compared with a case-base of concepts. The case-base contains concepts with labels already assigned. Based on similarity measures, these assistants decide which of the cases is the most similar to the concept at hand and then proposes the usage of that case's label to the user. The different heuristics can be implemented in different similarity measures. Normally, the different similarity measures are first weighted, then added and last normalized.

Once the user has decided on a label for the concept, this concept may be added to the case–base. Through this adding the assistants improve their skills — they learn the labeling.

Given a concept $O_1$ and a case-base of already labeled concepts, the goal is to find the case $C_i$ that matches $O_1$ best to determine the label of $O_1$.

The first assistant is based on a comparison of the names of concept $O_1$ and case $C_i$. A semantic net (in our case WordNet [16]) is used that arranges terms as nodes in a directed graph which are connected via different edges representing synonym, hypernym (generalization) and meronym (part-of) relationships. For any two terms, their distance, i.e. the length of the shortest path along a certain kind of edge connecting them, can be determined. The shorter this path, the more similar are the two terms compared. In order to be able to compare names, it may be necessary to decompose them into their subparts as indicated by the occurrence of hyphens or underscores in the names. Each part of each name can then be compared with each part of the other and the results can be appropriately combined. Consider as an example the `Frankfurt-market` concept and a case-base containing the `NewYork-market` concept. Both concept names can be decomposed in two subparts. The comparison of these subparts determines that `market` and `market` are identical, i.e. their distance is 0, and that both `New York` and `Frankfurt` are children of the node `city`, their distance being 2. From this, the degree of similarity can be computed[4].

The second and third assistant try to determine whether concept $O_1$ and case $C_i$ possess similar attributes. Both compare the concept descriptions of the attributes. Assistant 2 can be used with at least some of these concepts that have been labeled already. In that case, it determines for each labeled concept of an attribute of $O_1$ whether $C_i$ possesses an attribute with the same or a similar label. Obviously, the better the match for each attribute and the higher the number of attributes for which a match can be found, the more similar are the concepts. Again, these comparisons can be quantified. At the beginning of the integration process, only few labeled concepts will exist, thus, Assistant 3 offers the possibility to compare the names of attribute concepts instead of their labels. The execution of this comparison is similar to the one described above.

[4]The exact formula depends on the depth of the hypernym hierarchy of the tool used.

## 3.2 Step two: Finding Semantic Inter-Correspondences

In the second step of our approach, we try to identify semantic inter–correspondences (SIC) between concepts of different ISs (cf. [19]). SICs are the representation of the semantic relationship between two concepts from different ISs. They are the means to describe the semantic differences between the concepts as described above. Before defining a classification of SICs, we first have a more detailed look at the concepts between which they are defined.

A concept defined in an IS represents a set of all possible instances in the database. Each instance of the concept represents a real-world object. Since the SICs describe the semantic difference between two concepts, we focus on the real-world objects which provide the semantic basis for the concepts, in the following referred to as *extensions* of the concept.

We classify four kinds of SIC's between concepts $O_1$ and $O_2$ (cf. [19]):

- *Semantic equivalence*: $O_1 \equiv O_2$. The extensions of $O_1$ and $O_2$ are the same – the concepts therefore represent the same real-world objects.
- *Semantic subsumption*: $O_1 \subset O_2$. The extensions of $O_1$ are a subset of the extensions of $O_2$.
- *Semantic intersection*: $O_1 \cap O_2$. The extensions of $O_1$ and $O_2$ overlap partially.
- *Semantic incompatibility*: $O_1 \neq O_2$. The extensions of $O_1$ and $O_2$ are semantically incompatible, they are representing different real-world objects.

These four SIC-types are sufficient since they cover all useful relations between two concepts. Note that an extension of a concept can only be estimated and described by a user through e.g. analyzing the data in the different information sources.

The labels described in section 3.1 and the SICs are in a close relationship to each other since labels can be verified by the SICs:

1. If two concepts are semantically equivalent (i.e. $O_1 \equiv O_2$), then their labels have to be identical (i.e. $L(O_1) = L(O_2)$).

2. If a concept $O_2$ subsumes a concept $O_1$ semantically (i.e. $O_1 \subset O_2$) then the label of concept $O_1$ is subsumed by the label of $O_2$ (i.e. $L(O_1) \sqsubset L(O_2)$) .

3. If two concepts intersect semantically (i.e. $O_1 \cap O_2$) then their labels overlap partially (i.e. $L(O_1)$ AND $L(O_2)$ is valid).

4. If two concepts are semantically incompatible (i.e. $O_1 \neq O_2$) then their labels are in no relation (i.e. $L(O_1)$ AND $L(O_2)$ is not valid).

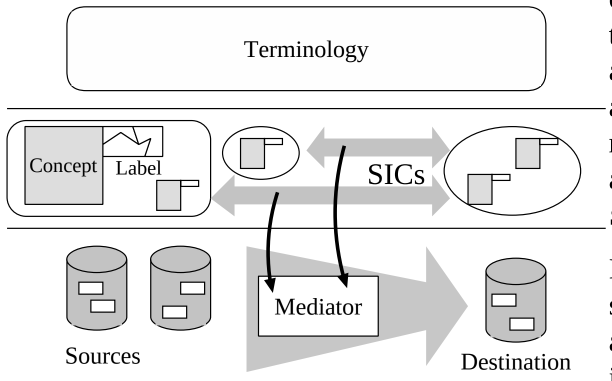

Figure 4: Semantic Inter-Correspondences

The user establishes SICs considering the extensions of the concepts instead of considering the labels. If the SICs are established there might be differences between the SICs and the labels of the concepts. Assuming that a previously defined SIC is correct, the label of one of the regarded concepts is not precise enough and needs to be refined. This is done by choosing a primitive term from the terminology or creating a new primitive term which is added to the terminology. The new term could either replace the old label of the concept or could be combined with one of the given operators `AND`, `OR` and `NOT`.

In the previously used example, the concept `<Broker-market>` is used to extract the information from the concepts `<New_York-market>` and `<Frankfurt-market>`, and offers an integrated view on their data. The extensions of the concept `<New_York-market>` cover stocks and bonds, and the concept has been labeled as `(SECURITY)`. Assume, that the extensions of the concept `<Broker-market>` cover stocks only, while the concept has been also labeled with the same label `(SECURITY)`. The user, taking only the extensions of the concepts into consideration in order to establish a SIC (and not their labels), would create a *semantic subsumption* correspondence (`<Broker-market>` $\subset$ `<New_York-market>`). But *semantic subsumption* implies subsuming labels and not identical labels. As a result, the label `(SECURITY)` for the concept `<Broker-market>` is identified as insufficiently defined and requires refinement: a new primitive term `STOCKS` is created which replaces the old label. The primitive term will be added to the terminology as a specialization of `SECURITY`.

In this step of the integration method software assistants could also provide help for the user. There are two ways how SICs could be automatically identified:

- The first assistant relies on an abduction principle. It uses the implication between SICs and labels in the "inverse direction, i.e. abducting SICs from the labels. The assistant compares the label of each concept in one information source with the labels of the concepts in the other information source. If the equivalent labels are found, these two concepts may be considered semantically equivalent and the *semantic-equivalence* SIC will be established.

- If labels are not available, the similarity of the concepts as discussed in the previous section could be taken into consideration by an assistant. The similarity measure between two concepts would need to exceed a certain threshold-value to consider the concepts semantically equivalent. This way, a *semantic-equivalence* SIC could be established.

Note, that the assistants make suggestions only. All the SICs found this way have to be validated by the user!

### 3.3 Step three: Generating Mediator Specifications

In the third step, the rules for the mediator are specified. These rules describe how queries posed against one conceptualization, specified by a concept description in the rule head, can be transformed into equivalent queries against the other conceptualizations, specified by a set of concepts in the rule tail. We represent rules in a PROLOG formalism: $O_h$:-$O_{t_1}, O_{t_2}, ..., O_{t_n}$ where $O_h, O_{t_1}, O_{t_2}, ..., O_{t_n}$ are concept definitions with variables as place–holders. For a detailed definition please refer to [23].

The semantic equivalence and semantic subsumption SIC can be used for the generation of rules. The semantic equivalence $O_1 \equiv O_2$ between the objects $O_1$ and $O_2$ can be represented by the two rules $O_1$:-$O_2$ and $O_2$:-$O_1$; the semantic subsumption $O_1 \subset O_2$ by the rule $O_2$:-$O_1$. The transformation of SICs into rules is also supported by a software assistant.

## 4 Conclusion

Mediator specification and terminology development are two labor-intensive tasks that are serious obstacles to the widespread use of mediator-based systems. In this paper, we have proposed a three–step approach to acquire mediator specifications and to gradually build a domain terminology at the same time. The approach also allows the support of software assistants, which help a user to acquire and verify the semantics of information sources.

We have implemented a first prototype. The integration approach (without the assistants) is currently being evaluated by some users. Furthermore, we started the delopment of a second prototyp which will include the couple of assistants mentioned in this article.

## References

[1] Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the sims information mediator. In *Advanced Planning Technology*, California, USA, 1996. AAAI Press.

[2] F. Baader. Logic-based knowledge representation. In M.J. Wooldridge and M. Veloso, editors, *Artificial Intelligence Today – Recent Trends and Developments*, volume 1600 of *Lecture Notes in Artificial Intelligence*, pages 13–41. Springer Verlag, 1999.

[3] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In *Proceedings of the First International Workshop on Processing Declarative Knowledge*, volume 572 of *Lecture Notes in Computer Science*, pages 67–85, Kaiserslautern (Germany), 1991. Springer–Verlag.

[4] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *Computing Surveys*, 18(4):323–364, 1986.

[5] A. Borgida and P.F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[6] Sophie Cluet, Claude Delobel, Jerome Simeon, and Katarzyna Smaga. Your mediators need data conversion! In Laura Haas and Ashutosh Tiwary,

editors, *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD 98)*, SIGMOD Record, Seattle, Washington, USA, June 1998.

[7] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration using a large knowledge base in carnot. *IEEE Computer*, 24(12):55–62, December 1991.

[8] S. Conrad. *Föderierte Datenbanksysteme: Konzepte der Datenintegration*. PhD thesis, Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg, 1996.

[9] T. Gruber. A translation apporach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[10] Joachim Hammer, Hector Garcia-Molina, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. Information translation, mediation, and mosaic-based browsing in the tsimmis system. June 1995.

[11] V. Kashyap and A. Sheth. Schematic and semantic semilarities between database objects: A context-based approach. Number TR-CS-95-001, LSDIS Lab, Univ. of GA, 1995.

[12] Vipul Kashyap and Amit Sheth. *Cooperative Information Systems: Current Trends and Directions*, chapter Semantic Heterogeneity in Global Information Systems: The role of Metadata, Context and Ontologies. Academic Press, 1997.

[13] Birgitta König-Ries. *Ein Verfahren zur semiautomatischen Generierung von Mediatorspezifikationen*, volume 55 of *DISDBIS*. infix Verlag, 1999.

[14] D. B. Lenat, M. Prakash, and M. Sheperd. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *The AI Magazine*, 6(5):65–85, Winter 1986.

[15] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Proceedings First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)*, pages 14–25, Brussels, Belgium, June 1996. IEEE Computer Society Press.

[16] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to wordnet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990.

[17] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In *Proc. of 24th VLDB*, New York City, USA, 1998.

[18] Yannis Papakonstantinou, Hector Garcia-Molina, and Jeffrey Ullman. Medmaker: A mediation system based on declarative specifications. In *International Conference on Data Engineering*, pages 132–141, New Orleans, February 1996.

[19] Christine Parent and Stefano Spaccapietra. Issues and approaches of database integration. *Communications of the ACM*, 41(5):166–178, May 1998.

[20] Michael Siegel and Stuart E. Madnick. A metadata approach to resolving semantic conflicts. In *Proceedings of the 17th International Conference on Very Large Data Bases*, Cambridge, MA, 1991.

[21] V. S. Subrahmanian, Sibel Adali, Anne Brink, Ross Emery, James J. Lu, Adil Rajput, Timothy J. Rogers, Robert Ross, and Charles Ward. Hermes: A heterogeneous reasoning and mediator system. Technical report, University of Maryland, 1995.

[22] G. van Heijst, A.T. Schreiber, and B.J. Wielinga. Using explicit ontologies for kbs development. *International Journal of Human-Computer Studies*, 46(2/3):183–292, 1997.

[23] Holger Wache. Towards rule–based context transformation in mediators. *Proceedings of the international Workshop on Engineering Federated Information Systems (EFIS 99)*, 1999.

[24] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992.

[25] Ling Ling Yan, M. Tamer Özsu, and Ling Liu. Accessing heterogeneous data through homogenization and integration mediators. In *Proc. of CoopIS 97*, Kiawah Island, USA, 1997.

[26] Gang Zhou, Richard Hull, and Roger King. *Intelligent Integration of Information*, chapter Generation Data Integration Mediators that Use Materialization, pages 111–134. Kluwer Academic Publishers, 1996.