RAMA : a lightweight rule-based tool for expressions analysis and code generation

Vincent Fischer, Loig Allain, Laurent Gerbaud
Laboratoire d'Electrotechnique de Grenoble, CNRS UMR 5529 INPG/UJF
ENSIEG BP 46, 38402 Saint Martin d'Hères, France
vincent.fischer@leg.ensieg.inpg.fr, loig.allain@leg.ensieg.inpg.fr, laurent.gerbaud@leg.ensieg.inpg.fr

Abstract - This paper presents a lightweight tool for mathematical expressions analysis and code generation. This tool, called RAMA (Rule Applicator for Mathematical Analysis) is based on rules written in a XML format. In this way, it is generic and extensible and it can be used for various purposes. RAMA is based on a representation for mathematical expressions, on which rules are applied in order to perform some actions, e.g. : symbolic differentiation.
In a first part, the specifications of the software are presented. In the second part, its architecture and its operating are explained. Then, the representation model of mathematical expressions is presented. The software architecture is the detailed. Finally, some application cases are presented.

1. INTRODUCTION

This paper presents RAMA, a lightweight tool written in JAVA, designed for the analysis and the treatment of mathematical expressions and code generation. This tool can be used for various purposes, from basic mathematical operations such as derivation, to high-level programming tasks like expression splitting.

2. SPECIFICATIONS

RAMA (Rule Applicator for Mathematical Analysis) is a Java tool dedicated to analysis and treatment of mathematical expressions, and code generation. RAMA is built in order to perform complex operations, decomposed into a set of elementary operations, on mathematical expressions. Such operations are named "rules" as RAMA acts like an inference motor.
The specifications of RAMA are then to be:

- lightweight : it has to perform quick operation without taking a large amount of time and of memory. Beside, it has to be easily distributable.
- generic : it has to be able to handle any type of mathematical expressions, and it has to apply every action defined by the user on these expressions. These actions are described by using a "rule" representation.
- easily extensible : new actions can be defined within a relatively short time and without any specific programming knowledge
- easy to use : actions are described in an intuitive way, and applying actions on an expression requires a small amount of instructions.

Symbolic softwares, like Maple [1], Mathematica [2] and Macsyma, etc., offer great possibilities for symbolic treatments. However, they are limited when it comes to generate computation code in several programming languages. In the paper, the aim of RAMA is mainly code generation with simple symbolic treatments, rather than complex ones.

3. SOFTWARE ARCHITECTURE

The RAMA architecture is based on an intern representation of mathematical expressions, and a set of operations to perform loaded from an XML file. In RAMA, these operations are known as "rules".
The intern representation of mathematical expressions is named MOM (which stands for Mathematical Object Model). Basically, a MOM is a tree, which is a classical representation for mathematical expressions [3][4].
RAMA is a "tree walker". By exploring the tree (MOM), rules are selected depending on the current explored structure.
An operation, for example derivation, is described by the user who provides a set of rules describing the operation to perform. These rules are transcriptions of the elementary mathematical actions that represent the result of symbolic treatment. For example, the result for the application of the differentiation operation on the node $\sin(x)$ is $dx \cdot \cos(x)$. This defines an elementary rule for the differentiation operation. These rules are written in a XML format, like shown on Fig. 1 :

```
<RAMA:RULE>
  <RAMA:CONTEXT priority="2">
    <RAMA:MOM name="sin"/>
  </RAMA:CONTEXT>
  <RAMA:RESULT>
    <RAMA:MOM name="*" type="operator">
      <RAMA:apply-ruleset name="differentiate" select="self/child::*"/>
      <RAMA:MOM name="cos" type="function">
        <RAMA:copy select="self/child::*"/>
      </RAMA:MOM>
    </RAMA:MOM>
  </RAMA:RESULT>
</RAMA:RULE>
```

Figure 1: the rule for the differentiation of sinus

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)

ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

An operation is defined by a certain number of rules, which form this operation's rule set. This rule set is coded in an XML format and written in a file (one file per rule set).

The general operating of RAMA is presented in Fig. 2.

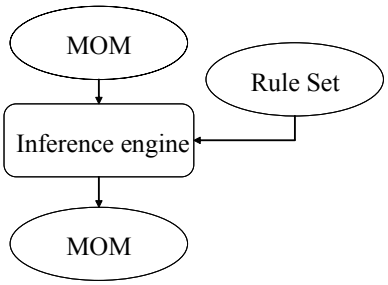

Figure 2: general operating of RAMA

The rule set is applied on a MOM object, and builds another MOM, which represents the result of the application of the operation defined by the rule set.

3.1. The Mathematical Object Model (MOM)

The Mathematical Object Model, or MOM, is a tree representation of mathematical expressions. The tree nodes are the operators, the functions, the variables and the constants of the expressions. A node is defined by :

- a type (operator, function, variable or constant)
- a name, represented by a string reproducing its identity
- some children.

An example of a MOM tree is given in Fig. 3.

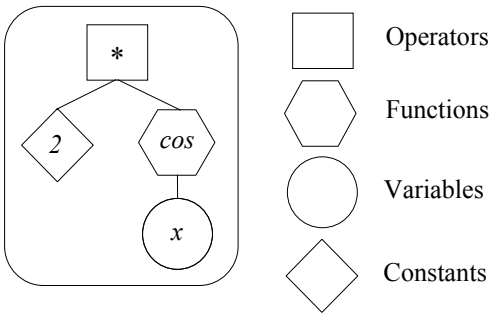

Figure 3: MOM representation of $2\cos(x)$

While nodes representing the operators (e.g. +,-,*,/) are binary ones, nodes representing the sign operators + and – are unary nodes. Nodes representing variables and constants cannot have any child. Finally, nodes representing functions are planar nodes (they can have any number of children). The MOM representation of expressions is rather simple but very efficient for RAMA purposes.

3.2. The inference engine architecture

The principle of the inference engine is similar to the one found in expert systems (see fig 4). While an expert system values action to be performed depending on the current context, the inference engine selects a rule considering the current selected node in a tree (a MOM). A rule is defined by a context and a result. The context is at least a single node, but can be more detailed, with a whole branch of a tree. The result contains the description of the tree to build by the application of the rule on the node which is compliant to the context. This operating is illustrated in Fig. 4.

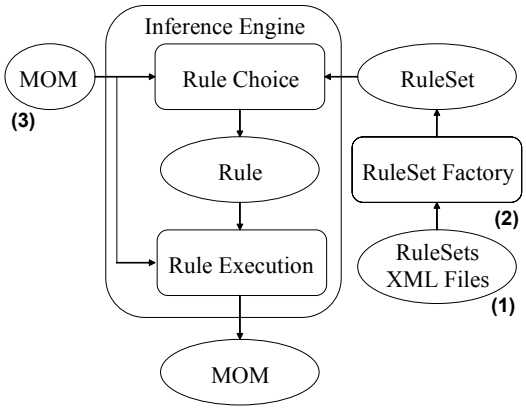

Figure 4: the operating of the rule applicator

The rule sets are described in an XML files (1), to ensure easy extensibility. Classical mathematical operations can be described as a list of rules formed as following :

"*if the object is compliant with <CONTEXT> then the result is <RESULT>*".

It is very convenient to declare actions as a set of rules, in a file. This file is parsed and a "*factory*" (2) instantiates each elementary rule to be used by the inference engine.

Then a MOM is explored by the inference engine. This inference engine selects a rule from the set that complies with this MOM (3), and the associated actions are finally performed. The rule choice strategy is made in two steps :

- in the first step, all the rules applicable to the MOM object are selected
- in the second step, among these ones, the rule with the highest priority is chosen.

As the application of a rule results in an other MOM, this one can be reused by the inference engine to apply another rule set, or to translate the resulting MOM into other formats, such as ANSI-C mathematical expression (C code generation), or to a Java computation code, or to any other language for which a translator has been written.

3.3. The XML Rule Set Format

The rules are gathered together into a rule set. This rule set defines an operation, and its rules are written in a specific XML format. This structure is easy to understand for the user.
A rule set is a file that contains as much rules as it is needed to describe the operation. Each of the rules is composed of two parts:

- an application context : this is the context in which the rules applies. When RAMA is "walking" the tree, it comes up with a node on which it has to apply the operation wanted by the user. This node is compared with all the contexts contained in the rule set. Among the compliant contexts found, the one with the highest priority is chosen. The priority of each context is given by the user in the rule set file.
- the result of the application of the rule on this context, given by its tree representation.

Some basic actions are available to build the result of the application of the rule, like the copy of nodes, the call to the application of a rule set on the selected nodes, etc…

3.4. Rule example

For example, the result of the application of the action differentiation on the node $\tan(x)$ is :

$$dx \cdot (1 + \tan^2(x)).$$

This is an elementary derivation rule. In this rule set for the differentiation operation, the rule defining the differentiation of tangent *("tan")* nodes will be written as shown in Fig. 5.

```
<RAMA:RULE>
  <RAMA:CONTEXT priority="2">
    <RAMA:MOM name="tan"/>
  </RAMA:CONTEXT>
  <RAMA:RESULT>
    <RAMA:MOM name="*" type="operator">
      <RAMA:apply-ruleset name="differentiate" select="self/child::*"/>
      <RAMA:MOM name="+" type="operator">
        <RAMA:MOM name="1" type="constant"/>
        <RAMA:MOM name="pow" type="function">
          <RAMA:copy select="self"/>
          <RAMA:MOM name="2" type="constant"/>
        </RAMA:MOM>
      </RAMA:MOM>
    </RAMA:MOM>
  </RAMA:RESULT>
</RAMA:RULE>
```

Figure 5: the rule for the differentiation of *"tan"*

When a "*tan*" node is encountered by the rule applicator, the result of the application of the derivation is a "*" node with two children :

- the result of the application of the derivation action on the child of the "tan" function node, namely its parameter.
- a "+" node with a child being a "1" constant node, and its other child being a "pow" function node with two children : the copy of the initial "tan" node and the constant node "2".

This result can be illustrated by the following tree (see fig. 6):

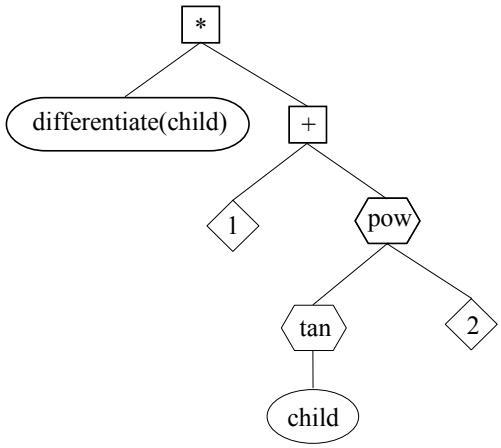

Figure 6 : the tree representation of the differentiation rule of *"tan"*

The path format used for node selection (like for example on the "apply-ruleset" node, to select the node on which the rule set will be applied) is derived from the W3C XPath specification [5], adapted to be specific to the MOM norm.

Finally, the application of an action on an expression consists in applying the dedicated rule set to the tree representation root. With the recursivity shown above (the call to the rule set application in the result of a rule), the symbolic treatment propagates itself along the branches of the MOM tree.

4. APPLICATIONS

At the present time, RAMA can perform various actions.
It is used for code generation purposes. The generated code may be written in C/C++, Java, or any other programming language. This code can be generated for simulation purposes [6], for sizing processes using optimization techniques [7], or any other purpose.
RAMA is also used to split up mathematical expressions into smaller ones, in order to reduce parsing times, and compilation times of the corresponding generated codes (for example in C programming). It is also used to split up complex expressions into their real and imaginary parts. Finally, RAMA also derivates or differentiates mathematical expressions.

4.1. Differentiation

The differentiation of mathematical expressions is based on a rule set containing 25 rules for the operators, the usual mathematical functions, the

variables and the constants. The computing time is equivalent to Maple® one, but the memory occupation is far lesser with RAMA. However, the obtained expressions are not so simplified than with tools like Maple. Mainly, factorization methods are not applied.

4.2. Java and C Code Generation

By defining a dedicated rule set, it is possible to transform the representation of an expression to be compliant with a language. For example, RAMA is used to perform such an operation in a tool called MAEL [6] that produces JAVA and C code for simulation processes and optimization processes.

4.3. Real And Imaginary Parts Of Expressions

RAMA can be used to perform any action that can be described by a set of rules. Another action is the separation of real and imaginary part from a complex expression. At the present time, this functionality is used in MAEL, and is described by 15 rules. Such an operation performed on 224 elementary expressions, has a lower cost than using the functions of MAPLE software as an independent process. However, as for differentiation, the obtained expressions are not so simplified than with tools like Maple.

5. CONCLUSION

At the present time, RAMA is used to performed treatment on mathematical expressions. Extensions to treat other structures which can be represented through a tree, like algorithm, may be also possible. The architecture of this tool is extensible to other activities thanks to the use of structured representations based on XML. Besides rules and actions, it is possible to build checking process, i.e. tests that are performed on a tree (e.g., to value the degree of a polynomial expression).

6. REFERENCES

[1] Maple : http://www.maplesoft.com/
[2] Mathematica : http://www.wolfram.com
[3] J. Davenport, Y. Siret, E. Tournier, "Calcul formel. Systèmes et algorithmes de manipulations algébriques". Editions Masson 1993 275 pp., ISBN : 2 225 84200
[4] M. Gondran and M. Minoux. “Graphes et algorithmes”. Eyrolles, Paris, 3rd edition, 1995.
[5] XPath, http://www.w3.org/TR/xpath
[6] Loïg Allain, Laurent Gerbaud, Ch. Van Der Schaeghe, "Modeling electromechanical actuators for simulation : MAEL performs model capitalization and symbolic treatement", EPE’2003 (European conference on Power Electronics and Applications), Toulouse, France, 2-4 September 2003
[7] Vincent Fischer, Laurent Gerbaud, Jean Bigeon, "Solving ODE for Optimisation : specific use of the Matrix Exponential Approach", OIPE'2002 (Optimization and Inverse Problem in Electromagnetism), Lotz, Poland, 12-14 September 2002