Compliance Check Software System

Ventsislav Nikolov

Abstract: *In this paper a set of functions for a compliance check language implemented in a limit software system is described. It is intended generally for the financial industry, but it can also be used for any other field requiring similar automatic observation of limits on the allowed operations. The basic system organization as well as the compliance check language constructs are described and clarified.*

Key words: *Compliance Check, Rule Based Software System, Limit, Finance, Portfolio, Sub-portfolio*

INTRODUCTION

In some organizations, companies and institutions some restrictions should be used and observed because of various reasons. This need could naturally emerge because of both common regulations and individual institution strategies. Such restrictions are often used in the financial software systems as a secure measure to control the investments and to have profitable financial portfolios [6]. Some investments should be restricted or forbidden because of their risk or unclear return or profitability. This means that the portfolio making, and operations are not arbitrary. For example, a given institution could prohibit the trade with precious metals or the amount of the government bonds should not exceed 80% of all investments. Building and handling a set of rules helps conforming specific investment strategies. Thus, the specific institution strategies could be adapted to the current environment and financial situation and the risk is taken into account.

In this paper a software solution is described which allows the restrictions to be expressed as limits by a set of rules in a simplified ad-hoc language called Compliance Check Language (CCL). It consists data types to describe portfolios, sub-portfolios and numeric values. A variety of operations can be performed to calculate aggregate values like: sum, average, minimum, maximum and other mathematical functions on the specified fields of the selected objects. The language is flexible enough to create a variety of sub-portfolios by different filters on objects fields' values. The objects in the system are mainly portfolio positions. An example of a positions filter is: only the positions for a given country or given currency or both could be selected to belong to a new sub-portfolio.

The CCL is implemented in a software library which functions can be called by arbitrary other applications. Within a given instance the library needs CCL limits expressions and the main set of positions which compose the data for which the limits should be checked. The limits should be given as a plain text and the positions are expressed as implementations of a given interface defined in the library. As the CCL is translated into production-based language the positions are represented and inserted into the system as production system facts. Thus, the whole input of the system is a simple string of the CCL statements and a set of production facts representing the portfolio positions. The output of the system is obtained by analysing the facts in the working memory of the production system and getting those which represent the limits. As the limits are in fact Boolean expressions, there are data fields in the facts representing the left-hand side and right-hand side of the limits comparisons.

Thus, it is also possible to indicate whether a given limit is not violated but close to violation and in this case a simple warning could be shown to the user.

BASIC SYSTEM ORGANIZATION

Every portfolio is considered as a set of positions. Every position comprises of fields that are properties and attributes with specific data type. For simplicity their definitions are considered the same for all available positions. Thus, the position is represented by a regular object with properties or dynamically calculated values by member functions. The system allows building of sets and subsets of positions and describing rules for the positions properties and attributes or their aggregates. The rules could be very complex including different mathematical formulas on the positions properties satisfying given limitations. It is also possible to express so called dynamic subsets which are created, destroyed and changes dynamically in the working memory. Thus, in some cases it is not possible to determine beforehand the sets and subsets residing in the memory.

The limits are very convenient for implementation by a production based system. Any such a system could be used and in our case, it is a free Java based system called Drools [1]. The whole CCL expressions and statements are parsed using ANTLR [5] and translated into Drools production rules. The positions are loaded from a database and inserted in the working memory as production system facts. Providing logical support to the facts allows the system automatically to changes its logical inferences by adding and removing new facts [2, 3]. This also allows building an alert system that always resides in the memory and every new position leads to necessary local changes and logical inferences. When a new fact is added to the working memory the logical inference is performed only by the production system engine. When the fact is removed the system automatically restores the state before the position addition and if new facts have been added in the left-hand side of the rule they are removed by the system.

CCL also supports so called sections. The main purpose of the sections is the expressions of the CCL to be logically organized in modules according to the regulations they present. The section starts with an opening and finishes with a closing tag. Technically all elements in a section are processed via last-in-first-out (LIFO) organization. The sections can be nested but cannot cross each other. This means that if the section A opens before section B the closing tag of section B must be before the closing of section A.

COMPLIANCE CHECK LANGUAGE (CCL)

When both the positions and the limits expressions are ready to be used the system performs two following stages.

- First the sub-portfolios are created as different subsets from the main set and they are the input of the system. It is possible some of the positions in the main set not to participate in other subset. The subsets compose tree-like structure though sometimes a graph of positions groups can also be used. The tree or the graph structure is created by defining filters for one or more positions values.

  The values of the variables for the primitive data type ‘double’ in the system could be defined by arbitrary mathematical expressions with some positions fields. An example is shown in fig. 1 where the sum of 1.0 + Field_2 is calculated for all positions in sub-portfolio A.

The sum could also be expressed by more than one position fields, e.g: sum(Field_1 * Field_2 - Field_4).

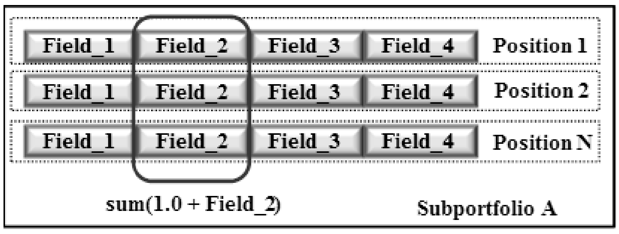

Fig.1 A sub-portfolio and an expression of the positions fields

Both the portfolios and sub-portfolios are represented as sets in the CCL. They can be defined by one of the following ways:

- By another set. In this case an alternative name is given for an existing set.
- By a subset of another set. Here different criteria are defined to select positions of the set and the filters could be combined with logical ‘AND’ and ‘OR’ conjunctions.
- By logical connections like AND, OR and NOT applied to other existing sets as well as brackets on complex expressions.

The values of the primitive data types are expressed by the following aggregate functions on the specified sub-portfolios.

- Set size.
- Average value of a field of the positions for a given set.
- Sum value of a field of the positions for a given set.
- A single value of a field of a position.

- Limits are defined comparing the values from the previous step. The results from the limits comparisons are the logical values TRUE or FALSE. It is possible to define a limit by existing limits using:
  - Another limit. Similarly, to the set the limit definition receives alternative name.
  - Logical connections like AND, OR and NOT applied to other existing limits.

DYNAMIC SETS AND DYNAMIC LIMITS

If a subset should be defined for every country, then dozens of lines code must be written. This situation could be avoided by stating that for every different country a set should be created automatically by the system. The subsets created in this way can be further separated according another field, for example currency. The obtained subsets can be further separated by other fields and as a result of that the tree structure as shown in fig. 2 is created.

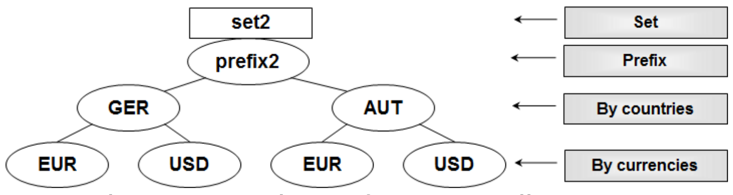

Fig. 2 Dynamic set from an ordinary set

In order to define a dynamic set, an ordinary set must exist. Some of the fields of the positions in the subset must accept enumerable values. Such fields could be for example countries and currencies. A new subset is created for every position whose country does not exist in the positions of already created dynamic subsets. Thus, by inserting or removing positions in the working memory new subsets can be dynamically created or destroyed. The whole tree structure of the dynamic subsets is identified by a string identifier called prefix. It is prefix because every single subset has its own name consisting of the prefix and the path in the tree where every new level is separated by the other with a separator. For example, if the separator is '_' then "prefix2_GER_EUR" is a set corresponding to the left most leaf in the tree in fig. 2.

When dynamic sets are created some limits, checks can be performed on them. For example, for every branch only specified sub-branches can be selected for the check. Such an example is shown in the fig. 3 where for every country only the positions with euro as a currency are checked by comparing calculated aggregate functions.

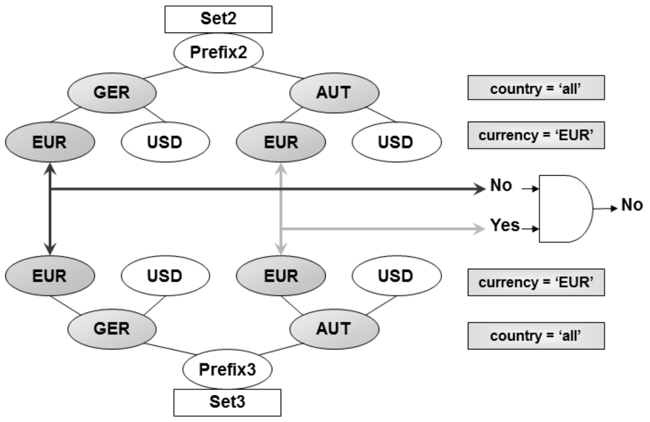

Fig. 3 Limits check on dynamic set

The limits definitions can be more complex if more than one dynamic set is used. In fig. 4 two ordinary sets are separated in the same way in two dynamic sets and in the check the corresponding nodes (sets) are used to calculate aggregate functions. Their results are together used to perform the considered checks.

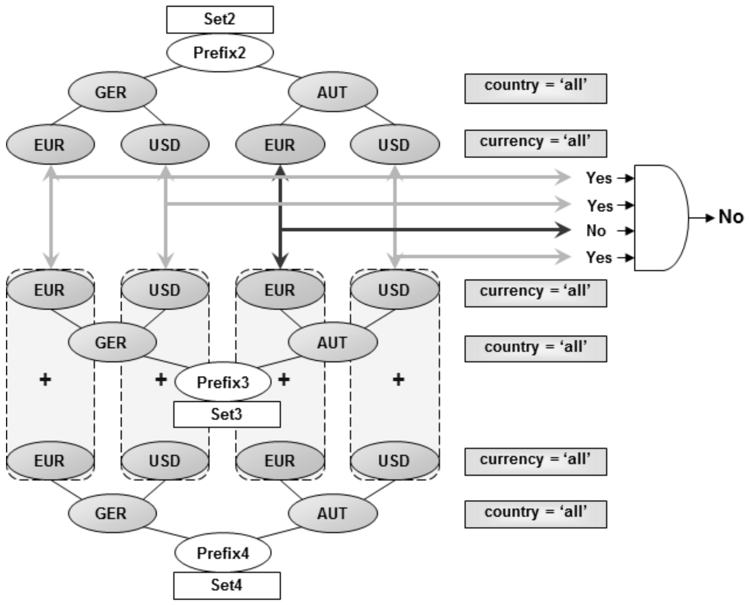

Fig. 4 Aggregate functions for corresponding dynamic sets are used to check a limit. If even one check fails, the whole limit check fails

It is possible the aggregate functions for the nodes of the whole tree to be compared to the corresponding aggregates of another similarly created tree of dynamic sets. The dynamic limits in this case are supposed to be defined in the same way for the sets being used. Otherwise non-existing sets will be searched, and the limit check fails.

It is also possible the dynamic subsets for which the check success to be combined in a new ordinary set as shown in fig. 5. Thus, arbitrary aggregate functions and limit checks may be applied to the new set.

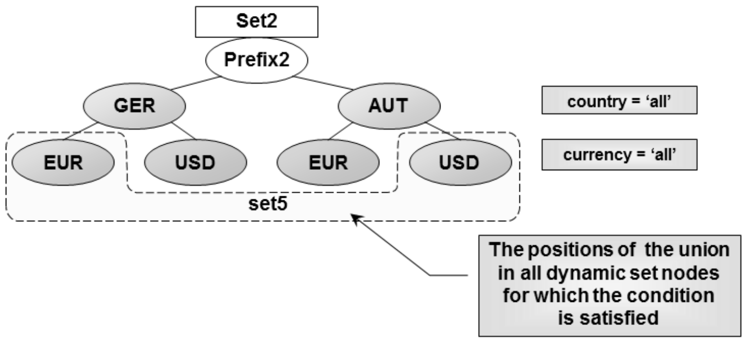
Set2
Prefix2
GER
AUT
EUR
USD
EUR
USD
set5
country = 'all'
currency = 'all'
The positions of the union in all dynamic set nodes for which the condition is satisfied

Fig.5 If a check for dynamic sets succeeds for some of them then they can be merged in a new set

All these possibilities of the CCL are designed to meet the needs of the formal definitions of the operation lows and regulations.

CONCLUSIONS AND FUTURE WORK

The tree based RETE algorithm implemented in the production system used for the CCL implementation leads to effective work of the system when new positions are inserted or removed. Thus, an alert system may be realized allowing a huge number of facts to resist permanently in the memory of the realized system – fig. 6 [4].

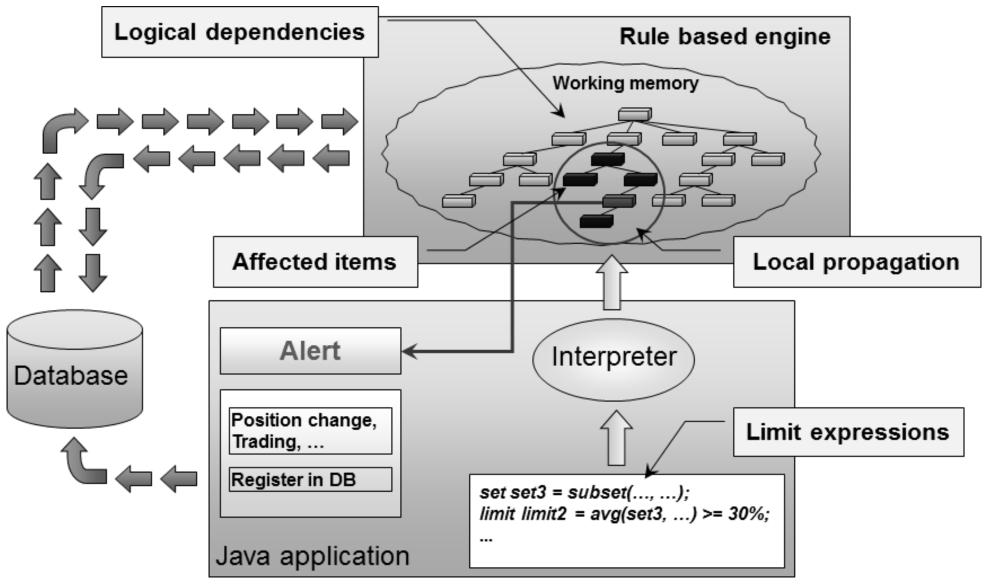

Fig.6 Alert system

The alert system is intended to be a multi-user software realization residing in a server machine. When some change in the database occurs, new positions are added to the working memory and local changes are performed only for the affected items. The other items are not changed, and the inference is done very fast. Similarly, when existing positions in the memory are removed only the facts which have been added in the right-hand side as a result of the matching of the removing facts in the left-hand side of the rule are removed. The other facts are not changed thus making the work of the system very effective.

Acknowledgement: The work presented in this paper was supported within the Start-up Scientific Project № 1, 2015, Technical University of Varna, “Research and development of algorithms for control of mobile robots under extreme conditions in virtual reality”.

REFERENCES

[1] Bali, M. Drools JBoss Rules 5.0 Developer's Guide. Develop rules-based business logic using the Drools platform. Packt Publishing, 2009.

[2] Ligeza, A. Logical Foundations for Rule-Based Systems. Springer Science & Business Media, 2006.

[3] Mayr, H. Database and Expert Systems Applications: Proceedings of the 12th International Conference, DEXA 2001 Munich, Germany, September 3 – 5, 2001.

[4] Nikolov, S., V. Nikolov, A. Antonov. A constraint-based approach for analysing financial market operations. Proceedings of the 14th International Conference on Computer Systems and Technologies, ACM New York, NY, USA, ISBN: 978-1-4503-2021-4, pp. 231 – 238.

[5] Parr, T. The Definitive ANTLR Reference: Building Domain-Specific Languages. Pragmatic Bookshelf, 2007.

[6] Strasberger, M. Risk Limit Systems and Capital Allocation in Financial Institutions. Banks and Bank Systems, Vol. 1, Iss. 4, 2006.

ABOUT THE AUTHOR

Dr. Ventsislav Nikolov
Senior Software Developer
Eurorisk Systems Ltd.
31, General Kiselov Str., 9002 Varna, Bulgaria
E-mail: vnikolov at euroriskSystems dot com