The Hierarchical Representation of Binary Relations: Application to the Design of Class Hierarchies

Amina YAHIA

ACSA Laboratory, Department of Mathematics and Computer Science
Faculty of Sciences, Mohammed 1st University
BP 717, BD Mohammed VI, 60000 Oujda, Morocco

Copyright © 2015 Amina YAHIA. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

This work is about formalizing the hierarchical nature of binary relations. We introduce a new representation for binary relations called the inheritance graph of the binary relation and denoted by H(R). We also present the concept of Isa hierarchy for abstracting class hierarchies. It is both formal and general. Indeed, no assumption is made about the underlying data model. Every where a binary relation holds, its corresponding inheritance graph may have potential benefits. These may be tremendous with regard to the wide range of applications and the correctness of the formal representation.

Keywords: Binary relation, Inheritance graph, DAG, Graph/Order homomorphism/isomorphism, Partially ordered set, Class/Isa hierarchies

1 Introduction

We position ourselves in the context of data and knowledge representation. More precisely, we are interested in the design of class or conceptual hierarchies. The question is how to handle the very difficult and crucial problem of designing such hierarchies.
Among the approaches for constructing class hierarchies, we can find conceptual clustering methods and taxonomic reasoning [2] [3], classification and learning methods [1]. We can also find approaches which are devoted to specific computer domains such as database design [4] [5] and software engineering [6] [7].

Nevertheless what method is used or what application is considered, the general representation of a class hierarchy is that of a directed acyclic graph (DAG). This opens doors for formalization without being obliged to specify a particular data model.

2 Example

Let us consider a very simple example, meant to ease understanding. The universe of discourse is composed of a set of students (the objects) and a set of modules. A student s1 can register for a course unit, say u1, or not. For example, student s1 is registered to attend for the following course units: u1, u3, u4 and u6. Each pair (si,uj) has the value ‘1’ if student si has registered for module uj. The value of (si,uj) is ‘0’ if si has not registered for uj. This binary relation can be represented by its matrix representation as follows:

<table>
<thead>
<tr>
<th></th>
<th>u1</th>
<th>u2</th>
<th>u3</th>
<th>u4</th>
<th>u5</th>
<th>u6</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>s2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1. The matrix representation for the binary relation between students and course units.

The binary relation can also be represented by its bi-partite graph as shown in figure1. Now, we would like to have an idea about how students are organized knowing the modules they attend to. The inheritance graph is an answer to this questions.

3 The inheritance graph of a binary relation

Let R be a binary relation defined on C x P and let f: C \rightarrow 2^P be the function defined by: \forall x \in C, f(x) = \{ y \in P | x R y \}. We assume that f is surjective.

Let us denote by \( \bar{C} \) the partition of the equivalence classes of C. We recall that the equivalence class of x is denoted by or \( \bar{x} \) or \([x]\) and is defined by: \( \bar{x} = [x] = \{ x' / f(x') = f(x) \} \).
**Definition 1.** We call inheritance graph of $R$, and denote by $H(R)$, any directed acyclic graph which satisfies the following predicates:

- The nodes of $H(R)$ are the elements of $\tilde{C}$ and eventually some other virtual elements denoted by $\Delta_j$, $j=0..k$. Each element $\Delta_j$ is fully determined by $\text{IP}(\Delta_j)$, which is the subset of its immediate predecessors in the graph. The following predicate holds: $f(\Delta_j) = \bigcap f(w)$ where $w \in \text{IP}(\Delta_j)$.

- An edge $(x,y)$ belongs to $H(R)$ if and only if:
  
  $f(x) \supseteq f(y)$ and $\not\exists z \in \tilde{C} / f(x) \supset f(z) \supset f(y)$.

- Each edge $(x,y)$ in $H(R)$ is labeled by $f(x)-f(y)$. This labeling may be empty, denoting the empty set.

- All edge labels are disjoint. The union of all edge labels is $f(C)$.

- All nodes are different with regard to $R$: For any two nodes $x$ and $z$, $f(x)$ and $f(z)$ are different.

- The graph has one root node, either original or virtual, denoted by $v_0$.

Back to our example, we can intuitively determine $H(R)$ shown in figure 2. Compared with the bipartite graph shown in figure 1, the inheritance graph of the binary relation seems to be a very interesting alternative. It may appear to be difficult to draw but our experience tells us that ease is gained through practice.

![Figure 1. Bipartite graph of $R$.](image1.png)  
![Figure 2. Inheritance graph of $R$.](image2.png)

### 4 Application to the design of object database class hierarchies

It is widely admitted that a class hierarchy is to be represented as a DAG where nodes are class names and the edges do represent the direct sub-class/super-class links.

In the object database data modeling, the syntactic characterization of the isa relationship is expressed in terms of conformity with the sub-typing relationship [8]. The sub-typing relationship is defined between types. Types are associated
with classes. The isa (or sub-class/super-class) relationship and the sub-typing relationship are partial order relations. Let us denote by $\sigma$, the function which associates types to class names. When, this $\sigma$ is an order homomorphism (defined below, see footnote), we say that the sub-typing is in conformity with the isa relationship.

The concept of isa hierarchy is an abstraction of that one of class hierarchy. It allows us to look at class hierarchies independently from the model within which they are defined. Let us consider the following definitions.

### 4.1 Class hierarchy

**Definition 2.** A class hierarchy, denoted by $H$, is defined as $H=(C, \prec)$ where:

- $C$ is a set of class names and $\prec$ is a strict partial order on $C$ such that the supremum of $C$ exists.

- There exists a set of attribute names, say $P$, and a function $f: C \to P$ such that $f$ is an order homomorphism from $(C, \prec)$ to $(f(C), \subseteq)$.

Let us consider the above example, with a little adaptation. Consider the following table:

<table>
<thead>
<tr>
<th></th>
<th>$u_1$</th>
<th>$u_2$</th>
<th>$u_3$</th>
<th>$u_4$</th>
<th>$u_5$</th>
<th>$u_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 2.** An extension of the binary relation shown in table 1.

In this table, each student $x_i$ is attending to a subset of course units denoted by $f(x_i)$. Figure 3 shows the corresponding class hierarchy.

---

1. Let $(P, \preceq_P)$ et $(Q, \preceq_Q)$ be any two posets. Some function $f: P \to Q$ is called (order) homomorphism from $P$ to $Q$ if and only if: $\forall x, x' \in P, x \preceq_P x' \Rightarrow f(x) \preceq_Q f(x')$. 

Hierarchical representation of binary relations

Figure 3. Graphical representation of the class hierarchy as a DAG.

Hence, given some class hierarchy, say H. By setting: \( xR_H \Leftrightarrow y \in f(x) \), for each \( x \) in C, we are defining a binary relation denoted by \( R_H \), which is associated with H. Thus, we can make the following statement: Any class hierarchy defines one binary relation and only one. But, the reverse is not true in general.

Nevertheless, we can isolate a category of class hierarchies which are thoroughly defined through yielding the binary relation between classes and their attributes. Let us call them binary hierarchies. Note that such terminology is a clear divergence from what is widely admitted in computer science where a binary graph is defined by the cardinality of internal (non leaf nodes) which is set to exactly two.

In the following sections, we give the formal definitions for isa hierarchies and binary isa hierarchies (or binary relation hierarchies). We also define the inheritance graph of an isa hierarchy and introduce some vocabulary and some propositions which are related to inheritance.

4.2 Isa hierarchy

Let \( C_{\text{names}} \) and \( P_{\text{names}} \) be two finite sets of symbols. The two sets are meant to represent class names and property (and/or attribute) names respectively.

Definition 3. An isa hierarchy is a triplet \( H=(C, \leq, f) \) such that the following holds:

- \( C \subseteq C_{\text{names}} \) and \((C, \leq)\) is a partial order having a supremum element.
- \( f: C \rightarrow P_{\text{names}} \) is an order homomorphism from \((C, \leq)\) to \((f(C), \supseteq)\).

Notations:

\[
\gamma x = \{ x' \in C / x' = x \}^2,
\]

\[
[x] = \{ x' \in C / f(x') = f(x) \}.
\]

Let us make the following precision. Equivalent classes and equal classes are not the same thing. Two (or more) equal classes belong to the same node in the

\[\text{We recall that}: \quad x = x' \Leftrightarrow (x \leq x' \text{ and } x' \leq x).\]
graph which represents H, i.e. its Hasse diagram. In the other side, two equivalent classes could be equal or different (in H). However, equality and equivalence within classes become the same thing when we deal with binary isa hierarchies (which are defined in the following paragraph).

This definition tells us that isa hierarchies are partial ordering, with a supremum element, and for which the following proposition holds:

\[(H) \quad \forall (x,x') \in C \times C, \text{ we have: } x' \leq x \Rightarrow f(x') \supseteq f(x).\]

We use the notation (H) to refer to this condition which holds by definition for isa hierarchies. (H) stands for “The Homomorphism condition”.

Using object oriented terminology, this can be expressed as: for each pair \((x, x')\) of classes, if \(x'\) is a subclass of \(x\) then each attribute which is related to \(x\) is also related to \(x'\). However, the reverse does not necessarily hold.

### 4.3 Binary isa hierarchy

**Definition 4.** An isa hierarchy \((C, \leq, f)\) is said to be a binary isa hierarchy if and only if \(f\) is an order isomorphism.

This definition makes the focus on special isa hierarchies, which are called binary isa hierarchies, and for which the following proposition holds:

\[(I) \quad \forall (x,x') \in C \times C, \text{ we have: } x' \leq x \iff f(x') \supseteq f(x).\]

We use the notation (I) to refer to this characterization of binary isa hierarchies. (I) stands for “The Isomorphism condition”.

Using object oriented terminology, this can be expressed as: for each pair \((x, x')\) of classes, \(x\) is a super-class of \(x'\) if and only if each of \(x\) is also an attribute of \(x'\).

The following two propositions are equivalent.

**Proposition 1.** A binary isa hierarchy \(B=(C, \leq, f)\) defines one binary relation and only one denoted by \(R_B\) and defined on \(C \times f(C)\) by:

\[x R_B y \iff y \in f(x), \text{ for any } x \in C.\]

**Proposition 2.** A binary isa hierarchy is an isa hierarchy such that:

\[(E) \quad \forall x \in C, \quad \gamma x = [x].\]

\[(S) \quad \forall x, x' \in C, \quad f(x) \supseteq f(x') \iff x \ll x'.\]

In proposition 2, (E) stands for “Equality and Equivalence” and (S) stands for strict ordering.

The (E) condition isolates an important characterization of binary hierarchies: equality and equivalence become the same. Remember that equivalence and equality
Hierarchical representation of binary relations

relationships within isa hierarchies are different things. The first one is defined relatively to the \((\leq)\) order, the second one is defined according to \(f\).

The \((S)\) condition expresses inclusive semantics for inheritance such as the one given in [9]. Binary hierarchies can be automatically constructed since they are fully defined by attribute subset inclusion. In the present study, we are not concerned with the algorithmic aspect of the problem. This may be addressed in a separate paper. Mathematical proofs were intentionally removed for the ease of presentation. This aspect should also be considered separately.

4.4 Inheritance graph

Any isa hierarchy \(H=(C,\leq,f)\) can be represented by a graph denoted by \(G(H)\) which is defined by the triplet \((\gamma C,\prec f)\) where \((\gamma C,\prec)\) designates the Hasse diagram of the partial order \((C,\leq)\) and where each node \(\gamma x\) is labeled by \(f(x)\).

Because of the homomorphism proposition, the graph of an isa hierarchy contains much redundancy from the point of view of node labels \((i.e.\ those\ attributes\ which\ are\ associated\ with\ classes)\). This is the reason why using an inheritance graph is useful.

The following examples show us how such inheritance graphs can be gradually generated. Despite the fact that this presentation is intuitive, it shows that it relies on strong formal and well defined concepts.

\[
\begin{align*}
f(x_0) &= \emptyset & \sigma(x_0, \emptyset) &= \emptyset \\
f(x_1) &= \{u_1,u_3,u_4,u_6\} & \sigma(x_1,x_0) &= \{u_1,u_3,u_4,u_6\} \\
f(x_2) &= \{u_1,u_2,u_3\} & \sigma(x_2,x_0) &= \{u_1,u_2,u_3\} \\
f(x_3) &= \{u_4,u_5,u_6\} & \sigma(x_3,x_0) &= \{u_4,u_5,u_6\}
\end{align*}
\]

Figure 4. The class hierarchy as a DAG.

This figure is quite simple. The emphasis is made on function \(f\) which holds the entire information about the associated binary relation. More precisely, it is important to introduce a new function denoted by \(\sigma\) and called the labeling function. The latter is a labeling function for edges which is devoted for the capture of inheritance of attributes (or properties) amongst classes.

\[
\forall (x_i,y_j) \in G, \quad \begin{cases} 
\sigma(x_i) = f(x_i), & \text{for } x_i = x_0. \\
\sigma(x_i,y_j) = f(x_i) - f(y_j), & \text{for all } x_i \neq x_0.
\end{cases}
\]

Note that, in figure 4, functions \(f\) and \(\sigma\) happen to be the same. This is because there exist only one node from which inheritance could be induced and this node happens to have nothing to be inherited \((\emptyset)\).
Figure 5. Class hierarchy with a new virtual element $\Delta_0$.

Figure 5 shows the graphical representation of the class hierarchy (in figure 4) after the insertion of a new virtual element $\Delta_0$ as the immediate parent of elements (or classes) $x_1$ and $x_2$.

![Class hierarchy diagram](image)

<table>
<thead>
<tr>
<th>$x_0$</th>
<th>$\Delta_0$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>${u_1, u_3}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${u_4, u_6}$</td>
<td>${u_4, u_6}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

Figure 6. Class hierarchy with another new virtual element $\Delta_1$.

Figure 6 shows the graphical representation of the class hierarchy (in figure 5) after the insertion of a new virtual element $\Delta_1$ as the immediate parent of elements (or classes) $x_1$ and $x_3$.

![Class hierarchy diagram](image)

<table>
<thead>
<tr>
<th>$x_0$</th>
<th>$\Delta_0$</th>
<th>$\Delta_1$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${u_1, u_3}$</td>
<td>${u_1, u_3}$</td>
<td>${u_1, u_3}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${u_4, u_6}$</td>
<td>${u_4, u_6}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${u_2}$</td>
<td>${u_5}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${u_4, u_5, u_6}$</td>
<td>${u_4, u_5, u_6}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

Now, let us go back to binary relations and their inheritance graphs. By labeling the edges of the graph in figure 6 (using the $\sigma$ function), it appears that this graph is exactly the same as the inheritance graph shown in figure 2.

4.5 Binary relation

We have shown that the inheritance graph is an alternative representation for a binary relation.

The following formula gives the expression of function $f$ in terms of the information which is captured (or represented) in the inheritance graph.
Let \( G \) be the inheritance graph of some binary relation, then:

\[
\forall x_i \in G, \quad \begin{cases} 
  f(x_i) = \sigma(x_i) & \text{for } x_i = x_0, \\
  f(x_i) = \bigcup_{(x_i, y_j) \in G} \left( f(y_j) \oplus \sigma(x_i, y_j) \right) & \text{for all } x_i \neq x_0.
\end{cases}
\]

5 Conclusion

This work suggests a new graph for representing binary relations. The new graph which is called inheritance graph of the binary relation is illustrated through a simple example. The bridge is made with the application domain of object database schema design. It is shown how easy the specification of the problem of designing the inheritance hierarchy becomes.

Future work should address, but is not limited to, the following aspects:

- Validation through comprehensive example.
- Computational aspects.
- Further investigation into theoretical aspects.
- Comparison with existing methods and algorithms.

Acknowledgements. This work is dedicated to the memory of my father Sidi Abderrahmane YAHIA (1937-2014). May his soul rest in peace through the infinite mercy of God.

References


Engineering - CAISE’ 93, 8-11 June 1993, Paris, France. C. Rolland, F. Bodart and C. Cauvet (Eds.). LNCS n° 685, (993), 54 - 70. http://dx.doi.org/10.1007/3-540-56777-1_4


Received: January 21, 2015; Published: April 6, 2015