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

Requirement Specifications for Dependency Graph Editor

KSR Tool

TR/CLAES/237/2002.4

By
Dr. Hesham Ahmed Hassam
Eng. Gamal Al-Shourbagy

April 2002

Table of contents

Requirement Specifications for Dependency Graph Editor

1. What is a Dependency graph?

DG is a graph representation that depicts the dependencies that determine the property value. It consists of two main objects, which are nodes and links. The remaining sections would describe the requirements specifications for an editor that facilitates the constructions of this graph.

2. What are the advantages of using this representation?

The following points justify why we have chosen this representation:

a) **Avoid recalculations:** To explain this point we have to consider the following situation. Suppose we have the parameters p1, p2, p3 and p4, used for determine a target value X, and if p1 is *dependent-on* p2, also p3 is dependent-on p4. If during run time the value of p2 only has been modified to relax special constraint(s), then instead of recalculating all the parameters from p1 up to p4 to determine X, we only need to re-determine p1, which gives us the opportunity to get more efficient solution. In short, this representation gives us the opportunity to only determine those parameters that have dependencies on the last modified parameters(s).

b) **Documentation and maintainability:** as we can see, the dependency graph allow the knowledge engineer to clearly represent the knowledge in an easy and straightforward way, therefore, even the domain experts could revise the knowledge. Error detection could also be done with minimum efforts.

c) **Reusability**: Model represented using this method would facilitate its reusability for similar crops.

3. What are the objects specifications?

In general, there are two types of objects as follow:

1. **Node objects**: This object represent an ontology attribute which has one of the following types of source of value:
    a. User or Database "U" or "D"
    b. Table "T"
    c. Function "F"
    d. Relation (Cluster) "R"
2. **Links objects**: This object illustrates that the super-node value depends on the sub-node value. There are two types of links:
    a. Conditional Link
    b. Unconditional Link

4. Description of Dependency Graph Editor (DGE):

As mention "DG" consists of two functions that are Nodes and Links. Each of these functions must be defined.

Node Function: is required to help users in defining their nodes. For each node, we need to define the following:

1. Node name, the attribute name from the ontology (concept. property)
2. Node source name and its type, it will be selected from the list of sources already defined by the knowledge base.
3. Sub nodes, list of sub-nodes names or null.
4. Set of actions, the following action is proposed:
    a. Invocation of value source (value source editor)
    b. Updating Node (Add, Edit, and Delete)
    c. Copy and Paste of the Node.

Hint: the graph will generates a default name of the node value in the form "node name + node type"

Link Function: is the second part of the DGE, we have two types of links in the DG as mentioned in section 3. So, a link function is required to help users in defining their links. For each link, we need to define the following:

1. 'From Node', the name of the node that the link start from it.
2. 'To Node', the name of the node that the link point to.
3. Link Condition, this condition will determine whether the 'To Node' will be visited or not.
4. Set of actions, the following action is proposed:
   a. Updating link (Add, Edit, and Delete)
   b. Copy and Paste of the link.

Hint: Links should have a unique id (Auto id)

Also, there are set of actions to handle the entire graph such as:

1. Preview and print, browsing the graph through as HTML Document, so it may be printed.
2. Copy and Past a sub tree
3. chick the graph connectivity

5. Global verification of the graph.

A valid graph should satisfy the following conditions

1. A graph should start with a node whose source value is derived from Function, Table, or Rule (Cluster). This node represents the root of the graph.
2. A graph should be connected.
3. there is no direct or indirect cycling.
4. there is one and only one link between any two nodes.

6. Graph Evaluation Algorithm (Dependency Mechanism) :

Graph navigation and association mechanism is an important point in this topic. So, a general algorithm that illustrates how to determine the node value would be written here:

Graph Evaluation main function:

Input: root node of the graph like tree

Output: node value

Process:

1. traverse (root node)
2. let list = root node subs
3. for each node in list do
   3.1 if node link condition is true
      3.1.1 traverse (node)
4. process (root node)

Process(node) function:

Input: node

Output: node value

Process:

1. let name = node source name
2. let type = node source type
3. select case(type)
   case table
      run(source name)
      Get from working memory ([in] name,[out] node value)
   case function
      let node value = execute(source name)
   case rule; cluster
      fire(cluster)
      Get from working memory ([in] name,[out] node value)

Hint: The above algorithm also could be called the post order traversal of the tree