Towards the Easy Manipulation of Graph-Based Content Representation of Multimedia Data

Teruhisa Hochin
Division of Information Science, Kyoto Institute of Technology
Goshokaidocho, Matsugasaki, Sakyo-ku, Kyoto-shi, Kyoto 606-8585, Japan
E-mail: hochin@kit.ac.jp

Abstract: The contents of multimedia data has complex relationships including deeply nested whole-part and the many-to-many relationships. A data model incorporating the concepts of directed graphs, recursive graphs, and hypergraphs has been proposed for representing the contents of multimedia data. This data model, an instance is represented with a directed recursive hypergraph called an instance graph. This paper studies on the characteristics of instance graphs. The depth of an edge of an instance graph is introduced. When the depth of an edge is equal to zero, the instance graph can be decomposed into sub-instance graphs. Decomposing instance graphs could make their treatment easy.

1. Introduction

In recent years, handling multimedia data stored in databases has extensively been investigated. Content retrieval of multimedia data is included in the topics on handling multimedia data. One approach of addressing to this issue uses feature values of multimedia data. For example, when a picture is given as a desired one, similarity between the picture and the one in a database is calculated by using feature values. Pictures having high scores of similarity are presented to the user as the query result. Another approach uses graphs representing the contents of multimedia data. Jaimes has proposed a data model representing the contents of multimedia by using four components and the relationships between them[1]. The contents of video data is represented with a kind of tree structure in XML[2]. These are examples of this approach. Directed labeled graphs are frequently used in this approach. This paper follows this approach.

A graph-based data model called Directed Recursive Hypergraph data Model (DRHM) has been proposed[3]. This data model incorporates the concepts of directed graphs, recursive graphs, and hypergraphs. An instance graph is the fundamental unit in representing data. A collection graph is a graph having instance graphs as its components. A shape graph of a collection graph represents the structure of the collection graph. An instance graph representing the contents of a piece of multimedia data may become large. The more complex the contents of a piece of multimedia data is, the more nodes and edges an instance graph has. Easy handling of a large instance graph is required.

This paper shows the characteristics of instance graphs for the purpose of easy manipulation of a large instance graph. The depth of an edge is introduced in order to identify whether the instance graph can be decomposed into sub-instance graphs. When the depth of an instance graph is equal to zero, the instance graph can be decomposed into sub-instance graphs. Decomposing instance graphs could make their treatment easy.

This paper is organized as follows: In Section 2, DRHM is informally described by using examples. Section 3 describes the structural aspect of DRHM. Section 4 gives the considerations on the characteristics of instance graphs. Lastly, Section 5 concludes this paper.

2. Descriptive Examples

In DRHM, the fundamental unit in representing data is an instance graph. An instance graph is a directed recursive hypergraph. An instance graph has a label composed of its identifier, its name, and its data value.

Here, DRHM is described by using an example.

Example 1 Consider the representation of the picture shown in Figure 1(a). In this picture, a butterfly is on flowers. Two fore-legs, one middle-leg, and two hind-legs of the butterfly as well as a head, a fore-wing, and a hind-wing appear in the picture. Two fore-legs are on a flower, and two hind-legs are on another flower. Figure 1(b) represents the contents of this picture in DRHM. In Figure 1(b), an instance graph is represented with a round rectangle. For example, n111, n112, g1, g11, and g12 are instance graphs. An edge is represented with a curve which is consisted of a broken curve and a dotted one. A broken curve surrounds a set of initial elements of an edge. A dotted one surrounds a set of terminal elements of an edge. For example, n111 and n112 are connected to n113.
by the edge e11. When a set of initial elements of an edge contains only one element, and that of terminal elements also contains only one element, the edge may be represented with an arrow for simplicity. The edge e16 in the instance graph g12 is an example of this representation. An instance graph may contain instance graphs, and edges. For example, g1 contains g11, g12, e13, and e14.

A set of the instance graphs having similar structure is captured as a collection graph. A collection graph is a graph whose components are instance graphs.

**Example 2** An example of a collection graph is shown in Fig. 2. A collection graph is represented with a dashed dotted line. A collection graph has a unique name in a database. The name of the collection graph shown in Fig. 2 is Picture. The instance graph g1 is the one shown in Fig. 1. The instance graph g2 is another picture. These instance graphs are called representative instance graphs.

The structure of a collection graph is represented with the graph called a shape graph.

**Example 3** Figure 3 shows the shape graph for the collection graph Picture shown in Fig. 2.

### 3. Formal Definition

#### 3.1 Instance graphs

The instance graph is the fundamental unit in representing data in DRHM. An instance graph is a directed recursive hypergraph.

**Definition 1** An instance graph g is an octuple \((V, E, L_v, L_e, \phi_v, \phi_e, \phi_{\text{connect}}, \phi_{\text{comp}})\), where V is a set of instance graphs included in \(g\), \(E\) is a set of edges, \(L_v\) is a set of labels of the instance graphs, \(L_e\) is a set of labels of the edges, \(\phi_v\) is a mapping from the set of the instance graphs to the set of the labels of the instance graphs \((\phi_v : V \rightarrow L_v)\), \(\phi_e\) is a mapping from the set of the edges to the set of the labels of the edges \((\phi_e : E \rightarrow L_e)\), \(\phi_{\text{connect}}\) is a partial mapping representing the connections between sets of instance graphs \((\phi_{\text{connect}} : E \rightarrow 2^{L_v} \times 2^{L_e})\), and \(\phi_{\text{comp}}\) is a partial mapping representing the inclusion relationships \((\phi_{\text{comp}} : V \cup \{g\} \rightarrow 2^{L_v \cup L_e})\). A label is a triple \((d_{id}, nn, d)\), where \(d_{id}\) is an identifier, \(nn\) is a name, and \(d\) is a tuple of a data type and a data value.

This definition does not inhibit that \(V\) is an empty set. In this case, the instance graph represents a node rather than a graph. This instance graph may only store a data value in its label. In the above definition, an instance node, which is a node rather than a graph, is not introduced. Instead of introducing an instance node, we permit \(V\) to be an empty set. This is because we let instance graphs have flexible structure. If an instance node was introduced, the change of the kind from an instance node to an instance graph will be required. This will result in the low flexibility of the structure as well as the complex management of instance graphs.

From here on, the notations shown in Table 1 are used.

<table>
<thead>
<tr>
<th>notation</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i(e))</td>
<td>the set of initial elements of an edge (e)</td>
</tr>
<tr>
<td>(t(e))</td>
<td>the set of terminal elements of an edge (e)</td>
</tr>
<tr>
<td>(V(g))</td>
<td>the set of instance graphs included in an instance graph (g)</td>
</tr>
<tr>
<td>(E(g))</td>
<td>the set of edges included in an instance graph (g)</td>
</tr>
<tr>
<td>(\text{id}(l))</td>
<td>the identifier in a label (l)</td>
</tr>
<tr>
<td>(\text{name}(l))</td>
<td>the name in a label (l)</td>
</tr>
<tr>
<td>(\text{val}(l))</td>
<td>(d) in a label (l)</td>
</tr>
</tbody>
</table>

As the instance graph has recursive nature, some terms on the construction of an instance graph are introduced.

**Definition 2** Let \(g = (V, E, L_v, L_e, \phi_v, \phi_e, \phi_{\text{connect}}, \phi_{\text{comp}})\) be an instance graph. A set of the \(n^{th}\) constructing elements \(V^n_{\text{ce}}(v)\) for an instance graph \(v \in (V \cup \{g\})\) is defined by the following recursion formula:

- \(V^0_{\text{ce}}(v) = \{v\}\)
- \(V^{i+1}_{\text{ce}}(v) = \bigcup \phi_{\text{comp}}(p), p \in V^i_{\text{ce}}(v) \cap (V \cup \{g\}) (i \geq 0)\)

Here, \(n\) is called the order of constructing elements. Instance graphs (edges, respectively) that are the \(n^{th}\) constructing elements of \(v\) are called the \(n^{th}\) constructing instance graphs (edges) of \(v\). The first constructing elements of \(v\) are called direct constructing elements of \(v\).

**Example 4** The direct constructing elements of the instance graph g1 shown in Fig. 1 are g11, g12, e13, and e14. The direct constructing instance graphs of g1 are g11 and g12.
3.2 Collection graphs

A collection graph is introduced to capture a set of instance graphs.

Definition 3 A collection graph is a graph having
name of instance graphs as its components. That is, when a collection graph \( cg \) has \( n \) instance graphs:
\[ g_i = (V_i, E_i, V_{i1}, V_{i2}, \ldots, \phi_{vi}, \phi_{ei}, \phi_{conn}, \phi_{comp}, \{i = 1, \ldots, n\}), \]
l, \( g_i \) is the label of \( g_i \), \( \phi_{gi}(g_i) = l_{yi} \), and \( \phi_{comp_{g_i}}(cg) = \{g_1, \ldots, g_n\} \), \( cg \) is represented with a 9-tuple \((nm, V, E, L_v, L_e, \phi_v, \phi_e, \phi_{conn}, \phi_{comp})\), where \( nm \) is the name of a collection graph, \( V = \{g_1, \ldots, g_n\} \cup V_1 \cup V_2 \cup \cdots \cup V_n \), \( E = E_1 \cup E_2 \cup \cdots \cup E_n \), \( L_v = \{l_{g_1}, \ldots, l_{g_n}\} \cup L_{v1} \cup L_{v2} \cup \cdots \cup L_{vn} \), \( L_e = L_{e1} \cup L_{e2} \cup \cdots \cup L_{en} \), \( \phi_v : V \rightarrow V_e \) is the union of the mappings \( \phi_{v1}, \ldots, \phi_{vn} \), and \( \phi_e : E \rightarrow L_e \) is the union of the mappings \( \phi_{e1}, \ldots, \phi_{en} \), \( \phi_{conn} : E \rightarrow 2^L_v \times 2^L_e \) is the union of the mappings \( \phi_{conn1}, \ldots, \phi_{connn} \), and \( \phi_{comp} : V \cup \{cg\} \rightarrow 2^V \times 2^E \) is the union of the mappings \( \phi_{comp1}, \ldots, \phi_{compn} \), and \( \phi_{comp_{cg}} \).

Each component instance graph \( g_i \) is called a representative instance graph. A database is a set of collection graphs. The name of a collection graph must be unique in a database.

3.3 Shape graphs

A shape graph is introduced to represent the structure of instance graphs in a collection graph. The instance graphs having the same name are mapped to a shape graph, whose name is that of the instance graphs. The edges in a collection graph, which have the same name, whose initial elements are mapped to the same shape graphs \( sg_{init1}, \ldots, sg_{initn} \), and whose terminal elements are mapped to the same shape graphs \( sg_{term1}, \ldots, sg_{termn} \), are mapped into a shape edge, whose name is that of the edges, whose initial elements are the shape graphs \( sg_{init1}, \ldots, sg_{initn} \), and whose terminal ones are the shape graphs \( sg_{term1}, \ldots, sg_{termn} \).

Creating a shape graph and updating it are driven by the insertion and/or the modification of instance graphs. A shape graph does not have to exist prior to the creation of a collection graph. This could make a database flexible.

Definition 4 The structure of a shape graph is the same as that of a collection graph. That is, it is represented with a 9-tuple. An edge in a shape graph is called a shape edge. The labels of shape graphs and shape edges are different from those of collection graphs. The label of a shape graph or shape edge is a triple \((sg_{id}, nm_{id}, DT)\), where \( sg_{id} \) is an identifier, \( nm_{id} \) is a name of a shape graph or shape edge, and \( DT \) is a set of data types of data values. This label is called a shape label. There are the following relationships between a collection graph \((nm_{cg}, V, E, L_v, L_e, \phi_v, \phi_e, \phi_{conn}, \phi_{comp})\) and its corresponding shape graph \((nm_{sg}, V_s, E_s, L_{v_s}, L_{e_s}, \phi_{v_s}, \phi_{e_s}, \phi_{conn_s}, \phi_{comp_s})\).
- \( nm_{cg} = nm_{sg} \)
- \( nm(L_{e_s}) \supseteq nm(L_e) \)
- \( nm(L_{v_s}) \supseteq nm(L_v) \)

There is a mapping \( \theta_v : V \rightarrow V_s \) such that \( \forall v \in V \exists v_s \in V_s((nm(\phi_v(v)) = nm(\phi_{v_s}(v_s)) \land \theta_v(v) = v_s)) \).

There is a mapping \( \theta_e : E \rightarrow E_s \) such that \( \forall e \in E \exists e_s \in E_s((nm(\phi_e(e)) = nm(\phi_{e_s}(e_s)) \land \theta_e(e) = e_s)) \), and \( \phi_{conn}(e) = (U, W) \Rightarrow \phi_{conn_s}(\theta(e)) = (\Theta_s(U), \Theta_s(W)) \), where \( \Theta_s(U) \) means a set of shape graphs \( \{\theta_v(v_1), \ldots, \theta_v(v_n)\} \) for a set of instance graphs \( U = \{v_1, \ldots, v_n\} \), and
- \( \phi_{comp}(v) = U \cup Z \Rightarrow \phi_{comp_s}(\theta(e)) = \Theta_s(U) \cup \Theta_s(Z) \), where \( \Theta_s(Z) \) means a set of shape edges \( \{\theta(e_1), \ldots, \theta(e_n)\} \) for a set of edges \( Z = \{e_1, \ldots, e_n\} \).

A shape graph has to be changed in order to satisfy the conditions described above when new instance graphs are inserted, or instance graphs are modified.

Figure 3 is a shape graph of the collection graph shown in Fig. 2. This shape graph includes three shape graphs: picture, object, and parts. For every instance graph shown in Fig. 2, there is a corresponding shape graph. There are three shape edges in Fig. 3. The names of these edges are the same one: \( \text{pos} \). However, the initial element and the terminal one are different from one another. The shape edge whose initial and terminal elements are the shape graph object, which is the right edge in Fig. 3, corresponds to the edge e21 shown in Fig. 2.

4. Characteristics of Instance Graphs

Here, we investigate the characteristics of instance graphs. The discussions are limited to simple instance graphs because of simplicity. The simple instance graph is the instance graph whose edge has only one instance graph as its initial element, and only one instance graph as its terminal element. The following discussions can naturally be extended to other kinds of instance graphs.

When any elements outside an instance graph are not directly connected to any elements in the instance graph by any edge, the instance graph could be represented separately from the other elements outside it. That is, the instance graph is represented as if it had no elements, and it and its elements are represented apart from the whole instance graph. An instance graph could be decomposed into a set of smaller instance graphs. This may enable us to handle instance graphs easily similar to the decomposition approaches in storing complex objects[4]. An instance graph \( g \) is said to be well-organized if any elements outside it are not directly connected to any elements in it by any edge.

Definition 5 Let \( VC \) be a set of the constructing instance graphs of an instance graph \( g \). The instance graph \( g \) is called well-organized if the set of the initial elements of an edge is a subset of \( VC \), and the set of the terminal elements of the edge is a subset of \( VC \) for all of the edges that have an element in \( VC \) as an element in the set of their initial elements or terminal ones.

Next, the depth of an edge in an instance graph is introduced in order to decide whether the instance graph is well-organized, or not.
Definition 6 Let \( g \) be an instance graph that has an edge \( e \) as a direct constructing edge. The smallest number \( n_{\text{init}} \), where \( i(e) \subseteq \bigcup_{m=1}^{n_{\text{init}}} V^m_{\text{ec}}(g) \), is called the depth of the initial element of the edge \( e \) in \( g \). Similarly, the smallest number \( n_{\text{term}} \), where \( i(e) \subseteq \bigcup_{m=1}^{n_{\text{term}}} V^m_{\text{ec}}(g) \), is called the depth of the terminal element of the edge \( e \) in \( g \). The largest depth of the initial and the terminal elements is called the depth of the edge \( e \) in \( g \).

Example 5 The edge \( e11 \) in the instance graph \( g1 \) shown in Fig. 1 is a direct constructing edge of the instance graph \( g11 \). The set of initial elements of \( e11 \) is \{\( n11, n112 \)\}. This set is a subset of the set of the direct constructing instance graphs of \( g11 \). That is, \( i(e11) \subseteq V^1_{\text{ec}} \). Therefore, the depth of the initial element of \( e11 \) is equal to zero. Similarly, the depth of the terminal element of \( e11 \) is equal to zero. As a result, the depth of the edge \( e11 \) is equal to zero. On the other hand, the edge \( e13 \) is a direct constructing edge of \( g1 \). The set of the initial elements of \( e13 \) is a subset of \( V^1_{\text{ec}} \cup V^2_{\text{ec}} \) because \( i(e13) = \{n114, n115\} \). Therefore, the depth of the initial element of \( e13 \) is equal to one. Similarly, the depth of the terminal element of \( e13 \) is equal to one. As the result, the depth of the edge \( e13 \) is equal to one.

The fact that the depth of an edge \( e \) is equal to \( k (k \geq 0) \) means that a \((k+1)^{\text{th}}\) constructing instance graph of the instance graph \( v \), which has the edge \( e \) as a direct constructing edge, is an initial or terminal element of the edge \( e \). The edge \( e \) crosses \( k \) boundaries of instance graphs. When \( k \) is equal to zero, that is, the depth of an edge \( e \) is equal to zero, the edge \( e \) does not cross any boundaries of instance graphs. Based on these considerations, the depth of an edge can be used in judging whether an instance graph is well-organized, or not.

Theorem 1 An \( n^{\text{th}} \) constructing instance graph \( g_n \) of a well-organized instance graph \( g \) is well-organized if one of the following conditions is satisfied for every \( k (1 \leq k \leq n) \).

1. \( g_n \) and its constructing elements are neither initial elements nor terminal ones of the \( k^{\text{th}} \) constructing edge of \( g \).
2. If \( g_n \) or one of its constructing elements is an initial or terminal element of a \( k^{\text{th}} \) constructing edge of \( g \), its depth is at most \( n - k \).

(Proof.) First, the case that all of the constructing edges of \( g \) are the \( k^{\text{th}} \) constructing edges is studied. When \( g_n \) and its constructing elements are neither the initial elements nor the terminal ones of the \( k^{\text{th}} \) constructing edge of \( g \), there is apparently no \( k^{\text{th}} \) constructing edge that has \( g_n \) or its constructing elements as the initial or terminal elements. As the \( k^{\text{th}} \) constructing edge is not the edge that has \( g_n \) or its constructing elements as the initial or terminal elements, \( g_n \) is well-organized.

When \( g_n \) or one of its constructing elements is an initial (terminal, respectively) element of a \( k^{\text{th}} \) constructing edge of \( g \), and its depth is at most \( n - k \), the set of the initial (terminal) elements of the edge is a subset of \( \bigcup_{i=1}^{n-k+1} V^i_{\text{ec}}(g) \) based on the definition of the depth of the initial (terminal) element. The \( m^{\text{th}} \) \((m \geq 0)\) constructing elements of \( g_n \) are the \((n+m)^{\text{th}}\) constructing elements of \( g \). Here, the condition \( n-k+1 \leq n+m \) is always satisfied because \( n-k+1 \leq n \) and \( n \leq n+m \). The equality is held under the condition \( k = 1 \) and \( m = 0 \). In this case, \( g_n \) is a direct (first) constructing element of \( g \). That is, \( g_n \) is an initial or terminal element of the edge. In the other cases, the inequality is held. When \( m > 1 \), the instance graphs are the constructing elements of \( g_n \). Therefore, there is no \( k^{\text{th}} \) constructing edge that has the constructing elements of \( g_n \), as the initial or terminal elements. From these discussions, \( g_n \) is well-organized.

When the conditions described above are held for every \( k (1 \leq k \leq n) \), no constructing edge of \( g \) has the constructing elements of \( g_n \) as the initial or terminal elements. As \( g \) is well-organized, \( g \) and its constructing elements are neither the initial nor the terminal elements of the edges outside \( g \). Therefore, \( g_n \) is well-organized.

The theorem is proved.

When a representative instance graph \( g \), which is well-organized, is given, well-organized constructing instance graphs of \( g \) can be obtained by analyzing the constructing elements of \( g \) according to their order.

5. Concluding Remarks

In this paper, the contents of a piece of multimedia data is represented with a kind of directed graph called an instance graph. As there are a lot of objects and relationships between them in a piece of multimedia data, an instance graph, which represents its contents, becomes large. This paper introduced the depth of an edge. This could show whether an instance graph can be decomposed or not. When an instance graph can be decomposed into several pieces of instance graphs, treatment of an instance graph becomes easy. The depth of an edge could become an indicator of easiness of the treatment of a large instance graph.

Development of the efficient storage structure for a database based on DRHM is a subject of future work. The efficient query processing is another subject of future work.

Acknowledgement

This work is supported in part by the Ministry of Education, Science, Sports and Culture of Japan under a Grant-in-Aid for Scientific Research (B), No. 20300037, 2008-2010.

References


