TRANSFORMATION OF DECLARATIVE RULES TO FORMAL RULES USING FORMAL CONCEPTS

Darius Jurkevičius, Olegas Vasilecas
Vilnius Gediminas Technical University, d.jurkevicius@isl.vgtu.lt

Abstract. Rules are a part of modern information systems. Rules that define or constrain some of the aspects in the application domain are usually written in a declarative form. However, to apply rules in the information system, we need to use formal rules. The article describes the method that allows transforming declarative rules into formal rules. Formal concepts describing rules used in written rules were used for transformation. The suggested model allows input and transforming a rule simply without having any specific skills.

Introduction

Designing information systems and seeking to successfully implement a project, it is important to make cooperation and understanding between businessmen and creators of an information system. A businessman describes the system requirements and constrains using rules or business rules. By Gottesdiener (Gottesdiener 1997), a business rule is ‘a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business’.

According to the principles of rule independence (Business Rules Group 2002) explained in the business rules Manifesto re-
leased by the Business Rules Group release, rules must be declarative: rules must be expressed declaratively in natural-language sentences for the business audience; if something cannot be expressed, then it is not a rule; a set of statements is declarative only if the set has no implicit sequencing; any statements of rules that require constructs other than terms and facts imply assumptions about system implementation; a rule is distinct from any enforcement defined for it. A rule and its enforcement are separate concerns; rules should be defined independently of responsibility for who, where, when, or how of their enforcement; exceptions to rules are expressed by other rules.

When a businessman describes a business rule, then the rule can be described ambiguously and be undefined. We can remove vagueness and ambiguity, if each business rule is resolved into elementary or atomic rules. An atomic business rule is declaratively written using natural language, easily understood by businessmen, and not ambiguous. Information system designers write atomic rules using formal language. During this stage of rule transformation, cross-purposes can occur because both businessmen and system creators have their own languages. Mistakes occurring during the stage of rule transformation process can be removed if a businessman/woman him/herself writes a declarative rule by the template proposed in natural language.

For example, the SVBR specification (OMG consortium 2006) of the OMG consortium is proposed, and therefore we can find information on how to describe business rules using structural natural language. The introduced situation shows that when formulating a sentence we can write different business rules using templates. We can automatically transform a declarative rule into the formal one using templates.
Used Terms

Concept can be defined:

– an abstract or general idea inferred or derived from specific instances (WORDNET 2008);

– A concept is an abstract idea or a mental symbol, typically associated with a corresponding representation in language or symbolism, that denotes all of the objects in a given category or class of entities, interactions, phenomena, or relationships between them (Wikipedia 2008);

– has an intention (deep definition), extension (set of objects or exemplars) (Martin, Odell 1994);

– the definition of a type of objects or events. A concept has an intentional definition (a generalization that states membership criteria), and an extension (the set of its instances) (Mayers, Maulsby 2004);

FCA is based on the philosophical understanding that a concept can be described by its extension—that is all the objects that belong to the concept and its intension which are all the attributes that the objects have in common (Tilley 2003);

Formal Concept Analysis (FCA) (Wille 1982) method is:

– a mathematization of the philosophical understanding of concept;

– a human-centred method to structure and analyze data;

– a method to visualize data and its inherent structures, implications and dependencies.

Formal context is the mathematical structure which is used to formally describe these tables of crosses (or briefly a context) (Wolf 1993).

Process of Rule Transformation

Next we will describe the proposed method of transforming declarative rules into the formal ones (Figures 1 and 2).
Making the rule input in the declarative form, it is suggested to make the input using the template or this rule can be written using semi structured natural language.

Figure 1. Transformation process of the declarative rule

Figure 2. Activity diagram of writing the declarative rule
Using the template input proposed by the system, the initial component of the rule (ex. If) is suggested (Figure 2). Afterwards other terms kept in the formal context are suggested. It is a step by step constructed rule. The mistake can be avoided using the first method, when the templates are suggested, and the rule can be immediately transformed into the formal form. This method is designed mainly for the businessmen. Hereunder we show that different rules have a generic form. Rules can be defined by declarative IF-THEN form with the priority and start and finish dates, for example:

Rule 1:
If a customer is from Vilnius, a DVD will be proposed for free.

Rule 2:
If a customer is older than 65, a ticket will be proposed for free;
Prima facie, these rules do not have much in common. But after detailed analysis, we can find them corresponding to the same template.
If a customer characteristic comparison value, then value commodity will be proposed for free.
If we put this rule in a more common way, it would be:
If exemplar characteristic comparison value, then action value characteristic exemplar.
If we put this rule in the most common way, it would be:
If condition, then action.
There are not so many ways to formulate the condition and action.

The second method is designed to input new terms to write a new kind of rules in the system.

The written declarative rule is transformed, and if there are variances or unknown words, we propose to input this word into the formal context. When the word is saved, the rule transformation is made once again (Figure 1).

When a programmer uses an application program (in edit mode), his/her first steps are creating objects (terms of specific domain, e.g. business). S/he gets terms from the declarative rules and inputs in the
attributes fragment of the SQL code (e.g. another language can be used: RuleML, OCL). They have relationship represented by binary relation. Additionally, they have relationship with other attributes describing characteristics of terms (objects), e.g. verb, noun, characteristic, comparison, value etc. Information about exemplars can be retrieved from databases or kept separately as meta data. Using the terms saved in the formal context, we can transform from a declarative rule into a formal represented rule. Using formal concept analysis, we can represent the formal concepts in concept hierarchical tree. The next stage is demonstration of how one component of the rule can be transformed using formal concepts (Figure 3).

Figure 3 indicates the use of attribute type (ex. SQL_). The type of the attribute needs a separate SQL code from the text (ex. SQL_If and Text_If). The types of the attributes can be used for different types of the tasks such as showing the SQL or RuleML codes, searching for a task, etc.

<table>
<thead>
<tr>
<th>When</th>
<th>Term from declarative rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attr1</td>
<td>Attr2</td>
</tr>
<tr>
<td>If</td>
<td></td>
</tr>
<tr>
<td>When</td>
<td></td>
</tr>
<tr>
<td>Then</td>
<td></td>
</tr>
</tbody>
</table>

Declarative rules input form uses formal context (one part is shown), and formal concept is described: \{If, When, That\}={SQL_If}.

**Figure 3.** Example of rule transformation

The form gets terms and conditions from the formal context. When the declarative rule is transformed into the formal rule, it must be saved.

General function architecture of the system is shown in Figure 4. The functions of the components are:

– declarative rules management component: input of declarative rules using the system; input of declarative rules with no system in use;
– terms input component: management of terms; input new terms;
– database of formal rules context: keeping set of formal rules;
– database of terms of specific domain context: keeping set of terms of specific domain.

Figure 4. Function architecture diagram of the rules transformation system

Storage of the Set of Formal Rules

We propose keeping transforming rules in the formal context (Tables 1 and 2) splitting it into 2 parts: condition and action. This solution allows avoiding information duplicate and keeping the set of rules in the formal context (ex. one action and a few conditions or one condition and a few actions).

Table 1. Saving the declarative rule in informal rule informal context (version 1)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action 1</th>
<th>Action 2</th>
<th>Action n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition 1</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Condition 2</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Condition n</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Saving the declarative rule in formal context (Version 2)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action 1</th>
<th>Action 2</th>
<th>Action n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition 1</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Condition 2</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Condition n</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Discussions

This section deals with debating typical questions that may arise during discussions on the subject presented in this paper.
What are the business cases of the approach? A ‘business case’ for our approach is to reduce using programmers for input business rules into the system. The businessman can him/herself insert the declarative rule into the system using natural language.

Conclusions and Future Work

Currently, the prototype of the mechanism of formal concepts inputting in the formal context is being designed. Planned activities are as follows:

– formulate a logical proposition to compose action and condition;
– make an experiment to verify the proposed method;
– ascertain the best way to save the formal represented rules: when an object is an action and an attribute is a condition or vice versa.

The latter method allows inputting a rule using the proposed templates without requiring specific knowledge. It is suggested to keep formally written rules in the formal context. Keeping a formal concept in the formal context allows representing concepts in the hierarchical order which allows analysing information in the future.

References


DEKLARATYVIŲ TAISYKLIŲ TRANSFORMACIJA Į FORMALIAS, NAUDOJANT FORMALIUS KONCEPTUS

D. Jurkevičius, O. Vasilecas

Santrauka

Šiuolaikiniuose informacinėse sistemose dažnai naudojamos taisyklės. Taisyklės, kurios apibrėžia ar apriboja tam tikrus dalykinės srities veiklos aspektus, neretai užrašomos deklaratyvia forma. Straipsnyje siūlomas metodas, kuris leidžia deklaratyviai užrašytas taisykles transformuoti į formaliai užrašytas taisykles. Šiam tikslui pasiekti naudos formalus konceptai, kurie aprašo terminus bei išraiškas, iš kurių yra sudarytos taisyklės. Siūlomas metodas leidžia paprastai, neturint specialių igūdžių, įvesti ir transformuoti taisyklę.