CONCEPTUAL MODELING OF MULTIDIALECT WORKFLOWS

L. Kalinichenko\textsuperscript{1,2}, S. Stupnikov\textsuperscript{1}, A. Vovchenko\textsuperscript{1}, and D. Kovalev\textsuperscript{1}

Abstract: This paper contributes to the techniques for conceptual representation of data analysis algorithms and data integration facilities as well as processes to specify data and behavior semantics in one paradigm. An investigation of a novel approach for applying a combination of semantically different platform-independent rule-based languages (dialects) for interoperable conceptual specifications over various rule-based systems (RSs) relying on the rule-based program transformation technique recommended by the W3C Rule Interchange Format (RIF) is extended here. Such approach is combined with the facilities aimed at the semantic rule-based mediation intended for the heterogeneous data base integration. This paper also presents a previous research of the authors in the direction of workflow modeling for composition of compositions of algorithmic modules in a process structure. A capability of the multidialect workflow support specifying the tasks in semantically different languages mostly suited to the task orientation is presented. A practical workflow use case, the interoperating tasks of which are specified in several rule-based languages (RIF-CASPD, RIF-BLD, RIF-PRD), is introduced. In addition, OWL 2 is used for the conceptual schema definition, RIF-PRD is used also for the workflow orchestration. The use case implementation infrastructure includes a production rule-based system (IBM ILOG), a logic rule-based system (DLV), and a mediation system.

Keywords: conceptual specification; workflow; RIF; production rule languages; database integration; mediators; PRD; multidialect infrastructure

DOI: 10.14357/19922264140413

1 Introduction

This work keeps on the intention of developing the facilities for conceptual declarative problem specification and solving in data intensive domains (DID). In [1] it was claimed that conceptual data semantics alone (e.g., formalized in ontology languages based on description logic) are insufficient, so that conceptual representation of data analysis algorithms as well as processes for problem solving are required to specify data and behavior semantics in one paradigm.

The results presented in this paper\textsuperscript{3} extend the research [1] aimed at the definition and implementation of the facilities for conceptually-driven problems specification and solving in DID aiming at ensuring eventually the following capabilities for expressing the specifications:

(1) an ability to provide complete and precise specification of the abstract structure and behavior of the domain entities, their consistency, relationship, and interaction;

(2) well-grounded diversity of semantics of the modeling facilities providing for the best attainable expressiveness, compactness, and precision of the definition of the problem solving algorithm specifications;

(3) arrangements for the extensions of the modeling facilities satisfying the changing technological and practical needs;

(4) specification independence from implementation platforms (languages, systems);

(5) specification independence from concrete information resources (IRs) (databases, services, ontologies, etc.) combined with facilities for their semantic integration and interoperability; and

(6) built-in methodologies for creation of unifying specification languages providing for construction of semantics-preserving mappings of conceptual specifications into their implementations in specific platforms.

The research reported in [1] investigated the conceptual modeling facilities for DID applying rule-based declarative logic languages possessing different, complementary semantics and capabilities combined with the methods and languages for heterogeneous data mediation and integration. Two fundamental techniques were combined: (i) constructing of the unifying extensible language providing for semantics-preserving mapping

\textsuperscript{1}Institute of Informatics Problems, Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation

\textsuperscript{2}Faculty of Computational Mathematics and Cybernetics, M.V. Lomonosov Moscow State University, 1-52 Leninskiye Gory, GSP-1, Moscow 119991, Russian Federation

\textsuperscript{3}This paper is an extended for the journal version of the results presented in the “Multidialect Workflows” report at the ADBIS’2014 Conference.
into it of various IR specification languages (e.g., such as data definition (DDL) and data manipulation (DML) languages for databases); and (ii) creation of the unified extensible family of rule-based languages (dialects) and a model of interoperability of the programs expressed in such dialects with different semantics.

The first technique is based on the experience obtained in course of the SYNTHESIS language development [2]. The kernel of the SYNTHESIS language is based on the object-frame data model used together with the declarative rule-based facilities in the logic language similar to a stratified Datalog with functions and negation. The extensions of the kernel are constructed in such a way that each extension together with the kernel is a result of semantic preserving mapping of some IR language into the SYNTHESIS [2]. The canonical information model is constructed as a union of the kernel with such extensions defined for various resource languages. Canonical model is used for development of mediators positioned between the users, conceptually formulating problems in terms of the mediators, and distributed resources. A schema of a subject mediator for a class of problems includes the specification of the domain concepts defined by the respective ontologies.

Another, multidialect technique for rule-based programs interoperability applied is based on the RIF standard [3] of W3C. Standard RIF introduces a unified family of rule-based languages together with a methodology for constructing of semantic preserving mappings of specific languages used in various RSs into RIF dialects. Examples of RSs include SILK, OntoBroker, DLV, IBM Websphere ILOG JRules, RIF4J + IRIS, and others (more examples can be found at http://www.w3.org/2005/rules/wiki/Implementations). From the RIF point of view, an IR is a program developed in a specific language of some RS.

In [1], the first results obtained were presented including the description of an approach and an infrastructure supporting:

- application domain conceptual specification and problem solving algorithms definitions based on the combination of the heterogeneous database mediation technique and the rule-based multidialect facilities;
- interoperability of distributed multidialect rule-based programs and mediators integrating heterogeneous databases; and
- rule delegation approach for the peer interactions in the multidialect environment.

The proof-of-concept prototype of the infrastructure based on the SYNTHESIS environment and RIF standards has been implemented. The approach for multidialect conceptualization of a problem domain, rule delegation, rule-based programs, and mediators interoperability were explained in detail and illustrated on an use-case in the finance domain [1]. For the conceptual definition of the use-case problem, the OWL was used for the domain concepts definition and two RIF logic dialects RIF-BLD [4] and RIF-CASPD [5] were used and mapped for implementation into the SYNTHESIS formula language and the ASP (answer set programming) based DLV [6] language, respectively.

The results obtained so far are quite encouraging for future work: they show that the mentioned in the beginning capabilities (1)–(6) sought for conceptual modeling become feasible. This paper reports the results of extending the research in the direction of modeling of the processes for the problem solving following the approach briefly outlined above. These results include extensions of the infrastructure and specification languages considered in [1] to the workflow level keeping the same approach and paradigm as well as aiming at the capabilities of the conceptualization (1)–(6) that were stated in [1] and mentioned in the beginning of the introduction.

For investigation of such extension with respect to the choice of rule-based languages, it was decided not to go outside the limits of the existing set of the published RIF dialects. Such decision would allow to retain well-defined semantics of the conceptual rule-based languages with a possibility to check preservation of their semantics by various languages of the implementing systems.

The production rule dialect RIF PRD [7] has been chosen as the language for the workflow modeling in such a way that the tasks of the workflow can have multidialect rule-based representation (as defined in [1]). This paper reporting the results of such investigation is structured as follows. To make the paper self-contained, the next section provides a brief overview of the infrastructure supporting multidialect programming defined in details in [1]. Here, it is stressed that this infrastructure is suitable for the workflow tasks specification. Workflow-oriented extension of the multidialect infrastructure is considered in section 3. Use case implementation in the proof-of-concept prototype is given in section 4. Related works are reviewed in section 5. Concluding Remarks summarize contributions of the research.

2 Basic Principles of the Workflow

Tasks Representation
in the Multidialect Infrastructure

Each workflow task (besides those that for pragmatic reasons are defined as externally specified functions) is assumed to be represented in the novel infrastructure
Figure 1 Conceptual schema and peer specifications

Informatica I EE Primeneniya — Informatics and Applications 2014 Volume 8 Issue 4

Semantics of a conceptual task definition in such setting becomes a multidialect one. The specification modules of a task are treated as peers. Mediation modules are assumed to be defined in RIF-BLD for representation of the mediator rules (to be interpreted in SYNTHESIS) supporting schema mapping and semantic integration of the IRs. Multidialect task is implemented by means of transformation of conceptual specifications into modular, component-based peer-to-peer (P2P) program represented in the languages of the MSs and RSs with the respective semantics. Interoperability of logic rule components of such distributed program is carried out by means of the delegation technique [1, section 3.3]. Production rule components are considered as external functions, interoperability is achieved through the mechanism of external terms.

A schema $S_R$ of a peer $R$ is a set of entities (classes or relations and their attributes) corresponding to extensional and intensional predicates of the resource implementing the peer $R$.

The RS or the MS of each peer $R$ should be a conformant $D_R$ consumer where $D_R$ is the respective RIF dialect (Fig. 1, left-hand part). Conformance is formally defined using formula entailment and language mappings [3].

The peer $R$ is relevant to a RIF-document $d$ of a conceptual specification of a problem (Fig. 1, right-hand part) if (i) $D_R$ is a subdialect of the document $d$ dialect (subdialect is a language obtained from some dialect by removing certain syntactic constructs and imposing respective restrictions on its semantics [4]; each program that conforms with the subdialect also conforms with the dialect) and (ii) entities of the peer schema $S_R$ (if they exist) are ontologically relevant to entities of the conceptual schema the names of which are used in $d$ for extensional predicates.

The schema of a relevant peer is mapped into the conceptual schema. The mapping establishes the correspondence of the conceptual entities referred in the document $d$ to their expressions in terms of entities of the schema $S_R$ using rules of the $D_R$ dialect. These schema mapping rules constitute separate RIF-document (Fig. 1, middle part).

Peers communicate using a technique for distributed execution of the rule-based programs. The basic notion of the technique is delegation-transferring facts and rules from one peer to another. A peer is installed on a node of the multidialect infrastructure. A node is a combination of a wrapper, an RS or an MS, and a peer (for the details, refer [1, Fig. 3]). A wrapper transforms programs and facts from the specific RIF dialect into the language of
the RS or MS and vice versa. A wrapper also implements
the delegation mechanism. Transferring facts and rules
among peers is performed in the RIF dialects.

A special component (Supervisor) of the architecture
defined in [1] stores shared information of the
environment, i.e., conceptual specifications related to
the domain and to the problem, a list of the relevant
resources, RIF-documents combining rules for the con-
ceptual specification and a resource schema mapping.

Implementation of the conceptual specification in-
cludes the following steps:

(1) rewriting of the conceptual documents into the
RIF-programs of the peers performed by the Su-
pressor. The rewriting includes also (i) replacing
the document identifiers (used to mark predicates) by
peer identifiers and (ii) adding schema mapping
rules to programs (Fig. 1, middle part);

(2) a transfer of the rewritten programs to nodes con-
taining peers relevant to the respective conceptual
documents. The transfer is performed by the Su-
pressor by calling the method loadRules of the
respective node wrappers;

(3) a transformation of the RIF-programs into the
concrete RS or MS languages. The transformation
is performed by the NodeWrapper or by the RS or
MS itself (if the RS or MS supports the respective
RIF dialect); and

(4) an execution of the produced programs in P2P
environment.

During the process of rewriting of the conceptual
schema into the resource programs, the relationships
between RIF-documents of the conceptual schema de-
ined by remote or imported terms are replaced by
relationships between peers also defined by remote or
imported terms. To implement remote and imported
terms, a rule delegation mechanism is used to transfer
facts and rules from one peer to another. The details
of rule delegation approach including description of the
related algorithms are provided in [1].

3 Workflow-Oriented Extension
of the Multidialect Infrastructure

The aim of the infrastructure proposed is a conceptu-
.al programming of problems in the RIF-dialects and
an implementation of conceptual specifications using
rule-based languages of the RSs and MSs. One of the
objectives of this particular paper is to introduce an
extension of the existing multidialect infrastructure [1]
aiming at the conceptual specification of rule-based
workflows.

Conceptual specification of a problem (class of prob-
lems) is defined in the context of a subject domain
and consists of a set of RIF-documents. Besides the
documents expressed in the logic dialects of RIF, the
documents expressed in the production rule dialect
(RIF-PRD) also can be a part of conceptual specifi-
cation of a problem. In particular, these documents are
aimed to express a process of solving the problem as the
production rule-based workflow.

3.1 Specification of workflow orchestration

A workflow consists of a set of tasks orchestrated by
specific constructs (workflow patterns [10], for instance,
sequence, split, join) defining the order of tasks execu-
tion. The specification of such orchestration is called
here a workflow skeleton. A skeleton is defined using RIF-
PRD production rules. Workflows and workflow patterns
can be represented using production rules in various
ways, e.g., as in [10, 11]. The approach applied in this
paper to represent workflows requires the extension of
RIF-PRD dialect by several built-in predicates (they are
considered to be a part of wkfl namespace referenced by
http://www.w3.org/2014/rif-workflow-predicate#. URI
similarly to func and pred namespaces defined in [12] for
built-in functions and predicates of RIF):

- predicate wkfl:end-of-task(?arg) where ?arg is an
  identifier of a task. The value space of ?arg is the
  XML-Schema built-in data type xsd:Name representing
  XML names. The predicate turns into true if a task ?arg
  has been completed;

- predicate wkfl:variable_definition(?arg1 ?arg2)
  where ?arg1 is the identifier of a variable and ?arg2
  is the identifier of a type of the variable. The value
  space for both arguments is xsd:Name. Turning the
  predicate into true means that a variable ?arg1 of
  type ?arg2 is defined in the context of a workflow;

- predicate wkfl:variable_value(?arg1 ?arg2)
  where ?arg1 is the identifier of a variable and ?arg2
  is the value of the variable. The value space for the
  first argument is xsd:Name, the value space for the
  second argument is the union of value spaces of all
  RIF built-in datatypes. Turning the predicate into
  true means that a variable ?arg1 has the value ?arg2;

- predicate wkfl:parameter_definition(?arg1 ?arg2
  ?arg3) where ?arg1 is the identifier of a work-
  flow parameter; ?arg2 is the identifier of a type of
  the parameter; and ?arg3 is the direction of the
  parameter. The value space for the first and for the
  second arguments is xsd:Name. The value space for
  the third argument is {IN, OUT, IN_OUT} (input, output,
  or input–output parameter). Turning the predicate
  into true means that a parameter ?arg1 of type ?arg2,
  and direction ?arg3 is defined for a workflow; and

INFORMATIKA I EE PRIMENENIYA — INFORMATICS AND APPLICATIONS 2014 volume 8 issue 4 113
— predicate \( \text{wkfl:parameter.value(?arg1 ?arg2)} \) defines values of workflow parameters in the same way as \( \text{wkfl:variable.value} \) defines values of workflow variables.

Predicates \( \text{wkfl:variable.definition} \) and \( \text{wkfl:variable.value} \) allow to specify workflow variables and their values and thus to organize the data flow within a workflow. Predicates \( \text{wkfl:parameter.definition} \) and \( \text{wkfl:parameter.value} \) allow to specify workflow parameters and their values and thus to define the interface of a workflow in terms of input and output parameters. Using of workflow parameters and variables is illustrated in the Appendix.

The predicate \( \text{wkfl:end-of-task(?arg)} \) allows to orchestrate the order of execution of workflows tasks using conditions and actions of production rules. In this section, the template rules intended for representation of several basic workflow patterns (Fig. 2) are provided.

![Figure 2](image-url)  
*Figure 2*  
Basic workflow patterns

Three well-known workflow patterns are considered below: Sequence, AND-Split, and AND-Join.

The AND-Split\(^1\) workflow pattern is represented in RIF-PRD by the following production rule template using \( \text{wkfl:end-of-task} \) predicate:

\[
\begin{align*}
&\text{If Not(External(wkfl:end-of-task(A)))} \\
&\text{Then Do (Act(A))} \\
&\text{Assert(External(wkfl:end-of-task(A))))} \\
&\text{If And(Not(External(wkfl:end-of-task(B)))} \\
&\text{External(wkfl:end-of-task(A))))} \\
&\text{Then Do (Act(B))} \\
&\text{Assert(External(wkfl:end-of-task(B))))} \\
&\text{If And(Not(External(wkfl:end-of-task(C)))} \\
&\text{External(wkfl:end-of-task(A))))} \\
&\text{Then Do (Act(C))} \\
&\text{Assert(External(wkfl:end-of-task(C))))}
\end{align*}
\]

The template includes three rules for tasks \( A, B, \) and \( C \), respectively. \( \text{Act(A)} \), \( \text{Act(B)} \), and \( \text{Act(C)} \) denote actions associated with tasks \( A, B, \) and \( C \). Orchestration (tasks \( B \) and \( C \) are executed concurrently right after task \( A \) is completed) is specified using \( \text{wkfl:end-of-task} \) predicate in conditions and Assert actions of rules.

Similarly, the AND-Split pattern is represented in RIF-PRD by the following production rule template:

\[
\begin{align*}
&\text{If Not(External(wkfl:end-of-task(A)))} \\
&\text{Then Do (Act(A))} \\
&\text{Assert(External(wkfl:end-of-task(A))))} \\
&\text{If And(Not(External(wkfl:end-of-task(B)))} \\
&\text{External(wkfl:end-of-task(A))))} \\
&\text{Then Do (Act(B))} \\
&\text{Assert(External(wkfl:end-of-task(B))))} \\
&\text{If And(Not(External(wkfl:end-of-task(C)))} \\
&\text{External(wkfl:end-of-task(A))))} \\
&\text{Then Do (Act(C))} \\
&\text{Assert(External(wkfl:end-of-task(C))))}
\end{align*}
\]

More complicated patterns like OR-, XOR- splits and joins, structured loops, subflows, and others are represented in RIF-PRD similarly.

### 3.2 Workflow tasks specification

Workflow tasks can be specified as:

— separate RIF-documents in various logic RIF-dialects (this is the way how multidialect infrastructure [1] is extended with workflow capabilities);
— separate RIF-documents in the RIF-PRD dialect;
— set of production rules embedded into the workflow skeleton; and
— external functions treated as “black boxes.”

Semantics of tasks specified as multidialect logic programs are defined in accordance with the RIF-FLD [3] standard and standards for the respective RIF-dialects (BLD, CASPD, etc.). Semantics of tasks specified as production rule programs are defined in accordance with the RIF-PRD standard. Semantics of external functions “are assumed to be specified externally in some document” [3].

All kinds of tasks (except those that are embedded into a workflow skeleton) are referenced in the workflow skeleton as external terms [3] like \text{External(t)} where term \( t \) is defined by an external resource identified by internationalized resource identifier (IRI) [3].

### 3.3 Workflow implementation infrastructure

Workflows defined in the conceptual specification are implemented in the environment shown in Fig. 3. Environ
Conceptual modeling of multidialect workflows

A production system compliant with the OMG Production Rule Representation [13] and with external functions, implemented as web-services. Implementation of the conceptual specification includes the following steps:

1. transfer of the conceptual RIF-documents constituting a workflow skeleton to the production rule-based system node (performed by the Supervisor component);
2. transformation of the conceptual RIF-documents constituting a workflow skeleton into the language of the production rule-based system (performed by the PRS Wrapper component);
3. transferring RIF logic programs related to tasks to the relevant nodes of the environment and transformation of the RIF-programs into the concrete RS or MS languages [1]; and
4. execution of the workflow.

The interface of the Supervisor includes methods for submitting and executing a workflow represented as a set of RIF-documents, and for getting the result of the workflow execution.

To provide a proof of the multidialect infrastructure concept, a use case in the financial domain has been implemented. The problem to be solved in the use case is called the investment portfolio diversification problem. The detailed description of the use case is included in the Appendix.

4 Related Work

Two types of workflow models, namely, abstract and concrete, were identified [14]. In the abstract model, a workflow is described in an abstract form, without referring to specific resources. In this paper, workflow representation in abstract and platform-independent form is suggested.

A classification model for scientific workflow characteristics [10] contributes to better understanding of scientific workflow requirements. The list of structural patterns discovered during this analysis (including sequential, parallel, parallel-split, parallel-merge, and mesh) influenced the choice of the required workflow patterns.

The OMG standard [13] reflects an attitude to production rules from the industrial side providing an OMG MDA (model-driven architecture) platform-independent model (PIM) with a high probability of support at the PSM (platform-specific model) level from the rule engine vendors. Similar capabilities though formally defined are used as the basis for the RIF-PRD [7].

Some vendors of such production rule engines have extended their languages with the workflow specification capabilities. IBM has extended ILOG to provide the ruleflow capability. Microsoft supports Windows Workflow Foundation as a platform providing the workflow and rules capabilities. The examples of specific formalisms for PIM rule-based process specifications are also provided in [11].

Comparing to the known variants of the PIM production rule representations, selection of the RIF-PRD is considered to be well grounded:

1. the RIF-PRD is formally defined;
2. RIF ensures support of interoperability of modules written in different rule-based dialects with different semantics;
3. RIF provides foundations for PIM to PSM semantic preserving transformation; and
4. RIF also provides ability for specification of the concepts in application domain terms combining rule-based specifications with the OWL ontologies.
Importance of providing the interdialect interoperability is advocated in [15] for combining the functionalities of production systems and logic programs for abductive logic programming (ALP). The ALP framework gives a model-theoretic semantics to both kinds of rules and provides them with powerful proof procedures, combining backward and forward reasoning.

Papers related to RIF-PRD experimentations are focused mainly on the issue of the PRD programs transformation to an implementation system. In [16], a case study of bridging the ILOG Rule Language (IRL) to RIF-PRD and vice versa is considered. In [17], implementation of RIF-PRD in three different paradigms: Answer Set Programming, Production Rules, and Logic Programming (XSB) is investigated.

The contribution of this paper with regard to previous works of the authors [1] consists in extensions of the infrastructure and specification languages considered in [1] to the workflow level.

5 Concluding Remarks

Progress in the investigation of the infrastructure [1] for the conceptual multidialect interoperable programming in the abstract, rule-based, platform-independent notations is reported. An extension of the coherent combination of the multidialect rule-based programming technique recommended by the W3C RIF with the approach for unifying modeling of heterogeneous data bases for their semantic mediation is presented. The extension of the infrastructure and specification languages considered in [1] in the direction of the workflow modeling is presented.

Sticking to the limits of the existing set of the published RIF dialects, a capability of the multidialect workflow support is presented with the tasks specified in semantically different languages mostly suited to the task orientation. Also, a realistic problem solving use case containing the interoperating tasks specified in several platform-independent rule-based languages: RIF-CASPD, RIF-BLD, RIF-PRD, is presented. In addition, OWL 2 is used for the conceptual schema definition, RIF-PRD is applied for the workflow orchestration. The platforms selected for implementation of the tasks include: DLV, SYNTHESIS, IBM ILOG.

Such approach retains well-defined semantics of the platform-independent rule-based languages with a possibility to check preservation of their semantics by various languages of the implementing systems. The principle of independence of tasks from the specific IRs is carried out by the heterogeneous database mediation facilitates contributing to the reuse of tasks and workflows. Alongside with the further extension of the approach, in the future work, the authors plan to apply the conceptual multidialect programming philosophy for support of the experiments in data intensive sciences. In particular, they plan to investigate modeling hypotheses in astronomy representing them as a set of rules applying the multiplicity of the dialects required.

APPENDIX A

MULTIDIALECT WORKFLOW USE CASE

A.1 Investment portfolio diversification problem extended

Motivation of the use case that illustrates the proposed approach comes from the finance area. The use case extends the investment portfolio diversification problem defined in [1, Appendix] by adding workflow orchestration applying the RIF-PRD. The idea of the portfolio diversification problem is as follows. The portfolio is a collection of securities of companies, and its size is the number of securities in the portfolio. The problem is to build a diversified portfolio of maximum size. Diversification means that the prices of the securities in portfolio should be almost independent of each other. If the price of one security falls, it will not significantly affect the prices of others. Thus, the risk of a portfolio sharp decrease is reduced.

The input data for the problem is a set of securities and respective time series of indicators of the security price for each security. Time series for each security is a set of pairs \((d, v)\) where \(d\) is a date and \(v\) is an indicator of the security price (for instance, closing price). The financial services Google Finance (https://www.google.com/finance) and Yahoo! Finance (http://finance.yahoo.com/) are considered. They include various indicators of the security price for all trading days of the last decades. For the diversified portfolio, the securities having noncorrelated time series should be used. Noncorrelation of the time series means that their correlation is less than some predetermined price correlation value. The output data for the problem is a set of subsets of securities of the maximum size, for which the pair wise correlation will be less than the predetermined one.

The maximum satisfying subset of securities is calculated in the following way. Let \(G\) be a graph where the vertices are the securities. An edge between two securities exists if absolute value of their correlation is less than a specified number. So, any two securities connected by an edge are considered as noncorrelated. In such case, the problem of finding the portfolio of the maximum size is exactly the problem of finding a maximum clique in an undirected graph. A maximal clique is a maximal portfolio. Note that several different maximal portfolios can be found.

The conceptual specification of the use case [1] used two RIF-dialects: RIF-BLD and RIF-CASPD. The use case was implemented in the environment containing a mediation system used as a platform for RIF-BLD [4] and ASP-based DLV system [6] — a platform for RIF-CASPD. The RIF-BLD was used to specify the problem of data integration, and RIF-CASPD — the problem of finding a maximum clique in an undirected graph.
In this work, the portfolio use case is extended in the following way. The goal is not only to build a set of diversified portfolios, but also to choose the “best” of them according to some criteria. There are several approaches to choose the most appropriate portfolio.

The most recognized one is based on the Markovitz portfolio theory [18]. The idea is to choose the portfolio, which has the maximum risk/return ratio. The most well-known metric to operate with risk/return is Sharpe-ratio [19]: \( (r_p - r_f)/\sigma^2 \). Here, \( r_p \) denotes the expected return of the portfolio; \( r_f \) denotes the risk free rate; and \( \sigma^2 \) denotes the portfolio standard deviation (risk). The more the Sharpe-ratio is, the better the investment is.

Another approach is based on an idea that with the advent of social networks, it became possible to monitor ideas, sentiments, actions of people and lots of available information to do with the markets and investments. In [20], Bollen et al. draw the connection between the mood of investor tweets and the move of Dow Jones Index, stating that correlation between them is more than 80%. The idea of using tweets to assess market movements has been implemented in several hedge funds.

Combining these two strategies could provide benefits of both of them, which leads to the following problem statement: having S&P500 (a stock market index maintained by the Standard & Poor’s, comprising 500 large-cap American companies) list of companies, compute the diversified portfolio of maximum size with the best risk/return and sentiment ratios.

A.2 Conceptual specification of the application domain and the problem

Conceptual schema (ontology) of the application domain of historical prices of securities is written in the simplified OWL functional syntax [8] (Declaration keyword is omitted; property, domain, and range declarations are combined).

```
Ontology(<http://synthesis.ipi.ac.ru/portfolio/ontology>)
Class(Portfolio)
  ObjectProperty(securities domain(Portfolio) range(Portfolio))
  DataProperty(expected_return domain(Portfolio) range(xsd:double))
  DataExactCardinality(1 expected_return Portfolio)
  DataProperty(std_dev domain(Portfolio) range(xsd:double))
  DataExactCardinality(1 std_dev Portfolio)
  DataProperty(sharpe_ratio domain(Portfolio) range(xsd:double))
  DataExactCardinality(1 sharpe_ratio Portfolio)
  DataProperty(twitter_positive_ratio domain(Portfolio) range(xsd:double))
  DataExactCardinality(1 twitter_positive_ratio Portfolio)
```

```
Class(Security)
  DataProperty(ticker domain(Security) range(xsd:string))
  DataExactCardinality(1 ticker Security)
  DataProperty(rates domain(Security) range(StockRate))
  DataProperty(positive_tweets domain(Security) range(xsd:double))
  DataExactCardinality(1 positive_tweets Security)
  DataProperty(sec_expected_return domain(Security) range(xsd:double))
  DataExactCardinality(1 sec_expected_return Security)
  DataProperty(sec_std_dev domain(Security) range(xsd:double))
  DataExactCardinality(1 sec_std_dev Security)
Class(StockRate)
  DataProperty(date domain(StockRate) range(xsd:date))
  DataExactCardinality(1 date StockRate)
  DataProperty(price domain(StockRate) range(xsd:double))
  DataExactCardinality(1 price StockRate)
```

A portfolio (the Portfolio class) is characterized by a set of securities (security attribute) contained in the portfolio, by several metrics: expected return (expected_return attribute), standard deviation (std_dev attribute), Sharpe ratio (sharpe_ratio attribute), risk free rate (risk_free_rate attribute), and ratio of positive tweets mentioning securities of the portfolio (twitter_positive_ratio attribute).

A security (the Security class) is characterized by identifiers (ticker attribute), time series of historical prices (attribute rates), ratio of positive tweets mentioning the security (positive_tweets attribute), expected return (sec_expected_return attribute), and standard deviation (sec_std_dev attribute).

The workflow of the extended portfolio problem is demonstrated in Fig. 4. The workflow contains six tasks:

1. **getPortfolios.** A set of diversified portfolio candidates is computed. The multidialect task specification consists of two RIF-documents in BLD and CASPD dialects [1, Appendix]. Portfolios received as a result contain only security tickers, they have to be augmented by financial and sentiments ratios;

2. **getPositiveTweetRatio.** This task is responsible for computing a sentiment ratio of tweets for every security. Every

---

1To save space, specifications are provided only for getPortfolios, getPositiveTweetRatio, and computePortfolioTwitterMetrics tasks.
tweet is assessed to be positive, negative, or neutral. The task is specified as a call of external function;

(3) computePortfolioTwitterMetrics. The portfolio sentiment ratio is computed as the average of its securities sentiment ratio. The task is specified using RIF-PRD;

(4) getSecurityFinancialMetrics. For every security in a portfolio the financial rates (the expected return and the standard deviation) are calculated on the basis of historical rates of securities specified as an OWL 2 class of the ontology of the application domain. The task is specified using RIF-BLD dialect;

(5) computePortfolioFinancialMetrics. The computation of the portfolio expected return, risk, and Sharpe-ratio is done within this task. The task is specified using RIF-PRD dialect; and

(6) choosePortfolio. The best portfolio is chosen according to maximizing the \((\text{Sharpe ratio} \ast \text{sentiment ratio})\) coefficient. The task is specified using RIF-PRD dialect.

Workflow skeleton is specified as a RIF-PRD document importing the ontology of the application domain:

```
Document( Dialect(RIF-PRD)
Base(<http://synthesis.ipi.ac.ru/portfolio/workflow#>)
Import(<http://synthesis.ipi.ac.ru/portfolio/ontology#>)
Prefix(<http://www.w3.org/ns/entailment/OWL-Direct>)
Prefix(ofws<http://synthesis.ipi.ac.ru/synthesis/projects/RuleInt/OpinionFinderWS#>)
Prefix(mws<http://synthesis.ipi.ac.ru/synthesis/projects/RuleInt/MediatorWS#>)
Group 2 ( Do( Assert(External(wkfl:parameter-definition( startDatexsd:string IN)))
Assert(External(wkfl:parameter-definition( endDatexsd:string IN)))
Assert(External(wkfl:parameter-definition( bestPortfolio ont:Portfolio OUT)))
Assert(External(wkfl:variable-definition( ps List<ont:Portfolio> IN)))
Assert(External(wkfl:variable-value(ps List())))
) )
Group 1 ( Forall ?sd ?ed such that ( External(wkfl:parameter-value(startDate ?sd))
External(wkfl:parameter-value(endDate ?ed)) )
( If Not(External(wkfl: end-of-task(getPortfolios)))
Then Do( Modify(External(wkfl:variable-value(ps)
External(mws:getPortfolios(?sd ?ed))))
Assert(External(wkfl: end-of-task(getPortfolios))) )
)
( If And( Not(External(wkfl: end-of-task(getTweets)))
External(wkfl:end-of-task(getPortfolios)))
Then Do( Modify(?s[positive_tweets->
External(ofws:computeSecPosTweets(?t))]
Assert(External(wkfl: end-of-task(getTweets))) )
)
Forall ?ps ?p such that ( External(wkfl:variable-value(ps ?ps))
?p#?ps)
( If And(Not(External(wkfl: end-of-task(countTwitterMetrics)))
External(wkfl:end-of-task(getTweets)) )
Then Do( Modify(?p[twitter_positive_ratio->
External(func:numeric-divide(
Sum{?pt | Exists
External(func:count(?ps)))))
Assert(External(wkfl: end-of-task(countTwitterMetrics)))
) )
))
)
```

Production rules of the document are divided into two groups. The first group with priority 2 contains rules defining workflow parameters and variable. Input parameters are start date and end date of historical rates used for calculation of portfolio metrics. Workflow variable ps denotes a set containing portfolio candidates.
The second group with priority 1 contains the orchestration rules — workflow skeleton. The only orchestration rule provided in the example above corresponds to the task getPortfolios. The external function getPortfolios encapsulates a multidialect logic program calculating portfolio candidates [1, Appendix]. A Modify action is used to call the function and to put the returned result into the ps variable.

A.3 Revised portfolio problem infrastructure

The implementation structure of the use case is shown in Fig. 5. The RIF-PRD workflow skeleton was transformed into a program (rule set) in the ILOG [21] language combining production rules and workflow facilities (like fork and sequence). The ILOG program was executed in the IBM Operational Decision Manager tool [22]. In order to execute ILOG programs, the underlying execution model (XOM) [23] was defined as a set of Java classes: Portfolio, Security, and StockRate. The Portfolio class corresponds to a financial portfolio and contains as attributes a set of securities in it, its expected return, standard deviation, Sharpe ratio, and twitter positive ratio. Code of this class is provided below:

```java
public class Portfolio {
    private Collection<Security> securities;
    private double expected_return;
    private double std_dev;
    private double sharpe_ratio;
    private double twitter_positive_ratio;
    // as of 05.04.14 US 5-year treasuries
    private static double risk_free_rate = 0.0169;
    private boolean recommended;
}
```

Class Security corresponds to real world financial securities. The class contains as attributes a ticker, ratio of positive tweet number to the sum of positive and negative tweets, a set of stock rates, security’s standard deviation, expected return. These attributes are set as responses to corresponding web services queries:

```java
public class Security {
    public String ticker;
    public double positive_tweets;
    public Collection<StockRate> rates;
    public double std_dev;
    public double expected_return;
    public static int number_of_periods = 5;
}
```

StockRate is a simple class and contains just two attributes — price and date:

```java
public class StockRate {
    public float price;
    public String date;
}
```

It is easy to see that the one-to-one mapping exists between conceptual schema entities and execution model entities. Parameters of RIF-PRD workflow skeleton (startDate, endDate, and bestPortfolio) are mapped into the respective parameters of ILOG rule set (Fig. 6).

The variable of RIF-PRD workflow skeleton (ps) is mapped into a local variable of the rule set. Specification of the variable looks as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ilog.rules.studio.model.base:VariableSetxmi:
    version="2.0",
    xmlns:xmi="http://www.omg.org/XMI",
    xmlns:ilog.rules.studio.model.base =
    "http://ilog.rules.studio/model/base.ecore">  
    <name>local_vars</name>  
    <variables name="ps" type="java.util.ArrayList"
        initialValue="verbalization="ps"/>
</ilog.rules.studio.model.base:VariableSet>
```

Rules of the RIF-PRD workflow skeleton are mapped into ILOG ruleflow [23]:

```
flowtask portfolio$_$flow {
    property mainflowtask = true;
    property ilog.rules.business_name =
```

Figure 5 Portfolio problem implementation infrastructure
"portfolio_flow";
body {
    portfolio$_$flow#getPortfolios;
    fork {
        portfolio$_$flow#getRates;
        portfolio$_$flow
computePortfolioFinancialMetrics;} &&
    { portfolio$_$flow#getTweets;
        portfolio$_$flow#computePortfolioTwitterMetrics;}
    portfolio$_$flow#choosePortfolio;
};

ruletask portfolio$_$flow#getPortfolios {
    property ilog.rules.business_name = "portfolio_flow>getPortfolios";
    body { getPortfolios.*}
};

ruletask portfolio$_$flow#computePortfolioTwitterMetrics {
    property ilog.rules.business_name = "portfolio_flow>computePortfolioTwitterMetrics";
    body { computePortfolioTwitterMetrics.* }
};

ruletask portfolio$_$flow#getTweets {
    property ilog.rules.business_name = "portfolio_flow>getTweets";
    property ilog.rules.package_name = "";
    body {getTweets.*}
};

The computePortfolioTwitterMetrics, computePortfolioFinancialMetrics, and choosePortfolio tasks are implemented as production rules in ILOG:
package computePortfolioTwitterMetrics {
use ps;
import portfolio.*;

rule computePortfolioTwitterMetrics {
    property status = "new";
    when { IlrContext() from ?context; }
    then {
        foreach (Portfolio p in ps) {
            double ?twitter_metrics = 0;
            int ?length = 0;
            foreach (Security security in p.securities) {
                ?twitter_metrics = ?twitter_metrics +
                    security.positive_tweets;
                ?length = ?length + 1;
            }
            p.twitter_positive_ratio =
                ?twitter_metrics / ?length;
        }
    }
};

The getPortfolios and computeSecurityFinancialMetrics tasks are implemented by the following production rules in ILOG:
package getPortfolios {
use ps;
import portfolio.*;

rule getPortfolios {
    when { IlrContext() from ?context; }
    then {
        ps = Supervisor.getPortfolios(startDate, endDate);
    }
};

Here, the Supervisor is the Java class wrapping execution of logic programs in multidialect infrastructure including two nodes [1]. The nodes correspond to the mediation system (which integrates Google Finance and the Yahoo! Finance services) and to a rule-based programming system DLV.

The getSecurityFinancialMetrics task uses the same instance of the mediation system as the getPortfolios task. The reason is that financial metrics are calculated using the historical rates of the securities. This is exactly the information that is extracted by the mediation system from Google Finance and Yahoo! Finance. The difference between two tasks is that the getPortfolios is implemented as a submission of a query to the DLV node, but the getSecurityFinancialMetrics is implemented as a submission of a different query to the Mediation Node.

The getPositiveTweetRatio task is implemented by the following production rule in ILOG:
package getTweets {
use ps;
import portfolio.*;

rule getTweets {
    when { IlrContext() from ?context; }
    then {
        foreach (Portfolio p in ps) {
            double ?twitter_metrics = 0;
            int ?length = 0;
            foreach (Security security in p.securities) {
                ?twitter_metrics = ?twitter_metrics +
                    security.positive_tweets;
                ?length = ?length + 1;
            }
            p.twitter_positive_ratio =
                ?twitter_metrics / ?length;
        }
    }
};
Table 1 Metrics for the securities

<table>
<thead>
<tr>
<th>Security identifier</th>
<th>Expected return</th>
<th>Standard deviation</th>
<th>Positive tweet ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>COG</td>
<td>0.163</td>
<td>0.201</td>
<td>0.507</td>
</tr>
<tr>
<td>DO</td>
<td>0.015</td>
<td>0.019</td>
<td>0.651</td>
</tr>
<tr>
<td>EQR</td>
<td>0.150</td>
<td>0.022</td>
<td>0.846</td>
</tr>
<tr>
<td>FOSL</td>
<td>0.513</td>
<td>0.030</td>
<td>0.579</td>
</tr>
<tr>
<td>SCG</td>
<td>0.050</td>
<td>0.010</td>
<td>0.622</td>
</tr>
</tbody>
</table>

Table 2 Metrics for the portfolio candidates

<table>
<thead>
<tr>
<th>Portfolio identifier</th>
<th>Expected return</th>
<th>Standard deviation</th>
<th>Risk free rate</th>
<th>Sharpe ratio</th>
<th>Positive tweet ratio</th>
<th>Sharpe ratio × Positive tweet ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.111</td>
<td>0.008</td>
<td>0.0169</td>
<td>11.755</td>
<td>0.660</td>
<td>7.758</td>
</tr>
<tr>
<td>2</td>
<td>2.400</td>
<td>0.507</td>
<td>0.0169</td>
<td>4.701</td>
<td>0.508</td>
<td>2.388</td>
</tr>
<tr>
<td>3</td>
<td>2.381</td>
<td>0.508</td>
<td>0.0169</td>
<td>4.662</td>
<td>0.557</td>
<td>2.597</td>
</tr>
<tr>
<td>4</td>
<td>2.347</td>
<td>0.505</td>
<td>0.0169</td>
<td>4.606</td>
<td>0.708</td>
<td>3.261</td>
</tr>
<tr>
<td>5 (best)</td>
<td>0.178</td>
<td>0.011</td>
<td>0.0169</td>
<td>14.227</td>
<td>0.641</td>
<td>9.120</td>
</tr>
<tr>
<td>6</td>
<td>0.147</td>
<td>0.008</td>
<td>0.0169</td>
<td>15.577</td>
<td>0.521</td>
<td>8.166</td>
</tr>
</tbody>
</table>

Here, WebServices is the Java-class wrapping invocation of a web-service. The WSDL specification of the web-service can be found at http://synthesis.ipi.ac.ru/synthesis/projects/RuleInt/OpinionFinderWS. The web-service, in its turn, encapsulates a Java-program. The program first collects tweets using the Twitter Streaming API. After that, a sentiment analysis is done by the Polarity Classifier of the OpinionFinder tool [24] which assesses if tweet is positive, negative, or neutral. Finally, the sentiment ratio for every security in a portfolio is calculated and returned as the result.

A.4 Result of the use case workflow execution

The results obtained by one of the use case runs are as follows. The task getPortfolioscomputes portfolio candidates on the basis of historical rates of daily closing prices of securities from S&P500 list for the 2011–2013. Six portfolios of size 5 were calculated. Each portfolio is a set of identifiers (tickers) of companies:

- Candidate 1: { ALXN, BF.B, EW, POM, VNO }
- Candidate 2: { BMC, JBL, LUK, MNST, POM }
- Candidate 3: { AVP, BMC, JPL, MNST, POM }
- Candidate 4: { ALTR, BF.B, BMC, DGX, PEG }
- Candidate 5: { COG, DD, EQR, FOSL, SCG }
- Candidate 6: { ADSK, GILD, INTC, POM, TJX }

The task getSecurityFinancialMetrics computes the expected return and the standard deviation for every security mentioned in portfolio candidates. The task getPositiveTweetRatio computes positive sentiment ratios for every security mentioned in portfolio candidates (500 latest tweets for every security were used for the computation). Financial and twitter metrics for several securities are provided in Table 1. The task computePortfolioFinancialMetrics computes financial metrics for every portfolio candidate on the basis of respective metrics for securities in a portfolio. The task computePortfolioTwitterMetrics computes sentiment metrics for every portfolio candidate on the basis of sentiment metrics for securities in a portfolio. Financial and twitter metrics for portfolio candidates are provided in Table 2. The task choosePortfolio identifies the best portfolio by maximum value of the products of Sharpe ratio and positive tweet ratio obtained for every portfolio (see Table 2).

Acknowledgments

This research has been done under the support of the RFBR (projects13-07-00579, 14-07-00548) and the Program for Basic Research of the Presidium of RAS.

References


Received November 3, 2014

Contributors

Kalinichenko Leonid A. (b. 1937) — Doctor of Science in physics and mathematics, professor; Head of Laboratory, Institute of Informatics Problems, 44-2 Vavilov Str., Moscow 119333, Russian Federation; professor, Faculty of Computational Mathematics and Cybernetics, M. V. Lomonosov Moscow State University, 1-52 Leninskiye Gory, GSP-1, Moscow 119991, Russian Federation; leonidandk@gmail.com

Stupnikov Sergey A. (b. 1978) — Candidate of Science (PhD) in technology, senior scientist, Institute of Informatics Problems, Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation; ssa@ippi.ac.ru

Vovchenko Alexey E. (b. 1984) — Candidate of Science (PhD) in technology, senior scientist, Institute of Informatics Problems, Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation; itshein@gmail.com

Kovalev Dmitry Yu. (b. 1988) — programmer, Institute of Informatics Problems, Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation; dm.koval@gmail.com
Концептуальное моделирование мультидиалектных потоков работ

Л. А. Калиниченко1,2, С. Ступников1, А. Вовченко1, Д. Ковалев1

1 Институт проблем информатики Российской академии наук
2 Московский государственный университет им. М. В. Ломоносова, факультет вычислительной математики и кибернетики

Аннотация: Рассматриваются методы концептуального представления алгоритмов анализа данных, средств интеграции данных, а также процессов, направленных на спецификацию семантики данных и поведения в единой парадигме. Расширяется новый подход к применению комбинации семантически различных платформонезависимых языков на правилах (диалектов) для создания интероперабельных концептуальных спецификаций над различными системами на правилах. Подход опирается на методику трансформации программ на правилах, рекомендованную стандартом W3C Rule Interchange Format (RIF). Подход, предлагаемый в стандарте RIF, сочетается со технологией семантической интеграции неоднородных баз данных в предметных посредниках. Статья расширяет предыдущие исследования авторов в направлении моделирования потоков работ для определения композиций алгоритмических модулей в процессной структуре. Рассмотрены возможности спецификации задач в мультидиалектных потоках работ с применением семантически различных языков, наиболее подходящих для конкретных задач. Приведен практический пример потока работ, задачи которого специфицированы с использованием нескольких языков на правилах (RIF-CASPD, RIF-BLD, RIF-PRD). Для определения концептуальной схемы использован язык OWL 2, для оркестровки потока работ использован язык RIF-PRD. Инфраструктура реализации примера включает систему на продукционных правилах (IBM ILOG), систему на логических правилах (DLV) и предметный посредник.

Ключевые слова: концептуальные спецификации; потоки работ; RIF; языки продукционных правил; интеграция баз данных; посредники; PRD; мультидиалектная инфраструктура

DOI: 10.14357/1992264140413

Литература


