Ministry of Agriculture & Land Reclamation
Agricultural Research Center
Central Lab for Agricultural Expert Systems

XML Representation for a Knowledge Base Structure

TR/CLAES/278/2004.4

By

Eng.Mohammed El Helly
Eng.Mohammed Said

April, 2004

1- Introduction

This document contains a design for a XML Knowledge Base Representation. This representation consists of Ontology, inference, Rule, Function, and table.

The main idea behind proposes such representations using XML is to facilitate the verification processing of the KB using standard representation. It can be easily to add the converting facility to any ES shell like KSR or KROL to transform its own KB representation to this standard XML representation.

This document consists of seven sections. Section two demonstrates an overall structure for knowledge base using XML. Section three presents Ontology representation. The structure of the inference is presented in Section four. Sections five, six, and seven give the design of our KB representation, which is Rules, Function, and Table respectively.

2- Overall Knowledge Base XML Structure

The over all structure of XML knowledge base representation is presented in Fig 1.

```
<KB Name="">
    <CONCEPTS>
        ....List of Concepts
    </ CONCEPTS>
    <INFERENCES>
        ... List of Inferences
    </INFERENCES>
    <RULE_CLUSTERS>
        ... List of Clusters
    </ RULE_CLUSTERS >
    <FUNCTION_CLUSTERS>
        ... List of Function
    </ FUNCTION_CLUSTERS>
    <TABLES>
        ... List of Tables
    </ TABLES >
</KB>
```

Fig. 1 Overall knowledge base structure

In Fig 1, the KB node has an attribute ‘name’, which holds the name of the knowledge base. The knowledge base has a list of concepts, list of inferences, list of rule clusters, list of function clusters, and list of tables. Those lists are represented as a sub node of the KB node.

3 Ontology Representation

The concepts representation is depicted in Fig. 2. Each concept has many attributes and many properties. The concept’s attributes includes:

```
< CONCEPTS >
    <Concept NameL="" NameA="" DescriptionA="" DescriptionL="“
                                SupperName="" SupperNID="" nID="">
        <Poperty NameL="" NameA="" Type="" PromptA=""
                                    PromptL="" Default="" Source="">
                <Legal   NameL="" NameA=""> </Legal>
                    …
                    …
        </Poperty>
            …
            …
    </Concept>
        ..
        ..
</ CONCEPTS >
```

Fig. 2 Concepts Representation

NameL: - The English name of the concept.
NameA: - The Arabic name of the concept.
DescriptionA: - The Arabic description of the concept.
DescriptionL: - The English description of the concept.
SupperName: - The name of the parent concept.
SupperNID: - The ID of the parent concept.

Each property has many attribute and many legal Childs. These attributes include:

NameL: - The English name of the property.
NameA: - The Arabic name of the property.
Type: - The type of the property (integer, real, string, etc.).
PromptA: - The Arabic prompt of the property.
PromptL: - The English prompt of the property.

Default: - The default value of the property.
Source: - The source of value of the property (user, database, relation, etc.).

The legal Childs have two attributes:

NameL: - The English name of the legal value.
NameA: -The Arabic name of the legal value.

4 Inferences Representation

The inferences representation is depicted in Fig. 3. Each inference has an attribute Name that holds the name of the inference and has many steps. Each step has Name attribute that represent the name of this step and the Type attribute that holds the type of this step (cluster, function or table etc.) .

```
<INFERENCES>
    <Inference Name="""">
        <Step Name="" Type="" ></Step>
            …
            …
    </Inference>
        ..
        ..
</INFERENCES>
```

Fig. 3 Inferences Representation

5 Rule Clusters Representation

The rule cluster representation is depicted in Fig. 4. Each cluster has an attribute Name that holds its name. Each clusters has many rules. Each rule has a 'Name' attributes that hold the name of the rule. Each rule has one condition and one action as a child node. The condition has many combination of "OR" and "AND" levels
Each level has four attributes:
1-Cpt : holds the concept name.
2-Prop: holds the property name.
3-Val: holds the value of the property.
4-Op: holds the operator name such as (==, !=, < , >,etc.)

```
<Rule_Clusters>
    <Cluster Name="""">
        <Rule Name="" >
            <Cond>
                <AND>
                    <OR>
                        <Premis Cpt=”” Prop=’’’  Op=””>
                            <Val>
                                <Const></Const>
                            </Val>
                        </Premis>
                        <Premis Cpt=”” Prop=’’’  Op=””>
                            <Val>
                                <SUB>
                                    <Fun><name> </name> </Fun>
                                    <premis cpt=" " prop=" " />
                                </SUB>
                            </Val>
                        </Premis>
                        <Premis Cpt=”Expr”   Op=””>
                            <Expr>
                                <SUB>
                                    <Fun>
                                        <name> </name>
                                    </Fun>
                                    <Premis Cpt=”” Prop=””/>
                                </SUB>
                            </Expr>
                            <Val>
                                <Const></Const>
                            </Val>
                        </Premis>
                        ….
                    </OR>
                </AND>
                        ….
            </Cond>
        <Action>
                <premis cpt=" " prop=" " oper="">
                        <val>
                                <const> </const>
                        </val>
                </premis>
                ….
        </Action>
    </Rule>
    ……
   </Cluster>
    ……
</ Rule_Clusters >
```

Fig. 4 Rule Clusters Representation

The following example represents a simple rules and its transformation.

```
Rule Name Rule2
Rule Cond
        @thrips.hypothesis == "yes"
        && @leaves_observations.color == "spotted"
        &&  @leaves_spots.color == "shiny silver"
        &&  @leaves_spots.position == "lower surface"
Rule Action
        @thrips.confirm="likely";
```

Fig. 4.1 Original Rule Representation (Example 1)

```
<Rule Name="Rule2 ">
   <Cond>
    <AND>
      <AND>
        <AND>
          <premis cpt="thrips "prop="hypothesis " oper="== ">
             <val><const>yes</const> </val>
           </premis>
           <premis cpt="leaves_observations " prop="color " oper="== ">
             <val> <const>spotted</const> </val>
           </premis>
        </AND>
        <premis cpt="leaves_spots " prop="color " oper="== ">
             <val> <const>shiny silver</const> </val>
        </premis>
      </AND>
      <premis cpt="leaves_spots " prop="position " oper="== ">
        <val> <const>lower surface</const> </val>
      </premis>
     </AND>
   </Cond>
 <Action>
       <premis cpt="thrips " prop="confirm " oper="=">
             <val><const>likely</const> </val>
       </premis>
 </Action>
</Rule>
```

Fig. 4.2 XML Rule Representation

Another example, which demonstrate the expression in rule

```
Rule Name Rule6
Rule Cond
        @iron_def.confirm!="none"
        &&@root_knot_nematode.confirm == "none"
        &&@root_lesion_nematode.confirm== "none"
        &&(Now - @plantation.date) > 14
Rule Action
        @obligatory_material.name="Iron_chelate";
        @Iron_chelate.method="foliage nutrition";
        @Iron_chelate.disorder_name="iron_def";
        @Iron_chelate.date = now;
```

Fig. 4.3 Original Rule Representation (Example 2)

```
<Rule Name="r6">
<Cond>
  <AND>
    <AND>
        <AND>
            <premis cpt="iron_def" prop="confirm" oper="!=">
                <val><const>none</const></val>
            </premis>
            <premis cpt="root_knot_nematode" prop="confirm" oper="==">
                <val><const>none</const> </val>
            </premis>
        </AND>
      <premis cpt="root_lesion_nematode" prop="confirm" oper="==">
                <val><const>none</const>  </val>
      </premis>
    </AND>
    <premis cpt="Expr" oper=">">
      <Expr>
            <SUB>
              <Fun><name>Now</name></Fun>
              <premis cpt="plantation" prop="date" />
            </SUB>
      </Expr>
            <val><const>14</const> </val>
    </premis>
  </AND>
</Cond>
<Action>
    <premis cpt="obligatory_material" prop="name" oper="=">
      <val><const>Iron_chelate</const>  </val>
    </premis>
  <premis cpt="Iron_chelate" prop="method" oper="=">
      <val><const>foliage nutrition</const> </val>
    </premis>
    <premis cpt="Iron_chelate" prop="disorder_name" oper="=">
      <val> <const>iron_def</const> </val>
    </premis>
    <premis cpt="Iron_chelate" prop="date" oper="=">
      <val> <Fun> <name>now</name>  </Fun> </val>
    </premis>
 </Action>
</Rule>
```

Fig. 4.4 XML Rule Representation

6 Functions Cluster Representation

The function cluster representation is depicted in Fig. 5. Each cluster has an attribute Name that holds its name. Each clusters has many functions. Each function has a 'Name' attributes that hold the name of the function. Each function has one Output and one Body as a child node.
The Output child contains at most one child node to represent the premise. The body is represented as a mathematical expression. Which is represented as a binary tree. The abbreviation of the arithmetic operators [*, +, -, /, ^] which is used when transform the function are. [MUL, ADD, SUB, DIV, POW] respectively.

```
<FUNCTION_CLUSTERS>
  <CLUSTER Name="">
    <Function Name="">
      <Output>
        <premis cpt=" " prop=" " />
      </Output>
      <Body>
        <ADD>
          <MUL>
          <const></const>
            <const></const>
          </MUL>
          <DIV>
            <MUL>
              <MUL>
                <SUB>
                  <premis cpt=" " prop="" />
                  <premis cpt=" " prop=" " />
                </SUB>
                <const></const>
              </MUL>
              <const></const>
            </MUL>
            <premis cpt=" " prop=" " />
          </DIV>
        </ADD>
      </Body>
    </Function>
  </CLUSTER>
  .....

</FUNCTION_CLUSTERS>
```

Fig. 5 Functions Clusters Representation

The following example represents a simple function and its transformation.

```
Cluster Name : gc_ve_st_f
Function Name: Function 24
Function Output: @gc.value
Function Body:
        100*0.55+(@task_parameter.age-@plant.initial_stage) *100* 0.4 / @plant.ve_stage
```

Fig. 5.1 Original Function Representation

```
<FUNCTION_CLUSTERS>
  <CLUSTER Name="gc_ve_st_f">
    <Function Name="Function 24">
      <Output>
        <premis cpt="gc" prop="value" />
      </Output>
      <Body>
        <ADD>
          <MUL>
            <const>100</const>
            <const>0.55</const>
          </MUL>
          <DIV>
            <MUL>
              <MUL>
                <SUB>
                  <premis cpt="task_parameter" prop="age" />
                  <premis cpt="plant" prop="initial_stage" />
                </SUB>
                <const>100</const>
              </MUL>
              <const>0.4</const>
            </MUL>
            <premis cpt="plant" prop="ve_stage" />
          </DIV>
        </ADD>
      </Body>
    </Function>
  </CLUSTER>
  .....

</FUNCTION_CLUSTERS>
```

Fig. 5.2 XML Function Representation

7 Tables Representation

The table representation is depicted in Fig. 6. Each table has an attribute Name that holds its name. Each table has one Input node and one Output node as well as Rows node as child nodes of that table. Its noted that the input and output child nodes are used to represent the table schema but the Rows child node is used to holds the values stored in that table.

The Input child node contain at least one premise child node, while The Output child node contains at most one child node to represent the premise. It is important to understand that, the number of attributes in the Row node is equal to the number of premise child nodes in the Input node and Output node.

```
<TABLES>
  <Table Name="">
    <Input>
        <premis cpt=" " prop=" " />
        <premis cpt=" " prop=" " />
        ..............
    </Input>
    <Output>
        <premis cpt=" " prop=" " />
    </Output>
    <Rows>
        <Row col1=" " col2=" " ........ out="" />
        ............
    </Rows>
   </Table>
  ........
</TABLES>
```

Fig. 6 Table Representation

The following example represents a table and its transformation.

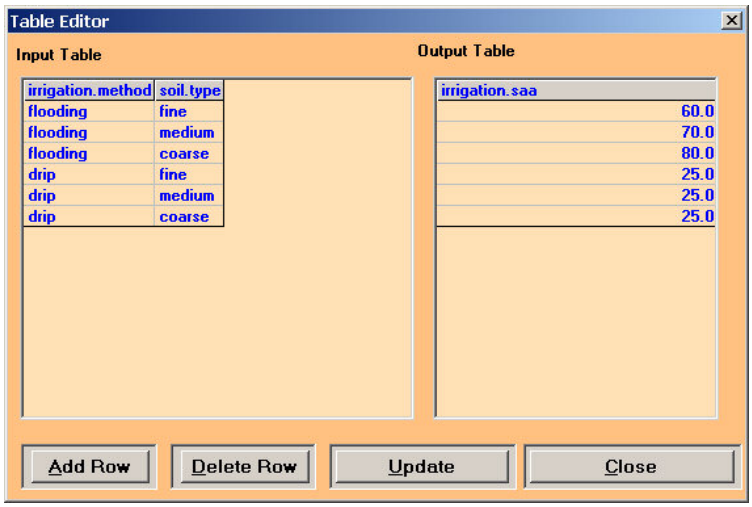

Fig. 6.1 Original Table Representation

```
<TABLES>
  <Table Name="saa_t ">
    <Input>
        <premis cpt="irrigation " prop="method " />
        <premis cpt="soil " prop="type " />
    </Input>
    <Output>
        <premis cpt="irrigation " prop="saa " />
    </Output>
    <Rows>
        <Row col1="flooding " col2="fine " out="60.0 " />
        <Row col1="flooding " col2="medium " out="70.0 " />
        <Row col1="flooding " col2="coarse " out="80.0 " />
        <Row col1="drip " col2="fine " out="25.0 " />
        <Row col1="drip " col2="medium " out="25.0 " />
        <Row col1="drip " col2="coarse " out="25.0 " />
      ..........
    </Rows>
   </Table>
  ........
</TABLES>
```

Fig. 6.2 XML Table Representation