A Novel Way of Family Tree Representation and Case Retrieval*

Gi-Chul Yang

Division of Information Engineering, Mokpo National University

***Abstract*— Efficient representation of family tree is essential for any system utilizing information in the family tree. Efficient representation of family tree and retrieval methods, which can handle large number of family trees, are introduced in this paper. It is a modified form of conceptual graph and can be used as a case representation form for a case-based genetic cancer risk assessment system.**

***Index Terms* —** Family Tree, Case-Based Reasoning, Conceptual Graph.

I. INTRODUCTION

A novel way of representing family tree is introduced in this article. The family tree representation method is working efficiently with the system CARA (CAncer Risk Assessment system). CARA is designed based on the *case-based reasoning* (CBR) technology and the backbone structure of the case for CARA is family tree. CBR [Leak96, Kolodner93, Aamodt94] is an AI technique, which relies on the concept that similar problems can be solved as similar cases in the past. In CBR, the basic problem solving process is case retrieval. In the process of case retrieval, relevant cases are retrieved from a database of case called *case base*. Hence, the organization of a case base and the retrieval method are very important in CBR.

There are many systems built based on CBR technique in industries [Althoff95] as well as in medicine. However, they are using different case representation and retrieval methods. Since, their problem domains are different each other. We are adapting CBR for genetic risk assessment of cancer. A family tree with cancer related information is represented as a case for CARA. It is difficult to represent this case as a tree or as a set of attribute/value pairs since each case contains information from many different sources (i.e. members of a family). Hence, the information required by CARA should be organized as a compound structure instead of a tree or linear structure.

Another problem should be considered CBR system is the size of the case base. The CBR system should be able to handle fairly large cases in a case base, so using a commercial DBMS would be a good idea instead of using a file system. CARA utilizes a commercial DBMS to store cases for reliability and large storage size while maintain an efficient retrieval mechanism with it.

The remainder of the paper is organized as follows. The conceptual graph is explained in next section. In section 3 we discuss how the family tree is represented while the indexing mechanism is described in section 4. Section 5 explains matching process and section 6 concludes this paper.

II. CONCEPTUAL GRAPH AND THREE DIMENSIONAL CONCEPTUAL GRAPH

A family tree is a backbone structure of the case for CARA. A case contains a family tree and related information on each individual in the family tree. A case is represented as a modified *Three Dimensional Conceptual Graph* (TDCG) in CARA. TDCG is a modified version of *Conceptual Graph* (CG). Short descriptions of CG and TDCG are in the following two subsections.

Introduction to conceptual graph

The *Conceptual Graph* (CG) is a graph-based logic language which has been developed from the Peirce's idea of EGs and it is more powerful than predicate calculus [Sowa86]. A CG has graph notation like the idea of Peirce's EGs to simplify the rules of logic and can represent higher-order relations. A CG is a finite, connected, bipartite graph. There are two kinds of nodes; concept nodes (displayed as a box in graph notation) which represent entities, attributes, states, and events, and relation nodes (displayed as a circle in graph notation) which represent the relationship among concept nodes. A single concept by itself (Fig. 1) may form a conceptual graph though this is not the case with a relation (Fig. 2), since every relation node should have one or more arcs each of which must be linked to some concept (Fig. 3).

[Cancer]

Figure 1. A single concept

(CAUSE)

Figure 2. A single relation

[Cancer]->(CAUSE)->[Alcohol: 6]

Figure 3. A single relation with two concepts

A CG can be constructed by assembling percepts. In the process of assembly, concept relations specify the role

* This research was supported in part by the Program for the Training of Graduate Students in Regional Innovation which was conducted by the Ministry of Commerce Industry and Energy of the Korean Government

that each percept plays and concepts represent percepts themselves. A concept can be an individual or generic. The function *referent* maps concepts into a generic marker or a set I = {#1, #2, #3, ...} whose elements are *individual markers*. The function *type* maps concepts into a set of *type labels*. A concept c with *type*(c) = t and *referent*(c) = r is displayed as [t : r] in the linear form.

Types of concepts (type labels) are organized into a hierarchy called type hierarchy. Type hierarchy forming operations include conjunction and disjunction operations, so the type hierarchy will be a formal lattice. The type hierarchy constitutes a partial ordering and becomes a type lattice when all the intermediate types are introduced..

A CG can be represented in three different forms. There is a graphic notation called the *display form* (DF), a more compact notation called the *linear form* (LF) as well as a concrete syntax called the *conceptual graph interchange form* (CGIF), which has a simplified syntax and a restricted character set designed for compact storage and efficient parsing. Both DF and LF are designed for communication with humans or between humans and machines. For communication between machines, the CGIF has a simpler syntax. Hence, we will develop an efficient storage and retrieval system for CG's represented in CGIF in this paper. Below descriptions of the three representation forms are adapted from [http://www. bestweb.net/~sowa/cg/ cgdpans.htm #Header_21.].

Figure 4 shows the display form of a conceptual graph that represents the prepositional content of the English sentence *John is going to Boston by bus* .

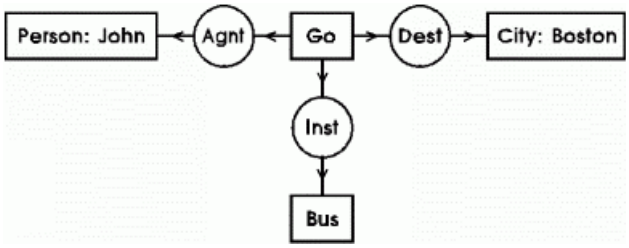

Figure 4. CG Display Form for "John is going to Boston by bus."

In DF, concepts are represented by rectangles: [Go], [Person: John], [City: Boston], and [Bus]. Conceptual relations are represented by circles or ovals: (Agnt) relates [Go] to the agent John, (Dest) relates [Go] to the destination Boston, and (Inst) relates [Go] to the instrument bus.

The linear form for CGs is intended as a more compact notation than DF, but with good human readability. Following is the LF for Figure 4:

```
[Go]-
    (Agnt)->[Person: John]
    (Dest)->[City: Boston]
    (Inst)->[Bus].
```

In this form, the concepts are represented by square brackets instead of boxes, and the conceptual relations are represented by parentheses instead of circles. A hyphen at the end of a line indicates that the relations attached to the concept are continued on subsequent lines. Following is the CGIF for Figure 4:

[Go *x] (Agnt ?x [Person: John]) (Dest ?x [City: Boston]) (Inst ?x [Bus])
or (Agnt [Go] [Person: John]) (Dest [Go] [City: Boston]) (Inst [Go] [Bus])

CGIF is intended for transfer between IT systems that use CGs as their internal representation. Also, CGIF can be translated into different logical languages such as *Knowledge Interchange Format* (KIF). Hence, it is better to use CGIF like notation instead of DF or LF to represent the cases in CARA.

Introduction to three-dimensional conceptual graph

Three-Dimensional Conceptual Graph (TDCG) has three-dimensional shape as in Fig. 6. There is only one difference between conceptual graph and TDCG. TDCG has concept stack instead of concept.

Definition 1 (Concept Stack) Concept stack is a concept node, which can hold multiple concepts in TDCG.

Definition 2 (Three Dimensional Conceptual Graph) TDCG is a conceptual graph which has concept stack(s) instead of concept(s).

The conceptual graph in Fig. 5. represents the meaning of a sentence "Three dimensional conceptual graph for risk assessment".

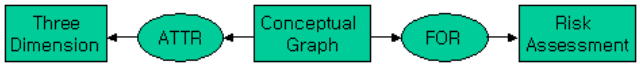

Figure 5. A conceptual graph

For the same sentence, we can draw the following TDCG as in Fig. 6.

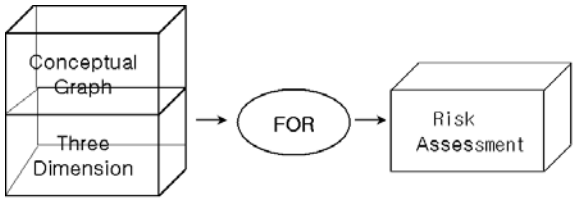

Figure 6. A three dimensional conceptual graph

Both concepts [ThreeDimension] and [ConceptualGraph] in Fig. 1 became one concept stack in Fig. 6. If we look at the TDCG from the top, we can not see the concept [ThreeDimension]. Only we can see is [ConceptualGraph]->(FOR)-> [CancerAssesment], even though there is a concept [ThreeDimension] underneath the concept [ConceptualGraph]. The conceptual relation '(ATTR)' in Fig. 5 is disappeared in Fig. 6. Therefore, we lose some semantic precision but we gain great simplicity in graph

construction and manipulation. It brings great simplicity in the process of automatic graph construction and make possible to represent the information in three dimensional graph structure [Yang98]. The TDCG in Fig. 6 is represented in modified CGIF as follows.

(FOR {[ThreeDimension][ConceptualGraph]}
{[CancerAssesment]})

It is always true that the number of conceptual relations in TDCG is less than or equal to the number of conceptual relations in corresponding conceptual graphs. Therefore the TDCG matching process is simpler than the CG matching process. Also, the concept stack in TDCG can hold more information then the concept in CG. These characteristics are well fit to our purpose.

III. FAMILY TREE REPRESENTATION

A case for CARA contains a family tree and related information on each individual node in the family tree. A case is represented as a modified *Three Dimensional Conceptual Graph* (TDCG) in CARA.

How a family tree in Fig. 7 is represented as a TDCG in modified CGIF is shown below.

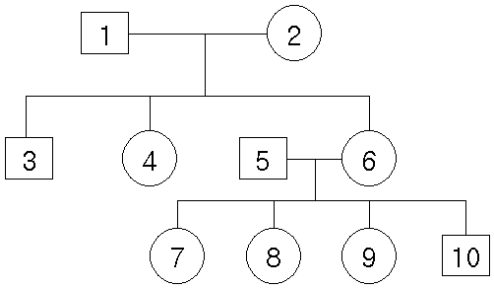

Figure. 7 A Family Tree

A family tree in Fig 7 can be converted as a tree in Fig 8. The converted tree is a *sibling tree* shown below.

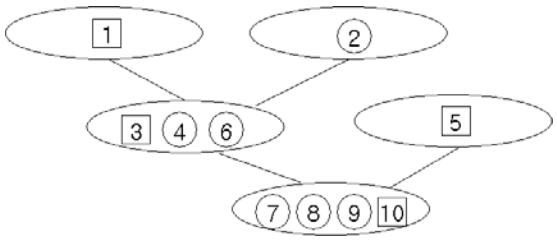

Figure. 8 A Sibling Tree

Each node in a sibling tree represents a set of all sibling nodes in the same level as one node. After converting the family tree into a sibling tree, available tree matching algorithms such as [Wang94, Wang98, Zhang94] can be applied to find the similar trees and common substructures.

Even though the sibling tree is not enough for case representation in case of CARA it can be used for preprocessing of the family tree matching that is necessary for speed up the matching process.

If the relation nodes are inserted in the middle of the arcs then the sibling tree became a conceptual graph like family tree as in Fig. 9.

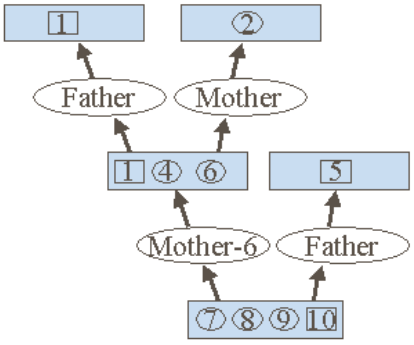

Figure. 9 A conceptual graph like family tree

There are two kinds of relation nodes Father and Mother in Fig. 9. The relation node with the label 'Mother-6' indicates that the node ⑥ in the (sibling) concept node [1 4 6] is the mother of the (sibling) concept node [7 8 9 10].

In order to represent all the information needed for CARA, we are using modified TDCG to represent the family tree. A family tree in Fig 7 is represented in modified TDCG as follows:

G-1: /*This is a graph id */
(Mother {[Cancer:?][Sibling:1101]}{[Cancer:Breast][Sibling:1001]})
(Father {[Cancer:?][Sibling:1101]}{[Cancer:NA][Sibling:0000]})
(Mother{[Cancer:Breast] [Sibling:1001]}{[Cancer:NA][Sibling:0000]})
(Father {[Cancer:Breast] [Sibling:1001]}{[Cancer:NA][Sibling:0000]})

All the nodes in the family tree are not appeared as a concept stack in the corresponding TDCG for simplicity. Sibling nodes are encoded and embedded as a type of a concept. There are four numbers in the sibling concept. They represent (number of sisters with cancer, number of sisters without cancer, number of brothers with cancer, number of brothers without cancer) respectively. Therefore, [Sibling:1101] stands for someone has one brother and two sisters and one of his/her sister has cancer. Fig. 10 depicts the encoding of the sibling concept.

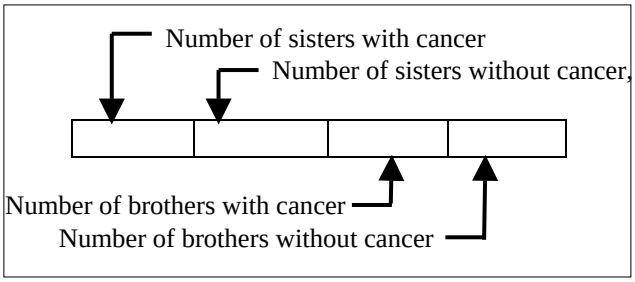

Figure. 10 Encoding of the sibling concept

Here are two other example cases represented in modified CGIF for TDCG. The cases are:

G-2:
(Mother {[Cancer:?][Sibling:1101]}{[Cancer:Breast][Sibling:1001]})
(Father {[Cancer:?][Sibling:1101]}{[Cancer:NA][Sibling:0000]})
(Mother {[Cancer:Breast][Sibling:1001]}{[Cancer:Breast][Sibling:0000]})
(Father {[Cancer:Breast][Sibling:1001]}{[Cancer:NA][Sibling:0000]})

G-3:
(Mother {[Cancer:?][Sibling:0101]}{[Cancer:NA][Sibling:0000]})
(Father {[Cancer:?][Sibling:0101]}{[Cancer:NA][Sibling:1001]})
(Mother {[Cancer:NA][Sibling:0000]}{[Cancer:NA][Sibling:0000]})
(Father {[Cancer:NA][Sibling:0000]}{[Cancer:NA][Sibling:0000]})

G-2 depicts the family tree of the patient who has two sisters and one brother. Among the siblings of the patient there is one sister with cancer. The patients' mother has one brother who does not has cancer and the grandparents are cancer free. There is no information available about siblings of the grandparents in this case.

An efficient indexing mechanism is necessary for a sophisticated retrieval approach. Section 4 will present an efficient indexing mechanism.

IV. INDEX STORAGE

As we can see here CGIF is organized with parenthesis and each parenthesis contains one conceptual relation. Hence, the relation name could be a table name for indexing in a database. When the first case is coming into CARA, the unique graph id called G-id is attached to the coming graph and stored in *case base* and the structure of the Index Storage (IS) will be

Mother :=
(1 {[Cancer:?][Sibling:1101]}{[Cancer:Breast][Sibling:1001]})
Father := (1 {[Cancer:?][Sibling:1101]}{[Cancer:NA][Sibling:0000]})
Mother-2 :=
(1 {[Cancer:Breast][Sibling:1001]}{[Cancer:NA][Sibling:0000]})
Father-2 :=
(1 {[Cancer:Breast] [Sibling:1001]}{[Cancer:NA][Sibling:0000]})

Now we have four lists, named Mother, Father, Mother-2 and Father-2 in the IS with one element (e.g., (1 {[Cancer:?][Sibling:1101]}{[Cancer:Breast][Sibling:1001] }) in the Mother list) in each list. Since, the first graph has four relations, four lists are created in an empty IS. The first number in each item is the graph id (called G-id). Mother-2 and Father-2 are the relation of grandmother and grandfather. Those relations are represented as Mother and Father in the graph, but they are changed to Mother-2 and Father-2 for the efficient case retrieval.

The list has been used as a data structure for the IS, since it is simple and easy to explain the basic concept of the IS. Actual implementation should be done via a more efficient data structure. We suggest extendable hashing for fast retrieval and a relational database for large knowledge bases. We used a database for the developed system. After the second case is accepted, the IS becomes

Mother := (1 {[Cancer:?][Sibling:1101]}{[Cancer:Breast][Sibling:1001]})
(2 {[Cancer:?][Sibling:1101]}{[Cancer:Breast][Sibling:1001]})

Father := (1 [Cancer:?][Sibling:1101]}{[Cancer:NA][Sibling:0000]})
(2 {[Cancer:?][Sibling:1101]}{[Cancer:NA][Sibling:0000]})

Mother-2 := (1 {[Cancer:?][Sibling:1101]}{[Cancer:NA][Sibling:0000]})
(2{[Cancer:Breast][Sibling:1001]}{[Cancer:Breast][Sibling:0000]})

Father-2 := (1 {[Cancer:?][Sibling:1101]}{[Cancer:NA][Sibling:0000]})
(2 {[Cancer:Breast][Sibling:1001]}{[Cancer:NA][Sibling:0000]})

We explained IS by using a data structure *list*, however, as we mentioned earlier it can be a table in a relational DBMS.

V. CASE RETRIEVAL

In this section, we show how the cases can be retrieved in the case base. The basic access mechanism is matching, in which a query representation is matched to representations of cases in the case base. A novel CG matching technique (partial and exact matching) was introduced in [Yang93].

A query match can be performed through the IS. CARA has the ability of exact matching and partial matching. For example, if the first case

G-1
(Mother {[Cancer:?][Sibling:1101]}{[Cancer:Breast][Sibling:1001]})
(Father {[Cancer:?][Sibling:1101]}{[Cancer:NA][Sibling:0000]})
(Mother {[Cancer:Breast] [Sibling:1001]}{[Cancer:NA][Sibling:0000]})
(Father {[Cancer:Breast] [Sibling:1001]}{[Cancer:NA][Sibling:0000]})

is accepted as a query then the query graph is separated into (relation concept) pairs. The first step is take the first pair from the query and search the IS. The first (relation concept stack) pair (i.e. (Mother {[Cancer:?] [Sibling:1101]} {[Cancer:Breast] [Sibling:1001]})) is picked and find that the relation is 'Mother' so search the 'Mother' table for the concept stack in the pair (i.e. {[Cancer:?][Sibling:1101]} {[Cancer:Breast][Sibling:1001]}). CARA will find two matched items. Take the G-ids of the matched items, 1 and 2. Next, take the second pair (i.e. (Father {[Cancer:?] [Sibling:1101]}{[Cancer:NA][Sibling:0000]})) from the query and perform the same procedure. We got the same matched items (i.e. G-id 1 and 2) with the previous results for the second pair. Each time any new elements are found then the G-ids of those elements (i.e., G-id 1 and 2 in this case) are intersected with the G-ids of old elements. There is no difference in this case but for the third pair, there is only one matched item G-1 since grandmother of G-2 has

Breast cancer while G-1's grandmother has no Breast cancer. Now, the result of the intersection between (G-1, G-2) and G-1 is (G-1). After perform the same task for all the pairs in the query we will find exact matching result that is G-1 in this case.

In the case of partial matching, CARA can retrieve similar cases from the case base. For example, if we want to retrieve the cases that the patient has at least one sister with cancer and don't care any other people in the family tree then look at the patient's sibling concept which has type number greater than 1000. Patient's sibling concept is [Sibling:1101] and it's type number is 1101. Various different partial matching can be performed by CARA depends on the constraints presented by the user. It is similar to a constraint satisfaction process. Followings are few examples of partial matching cases;

- Find the cases that the patient has a mother who has more than 50% of her sisters with a breast cancer.
- Find the cases that the patient has both mother and grandmother who has at least one of their siblings has cancer.
- Find the cases that the patient has a father who has brother with cancer.

VI. CONCLUSION

We found that ordinary tree structure is not a suitable approach to illustrate the accurate family tree, since the structure of the family tree is different with the structure of the ordinary tree. An Efficient representation of family tree as well as the indexing and retrieval mechanism has been described in this paper. It can be used for medical information system as well as other application systems.

REFERENCES

[1] Aamodt, A. 1994. Relating case-based reasoning: foundational issues, methodological variations and system approaches, AI Commun. 7(1) pp. 39-59.

[2] Althoff, K.-D. et al. 1995. A Review of Industrial Case-based Reasoning Tools, AI Intelligence, Oxford.

[3] Althoff, K.-D. et. al 1998. Case-based reasoning for medical decision support tasks: The Inreca approach, AI in Medicine 12. pp. 25-41.

[4] Jurisica, I. et. al 1998. Case-based reasoning in IVF: prediction and knowledge mining, AI in Medicine 12. pp. 1-24.

[5] Kolodner, J.L. 1993. Case-based Reasoning, Morgan Kaufmann, San Mateo.

[6] Leak, D. (Ed.). 1996. Case-Based Reasoning: Experiences, Lessons, and Future Directions, AAAI Press, Menlo Park, CA.

[7] Poole, J., J.A. Campbell. 1995. A novel algorithm for matching conceptual and related graphs, Conceptual structures : applications, implementation and theory, Eds. G. Ellis et al., New York, (LNAI 954).

[8] Rau, L. 1988. Exploring the semantics of conceptual graphs for efficient graph matching. In Proceedings of the 3rd Annual Workshop on Conceptual Graphs, Boston.

[9] Roberts, D.1973. The Existential Graphs of Charles S.Peirce, Mouton, The Hague,.

[10] Sowa, J. 1984. Conceptual Structure: Information Processing in Mind and Machine, Addison Wesley, Massachusetts.

[11] Yang, G-C., Y. Choi, & J. Oh. 1993. CGMA : A novel conceptual graph matching algorithm, Conceptual structures : theory and implementation, Eds. H.D. Pfeiffer, T.E. Nagle, New York, (LNAI 754)

[12] Yang, G-C., 1998. Three Dimensional Conceptual Graph for Document Retrieval, ICCS98, Daegu, Korea.

[13] http://www.bestweb.net/~sowa/cg/cgdpans.htm# Header_21.

[14] http://www.bestweb.net/%7Esowa/cg/cgdpans.htm

[15] Wang, J., et. al., 1994. A System for Approximate Tree Matching, IEEE Transactions on Knowledge and Data Engineering, 6(4):559-571.

[16] Wang, J., et. al., 1998. An Algorithm for Finding the largest approximately common substructures of two trees. Ieee Transactions on Pattern Analysis and Machine Intelligence, 20(8):889-895.

[17] Zhang, K., et. al., 1994. Approximate Tree Matching in the Presence of Variable Length Don't Cares, Journal of Algorithms, 16(1):33-66.