Towards Rule–Based Context Transformation in Mediators

Holger Wache
University of Bremen
Universitätsallee 21-23
D–28359 Bremen, Germany
e-mail: wache@informatik.uni-bremen.de

Abstract

Mediators are used for the integration of heterogeneous information sources. Rule–based mediators map the sources to an integrated view by transformation rules reconciling structural heterogeneity problems. But the reconciliation of semantic heterogeneity problems play a subordinate role. In this paper, we present an approach that extends rule–based mediators for the reconciliation of semantic heterogeneity problems.

The discovery of the semantic heterogeneity problems requires an explicit representation also indicating the semantics of an information. In this paper, the semantic level of an information item is described by contexts. The transformation of several information sources into the integrated view includes the transformation into the context of the integrated view — the context transformation. The set of transformation rules is extended with so–called context rules for the context transformation. Context rules need an extension of the representation formalism and a new inference mechanism provided by the mediator. The declarative representation of context rules allows to propagate the context transformation into the subqueries sent to the heterogeneous information sources and provides potential for optimization.

1 Introduction and Overview

Mediators [15] provide a flexible integration of several information systems such as database management systems, geographical information systems, or even the world wide web. Then a mediator combines, integrates, and abstracts the information provided by the sources [15]. Normally the sources are encapsulated by wrappers. Wrappers hide the internal data structure model of a source and transform the contents to a uniform data structure model [14].

In the last years a lot of different mediators were developed (see e.g. [1, 2, 3, 7]). A popular exponent is the rule–driven TSIMMIS mediator [7]. The rules describe how information of the sources can be mapped to the integrated view. In a simple case, a rule converts the information of the sources into information of the integrated view. The mediator uses the rules to split the query, which is formulated with respect to the integrated view, into several sub–queries for each source and combine the results returned by the sources according to query plan.

A mediator has to solve the same problems which are discussed in the federated database research area, i.e. *structural heterogeneity* (schematic heterogeneity) and *semantic heterogeneity* (data heterogeneity). Structural heterogeneity means that different information systems store their data in different structures. Semantic heterogeneity considers the content and its semantics of an information item. In rule–based mediators, rules are mainly designed in order to reconcile structural heterogeneity. Discovering semantic heterogeneity problems and their reconciliation play a subordinate role. But for the reconciliation of the semantic heterogeneity problems, the semantical level also has to be considered [8, 5]. *Contexts* are one possibility to describe the semantical level. A context contains "meta data relating to its meaning, properties (such as its source, quality, and precision), and organization" [12]. A value has to be considered in its context and may be transformed into another context (so–called *context transformation*).

In this paper we propose a rule–based representation formalism for both integration and context transformation. The formalism bases on an object model, which is enriched by contexts. In the formalism two kinds of rules can be distinguished: *combination rules* and *replacement rules*. Combination rules gather the information of the sources. Replacement rules replace an information by another information. Replacement rules are needed for the context transformation. We will see that they go beyond traditional rule–based programming paradigms.

In the formalism, the rules for the integration and the context transformation are represented. The rules for the integration (*integration rules*) are similar to other rule–based mediators. But the context transformation with the reconciliation of the semantic heterogeneity problems is also specified by rules (*context rules*). Context rules implement the context transformation in a declarative and expressive representation formalism. Context and integration rules together compose the set of *transformation rules* which configure the rule–based mediator.

A context rule (and integration rule) is represented by a replacement rule or combination rule. The different view on the rule sets is pragmatic: context rules and integration rules are seen on the application level (i.e. what is the task of the rules), whereas the combination and replacement rules are seen on the representation level (i.e. how the rules are represented).

The distinction between context and integration rules helps to concentrate on the specific heterogeneity problem. While formulating integration rules the aspect of semantic heterogeneity problems can be discarded; the context transformation will be done by the separate set of context rules. In our opinion the separation between context rules and integration rules are more declarative and adequate than their combination into one kind of rule. The explicit consideration of a context and its transformation with context rules helps to reduce errors and frees application programmers from being concerned with conversions [12].

Furthermore we sketch a rule–based mediator which can interpret both kinds of transformation rules. During the inference process, where the query is divided into sub-queries for each information source, the context transformation and the integration are interleaved.

This paper is organized as follows: In the next section we define transformation rules (i.e. combination rules and replacement rules) for context transformation and integration. The expressiveness of the transformation rules is demonstrated in section 2.4 by example. In section 3 we briefly outline the mediator and explain how queries are divided into several sub-queries sent to the information sources with a focus on the context transformation. Our approach is compared with the existing ones in section 4. We conclude in the last section and show future directions.

2 Transformation Rules

Transformation rules guide the mapping of the information sources to the integrated view provided by the mediator. The building–blocks for transformation rules are *templates*, which we introduce at first.

2.1 Templates

Templates are a generic, object–centered description of information. We see information as instances of these templates. For the handling of semi–structured data, we omit a schema definition. Instead, we assume that each information (resp. template) describes itself. A template has a *name*, a *context* addressing the semantics of the concept (see below), a *type* determining the data type, the *value* containing the value, and the last identifier *source* denoting which source the template belongs to. The value can be a simple value, e.g. a number, or a string, or a list of attributes. An *attribute* consists of a name and a template the attribute refers to. The type is `complex` if the value is a list of attributes. Otherwise the type denotes the basic data type of the value, e.g. `string` for the value "Hello World". In the following templates are written as:

$$T = \langle name, context, type, value \rangle @source$$

The difference between templates and their instances is that in templates variables are allowed on each place[1]. For example, the value or the name can be replaced by a variable. The template represents all those instances which can be unified[2] with the template.

Type and value describe the structure of an information while the context captures the semantics of the information. Contexts are designated to make semantic conflicts obvious and to guide the reconciliation of semantic heterogeneity problems. In terms of Kashyap and Sheth [9] a context represents content–descriptive information. A context is represented as set of meta-attributes. A meta-attribute consists of its name and a value. The meta attributes and their values can stem from a common domain ontology [6] ensuring a global semantics over all information source. We will write context as follows:

$$C = \langle (C_1, V_1), (C_2, V_2), ...(C_n, V_n) \rangle$$

[1] with the exception of attribute and meta–attribute in contexts (see below)!

[2] in the sense of logic programming

```
<nyse-stocks, <>, complex,
    { tckr -> <ticker, <(format,nasqaq)>, string, _>@nyse,
      nm -> <name, <>, string, _>@nyse,
      prc -> <price, <(currency, dollar), (scale,0.01)>, real, _>@nyse,
      dt -> <date, <(format,american)>, string, _>@nyse,
    }>@nyse

<franfurt-stocks, <>, complex,
    { wkn -> <wkn, <(format,wkn)>, integer, _>@frankfurt,
      name -> <name, <>, string, _>@frankfurt
    }>@frankfurt,
<frankfurt-prices, <>, complex,
    { wkn -> <wkn, <(format,wkn)>, integer, _>@frankfurt,
      price -> <price, <(currency, DM), (scale,1)>, integer, _>@frankfurt,
      date -> <date, <(format,german)>, string, _>@frankfurt
    }>@frankfurt

<broker-stocks, <>, complex,
    { name -> <name, <>, string, _>@broker,
      price -> <price, <(currency, pound), (scale,1)>, integer, _>@broker,
      date -> <date, <(format,american)>, string, _>@broker,
      place -> <place, <>, set,
                     { <place-german, <>, string, "frankfurt">@broker,
                       <place-america, <>, string, "wallstreet">@broker
                     }>@broker
    }>@broker
```

Figure 1: Description of the information sources and of the broker's view of the information sources

Templates with attributes can represent tables (relations) in a relational data structure model. The attributes of the template are the attributes of the relation. The value of the template attributes are templates encapsulating the basic data types of the relation attributes.

An example from a financial application domain illustrates how information sources can be described in the proposed representation formalism. Suppose a broker sitting in London wants to have an integrated view of the stock market in New York, USA, and Frankfurt, Germany. Both stock markets provide a (relational) database storing the requested information. In New York, each stock is stored by a ticker, its name, a date, and the price of the stock.[3] In Frankfurt, the information is distributed over two tables. The first table stores only the Wertpapierkennummer (WKN), which is semantically similar to the ticker but represented as integer, and the name of the stock. In the second table the date and price of a stock is stored. Both tables are connected by the WKN. The broker also has to describe the integrated view how he wants to see the stock markets. The figure 1 illustrates three templates describing the sources and the integrated view.

[3]The sign '_' means a don't–care variable.

In the example also the contexts of the information sources are given. We focus on the context of the template `price`. The context contains the meta-attributes `currency` and `scale`. In New York, the prices are stored in `Dollar` with scaling factor `0.01`, whereas in Frankfurt the prices are stored with currency `DM` and scaling factor `1`. The broker wants to see the prices with the currency `pound` including the scaling factor `1`.

2.2 Definition of Combination and Replacement Rules

In this section we explain the *representation* of transformation rules. Two kinds of rules can be distinguished: *combination rules* and *replacement rules*. In section 2.3 we discuss how context transformation and integration can be represented with these two kinds of transformation rules.

In general, every transformation rule consists of a head and a body. The body contains a set of templates; whereas the head is built up by one template. Analogous to most rule formalism, the templates in the bodies are connected by conjunction. Common variables are used for representing interrelationships between the templates.

A combination rule is designed for the combination and integration of information and primarily for the reconciliation of structural heterogeneity problems. The information is collected according to the templates in the body of the rule. Then the new information is constructed with respect to the head of the rule. To be more precise, the propagation of the information pieces is done by the variables in the templates. First, the information items are bound to the variables in the body templates. Then the information items are propagated to the head and are rearranged according to the structure of the head template. A combination rule is defined as follows:

$$H \longleftarrow B_1, ..., B_n, \phi_1, ..., \phi_m$$

H and $B_1, ..., B_n$ are templates; $\phi_1, ..., \phi_m$ are expressions. Expressions are formulated over the variables in the templates and constrain the possible substitution for the variables. For example, the expression $?Price > 100$ restricts the possible values for the variable $?Price$ to be greater than 100.

Combination rules construct new information; replacement rules are needed for the exchange of information. With replacement rules an existing information is transformed into another structure. But in contrast to the combination rule, the information is replaced. Combination rules can only be applied to top–level templates, but replacement rules can also be applied to an information which is nested in the structure of a top–level information e.g. in an attribute. The replacement rule replaces the information in the attribute with the transformed information.

In general, a replacement rule transforms one template into another template. A replacement rule is represented as follows:

$$H \longleftarrow \overline{B} \& B_1, ..., B_n, \phi_1, ..., \phi_m$$

In the replacement rule one template $\overline{B}$ of the body templates is needed to be signed. The signed template replaces the head template H of the rule. The other templates in the body $B_1, ..., B_n$ and the expressions $\phi_1, ..., \phi_m$ are required to support or to restrict the replacement.

If the replacement rules are compared with a logical rule formalism e.g. horn logic, they allow the application to nested predicates. For example, in the goal $p(f(q(X)))$ the predicate $q(X)$ is nested in the predicate p. In terms of the templates, the template p contains the template q in the attribute f. In contrast to other rule formalisms, the replacement rule $q(X) \longleftarrow r(X) \& ...$ is applicable to the goal. The rule replaces $q(X)$ with the signed predicate $r(X)$ resulting in the refined goal $p(f(r(X)))$. The replacement rules build an extension of the traditional rule formalisms. Their inference principle is described in detail in [13].

2.3 Context Transformation and Integration with Transformation Rules

The integration task and the context transformation is represented by the combination and replacement rule. On the application level the set of transformation rules is divided in integration rules and context rules describing to which task a rule will be used.

For the integration, the information sources are mapped to the integrated view provided by the mediator. Normally for each information source there exists a combination rule which maps one source to the integrated view. The combination rule describes an information item of the integrated view in the terms of item of one information source. Other information sources also (re)define the integrated view but in their terms. If one source does not provide enough information which is desired in the integrated view, this source is integrated with other sources providing the missing part. With combination rules the mediator is comparable to a deductive database management system over the heterogeneous information source.

During context transformation, an information is transformed into another context. The formulation of a context rule should not depend on where the information is located, which has to be converted into another context (e.g. nested in an attribute). If combination rules are used, the context transformation is restricted to top-level templates. Replacement rules allow to exchange information even in nested template structures. Therefore replacement rules are the adequate representation formalism for context rules. For simple templates the expressions $\phi_1, ..., \phi_m$ determine how the value of the template has to be converted into the new context.

In general, context rules or integration rules, resp., are represented as replacement rules or combination rules, resp. But it should be noted that some context rules can be described as combination rules and, vice versa, some integration rules can be represented as replacement rules.

```
<broker-stocks, ?Stock-Context, complex,
    { name -> <name, ?Name-Context, string, ?Name>@broker,
      price -> <price, ?Price-Context, integer, ?Price>@broker,
      date -> <date, ?Date-Context, string, ?Date>@broker,
      place -> <place-american, <>, string, "wallstreet">@broker
    }>@broker
<-
<nyse-stocks, ?Stock-Context, complex,
    { nm -> <name, ?Name-Context, string, ?Name>@nyse,
      prc -> <price, ?Price-Context, integer, ?Price>@nyse,
      dt -> <date, ?Date-Context, string, ?Date>@nyse,
    }>@nyse.
```

Figure 2: A simple structure transformation

```
<broker-stocks, ?Stock-Context, complex,
    { name -> <name, ?Name-Context, string, ?Name>@broker,
      price -> <price, ?Price-Context, integer, ?Price>@broker,
      date -> <date, ?Date-Context, string, ?Date>@broker,
      place -> <place-german, <>, string, "frankfurt">@broker
    }>@broker
<-
<franfurt-stocks, ?Stock-Context, complex,
    { wkn -> <wkn, _, integer, ?WKN>@frankfurt,
      name -> <name, ?Name-Context, string, ?Name>@frankfurt
    }>@frankfurt,
<frankfurt-prices, _, complex,
    { wkn -> <wkn, _, integer, ?WKN>@frankfurt,
      price -> <price, ?Price-Context, integer, ?Price>@frankfurt,
      date -> <date, ?Date-Context, string, ?Date>@frankfurt
    }>@frankfurt
```

Figure 3: Integration of information

2.4 Examples for Transformation Rules

We illustrate the transformation rules by continuing the example of section 2.1. We give one combination rule for the mapping of Wall Street to the broker in London and one combination rule for the mapping of Frankfurt to the broker in London. These two rules are shown in the following.

The simplest case in our example is the mapping of Wall Street to London. The template of Wall Street has to be projected to the template of London. The appropriate combination rule is reduced to a pure structural transformation. The rule is given in figure 2.

The name, price, and date can be easily propagated to the template of the broker. Variables starts with a question–mark. The example shows how structural heterogeneity problems can easily be reconciled. The problems in the example reconciled by the rule are attribute name conflicts [10] and table structure conflicts [10], where the attribute `tckr` is deleted and the attribute `place` is deduced. Of course, more structural heterogeneity problems can be reconciled with combination rules.

Note the variables for the context definition. The combination rule does not address a specific context but propagate the context of the source to the target in the head. The result of the application of the rule is an information whose structure is equal to the desired target in the head, but may differ in the context. The task of the context transformation is to keep the structure of an information and to convert the value with respect to the context differences.

If a combination rule contains variables for the context the rule is independent from the context of each information. The variables mean that the rule is applicable in every context. But sometimes the integration can depend on a context because the integration

```
<?Price-Template, <(currency, pound)>, integer, ?Price-Pound>@?Source
<-
<?Price-Template, <(currency, dollar)>, integer, ?Price-Dollar>@?Source &
?Price-Pound = ?Price-Dollar * 1.30.
```

Figure 4: A simple context rule

```
<?Price-Template, <(currency, dollar), (date, ?Date)>, integer, ?Price-Dollar>@?Source
<-
<?Price-Template, <(currency, DM), (date, ?Date)>, integer, ?Price-DM>@?Source &
<price-conversion, <>, complex,
    { from-currency -> <currency, <>, string, "DM">@London,
      to-currency -> <currency, <>, string, "Dollar">@London,
      date -> <date, ?Date-Context, string, ?Date>@London
      con-val -> <conversion-value, <>, real, ?Conversion-Value>@London
    }>@London,
?Price-Pound = ?Price-Dollar * ?Conversion-Value.
```

Figure 5: Context rule with information gathering

is restricted to a specific context.[4] Then the context definition of the head template in a combination rule defines in which contexts the combination rule is applicable.

The next example of a combination rule integrates the information of two different templates (many–to–many table conflicts [10]). In our scenario, both templates are located in one source, but combination rules may also collect information from different sources.

In figure 3 a combination rule is given which maps the stock market of Frankfurt to the broker in London. In the body, two templates requested different information from Frankfurt. The templates are connected through the common variable `?WKN`.

Figure 4 illustrates a simple example of a replacement rule for context transformation. The rule converts a price with the currency unit `dollar` into a price with the currency unit `pound`. The appropriate conversion of the value is specified by the expression $?Price\text{-}Pound = ?Price\text{-}Dollar * 1.30$ in a declarative way.

It is important to remark that the context specification in the head and body only relates to a *part of* a context definition. The rule is applicable to every template whose context contains at least the specified meta-attributes. In the example given in figure 4, the price currency conversion rule is applicable to all information containing the meta-attribute `currency` and value `dollar` in their context — in spite of the other meta-attributes in the context of the template.

[4] Kashyap and Sheth [8] even argue that every integration has to be considered with respect to a context.

The other meta-attributes in the context remain unchanged by the conversion. Other context rules affect other meta-attributes. For example, another replacement rule can focus on the scaling factor in a context. In our example, both rules are needed to completely transform a value to a target context. In general, several rules are applied for a context transformation. The order in which the rules are applied does not affect the result of the context transformation.[5] The splitting of the context transformation into several context rules supports the maintenance and reuse of the rules for context transformation.

The next example in figure 5 shows how a context rule can refer to more than one meta-attribute. The replacement rule addresses all templates whose context includes at least the attributes `currency` and `date` with their appropriate values.

The replacement rule is also an example where additional information from other sources can be used for the conversion. The rule asks the source `London` for the conversion rate between `DM` and `Dollar`. Note, that the date `?Date` comes from the meta-attributes in the context.

It should also be noted that the price conversion above does not directly transform the prices coming from Frankfurt with currency `DM` into a price with currency `pound` which are needed by the broker. The rule above first converts the prices in prices with currency `dollar`. After an additional application of the rule

[5] However, this is not always true. If the order of application is important for a context transformation, this problem can be solved by priorities for the rules. Rules with a higher priority are applied at first.

in figure 4, the prices are represented in the desired currency `pound`. The application of context rules can be nested in order to convert a value from a source context to a target context. Because context transformation rules can be nested, a much smaller set of rules is needed, which also supports the maintenance of the rule set.

3 Querying the Mediator

In this section we will explain how a query can be formulated and sent to the mediator. A detailed overview is given in [13]. It is shown by example, how the mediator decomposed the query to several sub-queries with the help of the transformation rules. The sub-queries are sent to the information sources. The interchange between context transformation and integration is also presented.

3.1 Definition of the Queries

Queries are formulated with respect to the integrated view provided by the mediator. A query consists of a set of templates derived from the integrated view. The variables in the query templates indicate which information is relevant for the answer. Attributes which are irrelevant to a query can be omitted in the query template. Also, the query templates define the context in which the answer has to be presented.

Additionally, some expressions can be formulated in a query constraining the possible set of answers. The answer set is filtered through the expressions. To be more precise, a query is written as:

$$Q = \{T_1, T_2, ..., T_n, \phi_1, \phi_2, ..., \phi_m\}$$

The mediator the query is sent to first plans how the query can be decomposed into several sub-queries. Then the sub-queries are sent to the wrappers of the information sources. The results of the wrappers are integrated according to the plan generated in the first step.

During the planning phase, the mediator uses the transformation rules. The mediator acts as a goal–driven rule–based mediator (comparable to a PROLOG inference engine). It searches for transformation rules which can be applied to one template in the query. The application of a transformation rule is formally decided if the query template can be unified with the head of a transformation rule. The unification is defined in detail in [13].

The invocation of the transformation rule depends on its kind. For a combination rule, the body formulates the subgoals which have to be solved for the head of the rule. Following the well–known resolution principle, the query is refined where the query template is replaced by the body of the combination rule.

Replacement rules modify a query template in a query. Because replacement rules can be applied to a nested template in the query template (e.g. to a template nested in an attribute), a part of the query template is exchanged. With the invocation of the replacement rules, the query template is replaced in the query by the modified template and the rest of the rule–body is added to the query.

The mediator applies the transformation rules until the query only contains templates which can be mapped to the information sources. These templates formulate the sub-queries to the information sources.

Here we want to illustrate the behavior of the mediator through an example. First we give a query example. Suppose the broker wants to retrieve information about all stocks which cost more than 100 pounds and where he can buy these stocks. The query can be written as:

```
Query = {<broker-stocks, <>, complex,
              { name -> <name, <>, string, ?Name>@broker,
                price -> <price,
                           <(currency, pound), (scale,1)>,
                           integer, ?Price>@broker,
                place -> ?Place
              }>@broker,
          ?Price > 100}
```

The required information is indicated by the variables `?Name`, `?Price`, and `?Place`. The expression `?Price > 100` constrains the set of possible answers.

Figure 6 illustrates how the mediator plans the decomposition of the query. It first tries to decompose the query with the two integration rules given in figure 2 and 3. Then it applies the desired context rules. For readability reasons, in the figure 6 the query expression is omitted and only those context rules are given which focus on the price conversion. Note, that the context rules apply to templates located in attributes.

3.2 Interaction of Context Transformation and Integration

The plan shown in figure 6 is not the only possible plan; several different plans exist and can be generated in order to answer the query. For example, the

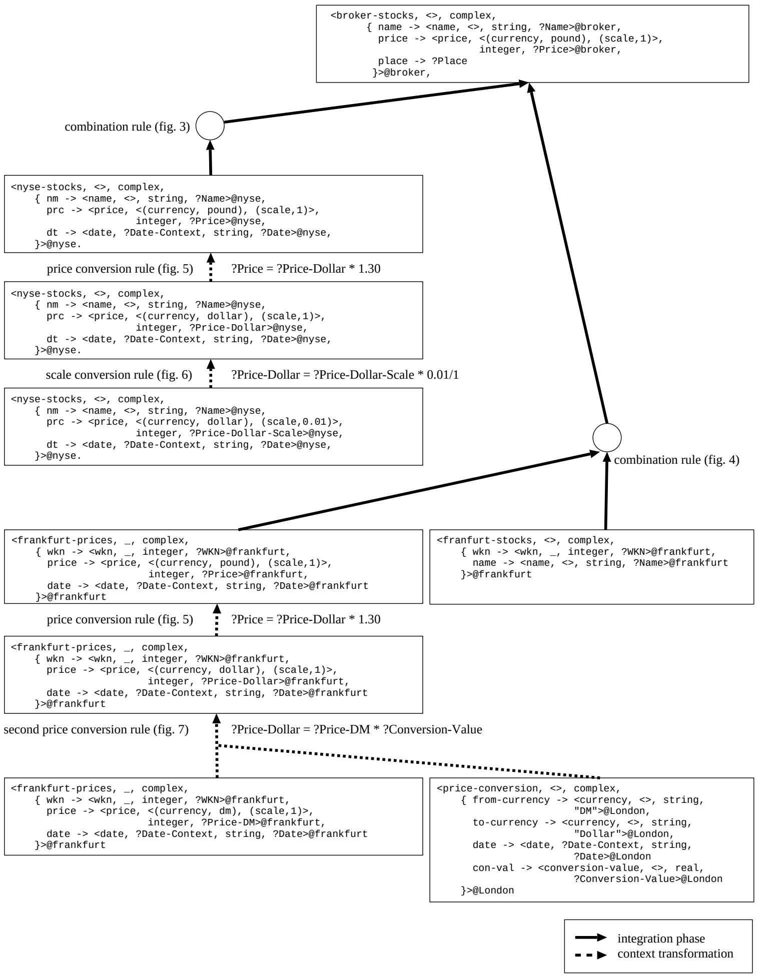

Figure 6: The plan generated by the mediator (illustrated in parts)

context transformation which converts the prices from `pound` to `dollar` can be shifted to the top. The mediator could first convert the query (i.e. the prices into `dollar`) and then distribute the query into the sub–query for both stock markets rather than first distribute the query and then make the conversion as illustrated in figure 6.

These plans illustrate how context transformation and integration can interact: context rules and integration rules are applied alternately. After the application of an integration rule, a set of context rules is applied before the next integration rule is invoked. However, the last phase of every plan is a context transformation. It cannot be expected that the sub-queries are in the desired context. So the last step of every plan has to be a context transformation which maps the sub-queries to the context of the information sources.

3.3 Sending Expressions to the Information Sources

Expressions formulated in the query are used to constrain the set of possible answers. But expressions also must be considered with respect to the context of the query. Therefore expressions cannot be sent to the information sources, but also have to be transformed in the contexts of the sources. For example, the query expression $?Price\text{-}Pound > 10$ expects that the price value appears in currency pound (and not in dollar or DM as the sources provide their price values). Forwarding the expression directly to an information source without a context transformation leads to an incorrect answer.

A first solution gathers all information of the information sources, transforms the information in the context of the query and then evaluates the query expression. Unfortually this approach gathers too much information from the source and therefore it is very inefficient. A better solution is to propagate also the expressions to a source in order to reduce the set of answers as early as possible. This needs a conversion of expressions to the context of an information source.

In our approach conversions are represented in context rules. More precisely, the expressions in the context rule describe the conversion in a declarative way. For the propagation of the query expression to the information source, the query expression and the conversion expression in the context rule have to be combined. For example, the query expression $?Price\text{-}Pound > 10$ can be combined with the conversion expression $?Price\text{-}Pound = ?Price\text{-}Dollar * 1.30$ into the expression $?Price\text{-}Dollar > 10/1.30$. The combined expression can be submitted to an information source reducing the set of instances returned by the information source.

For the propagation a mechanism is needed which can combine and reformulate expressions. Expressions can be considered as so–called constraints. Forbus and De Kleer [4] present a mechanism (i.e. a *constraint solver*) which is able to reformulate constraints symbolically. Such a constraint solver can easily be attached to the rule–based mechanism like the mediator. The integration follows the general CLP schema where standard logic programming paradigm is extended by any constraint solver. Constraints are not handled in the logic program but in the constraint solver. Whenever a constraint should be solved it is simply sent to the constraint solver. Formally, the logic program succeeded iff it can deduce the empty clause and all constraints in the constraint solver do not lead to a contradiction.

The mediator starts its query splitting in the normal way. But each expression is propagated to the constraint solver. When the mediator has reduced the query to the sub-queries for the information sources, the combined and reformulated expressions are extracted from the constraint solver. The sub–query together with the reformulated expressions now can be forwarded to the information source. Because of the reformulated expressions, the answer of an information source is directly reduced.

However, not all constraints stored in the constraint solver can be extracted. For a sub–query represented by a template only those expressions can be extracted which contain only variables of the template. Especially those expressions which connect the variables of more than one templates cannot be retrieved. But when the mediator receives the results from the first information sources, the variables in the sub–query are bound to concrete values. Before the next sub–query is sent to the next information source, these bindings are propagated again to the constraint solver leading to a refinement of the constraints in the constraint solver (i.e. eliminating some variables in the expressions). While continuing this process the variables in the expressions will be eliminated successively. When all instances of the information source are retrieved, their bindings are propagated to the constraint solver, and if the constraint solver does not detect a contradiction, this set of instances constitutes a valid answer of the information sources. The mediator then combines and integrates these instances according to the transformation rules.

4 Related Works

In TSIMMIS [7] a rule-based mediator is proposed. The information is described in the OEM model [11]. In OEM the objects are similarly represented as the templates. But in contrast to the labels in the OEM object description, templates contain in their contexts a more expressive and detailed representation of the semantics.

Furthermore, the context transformation can only be represented interleaved with the integration task in one rule. There is no distinction between either the different task or the integration and context rules. The separation first allows to concentrate on the two different heterogeneity problems separately. Second, it modularizes the set of rules and makes it much more easier to maintain and to reuse. Third, the declarative representation of context transformation allows also to propagate (query) expressions to the information sources and therefore to reduce the answers returned by the sources as early as possible.

Sciore et. al. [12] consider contexts and attached them to the simple values in order to get semantic values. They also define a context as a set of properties. For the context transformation Sciore et. al. define for each property a conversion function [12]. The conversion of a simple value into a target context results in the application of a set of conversion functions depending on the target context.

Sciore et. al. provide a simple but expressive mechanism for solving semantic heterogeneity problems. But unfortunally their approach lacks the reconciliation of structural heterogeneity problems.

In contrast to the conversion functions of Sciore et. al. [12], the context transformation is not hidden in the conversion functions but represented declaratively. The declarative representation allows first to apply context transformation also to complete expression (rather than a simple value) and second to propagate the expressions directly to the information sources in order to reduce their answers.

5 Conclusion

Mediators are developed for the flexible integration of several heterogeneous information sources. Rule–based mediators are configured with the help of rules. A rule describes the mapping between an information source and the integrated view provided by the mediator. But the rules mainly reconcile the structural heterogeneities.

In this paper we have shown how a rule–based mediator can be extended for the reconciliation of semantic heterogeneities. Contexts are used to clarify the representation details and semantics. Contexts explains how the semantics of information can differ and that a context transformation is needed. Based on the context, a new subset of rules — the context rules — are introduced for the context transformation. Context rules implement the conversion functions of Sciore et. al. [12], but more declarative and more expressive. These rules help to reduce errors and free an application programmer from being concerned with conversions.

Context rules require an extension of the rule representation formalism, because they have to be applied in nested object structures. Replacement rules are specified in order to exchange parts of an object. They are the adequate representation formalism for context transformation. Moreover they allow to specify the context transformation in a declarative way. From a logical point of view, they go beyond traditional logic programming paradigms. Furthermore, expression can be considered with respect to a context and can also be transformed. The transformed expressions can be forwarded to the information sources restricting the possible set of answers as early as possible.

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] J. Calmet, S. Jekutsch, P. Kullmann, and J. Schü. Komet — a system for integration of heterogeneous information sources. In *Proceedings of Tenth International Symposium on Methodologies for Intelligent Systems (ISMIS'97)*, October 15-18 1997.

[3] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards heterogeneous multimedia information systems: The garlic approach. In *Proceedings of the Fifth International Workshop on Research Issues in Data Engineering (RIDE)*, 1995.

[4] Kenneth D. Forbus and Johan De Kleer. *Building Problem Solvers.* Artificial Intelligence. The MIT Press, Cambridge, Massachusetts, USA, London, England, a bradford book edition, 1993.

[5] P. Frankhauser, M. Kracker, and E. Neuhold. Semantic vs. structural resemblance of classes. *SIGMOD Record*, 20(4), December 1991.

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

[7] 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.

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

[9] 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. 1997, Academic Press.

[10] W. Kim and J. Seo. Classifying schematic and data heterogeinity in multidatabase systems. *IEEE Computer*, 24(12):12–18, 1991.

[11] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proceedings Data Engineering Conference*, Taipei, Taiwan, March 1995.

[12] Edward Sciore, Michael Siegel, and Arnie Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, pages 254–290, June 1994.

[13] Holger Wache. A rule-based mediator for the integration of heterogeneous sources (extended version). TZI–berichte, University of Bremen, Bremen, 1999. to appear.

[14] David Wells. Wrappers. WWW: http://www.objs.com/survey/wrap.htm, 1997. Survey.

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