A Framework Design of Dataset Relation Discovery
for Solving Inconsistencies by Connection of Heterogeneity in the Big Data Era

Takafumi Nakanishi, Kiyotaka Uchimoto, and Yutaka Kidawara

Online Handwritten Chemical Expression Recognition

Peng Tang, Siu Cheung Hui, and Chi-Wing Fu

Semantic Specialization: Specialization Using Explicit Specification and Semantic Information

Teruhisa Hochin and Hiroki Nomiya

Adapting CRM Systems for Mobile Platforms: An MDA Perspective

Federico Améndola and Liliana Favre
A Framework Design of Dataset Relation Discovery for Solving Inconsistencies by Connection of Heterogeneity in the Big Data Era

Takahumi Nakanishia, Kiyotaka Uchimotoa, Yutaka Kidawaraa
National Institute of Information and Communications Technology, Japan

Abstract

We highlight the inconsistencies of past research on the connections among such heterogeneous fields as Linked Data, Semantic Web, Bridge Ontology, and Schema Mapping, as well as our own past researches. Graph structures are commonly represented as links in relationships. For the same domain, relationships agree with each other in the domain because the transitive and order relations are defined. However, in most heterogeneous domains, we have to define the new order relation to link heterogeneous sets. This limitation exists when we consider the relation among heterogeneous fields in set theory. There are three inconsistencies of linking heterogeneous resources: 1) where the relation does not guarantee the future; 2) where no transitive relation is true when anyone connects links for heterogeneous fields; and 3) where no relation in heterogeneous fields can be discovered in set theory. Closed assumption systems have already reached their limits. We call this problem the “Three Opened Assumption’s Inconsistencies,” which is the limit through which we consider the relation between heterogeneous fields in set theory. In the big data era, we must consider a new framework for the Three Opened Assumption’s Inconsistencies. As a solution, we propose a map transformation method from set theory to the Cartesian coordinate system to interconnect these heterogeneous sets and the Three Opened Assumption’s Inconsistencies using two easy mathematical proofs of transitive and order relations to interconnect the heterogeneous resources. In addition, we define a new functional predicate as an example of map transformation from set theory to a Cartesian coordinate system to interconnect the heterogeneous resources for our solution. We also define a correlate function as an example of this framework.

Keywords: association discovery method, inconsistencies of linking heterogeneous resources, open assumption, Three Opened Assumption’s Inconsistencies, mathematical proofs.

1. Introduction

Information science researchers have already constructed data sensing, aggregation, retrieval, analysis, and visualization environments via web portals, software, application programming interfaces (APIs), and more on the web. However, the exponential growth in the amount of field data means that revolutionary measures are needed for data management, analysis, and accessibility. Online databases have become crucial for accessing and publishing various data depending on the user’s purpose, task, or interest. Science and sensor data, including such human sensors as Twitter, are increasing rapidly with keywords and phrases like "data intensive science" in this big data era. The difficulty arises because information resources are not well organized. In fact, currently, information resources either exist independently or are connected by static links that do not describe the significance of the connections. Most search engines still rely on keywords from users as queries, and provide retrieval results by keyword matching and a long list of fragmentary information topics. Users still need to input keywords and repeatedly browse retrieval results for understanding.

This situation illustrates the failure to achieve a system that integrates and represents various data on the web where the information on various fields is stored and accessed based on user demands. There are three inconsistencies of linking heterogeneous resources: 1) where the relation does not guarantee the future; 2) where no transitive relation is true when anyone connects links for heterogeneous fields; and 3) where no relation in heterogeneous fields can be discovered in set theory. Closed assumption systems have already reached their limits. We call this problem the Three Opened Assumption’s Inconsistencies, which is the limit through which we consider the relation between heterogeneous fields in set theory. A change in direction is necessary.

In this paper, we present a solution by considering a new mathematical paradigm shift of map transformation from set theory to the Cartesian coordinate system for the interconnection of heterogeneous resources. We represent the Three Opened Assumption’s Inconsistencies in two easy mathematical proofs of transitive and order relations for the interconnection of heterogeneous resources, and define the new functional predicate as an example of the map transformation from set theory to the Cartesian coordinate system for the interconnection of heterogeneous resources. We also represent a correlate function as an example. This
correlate function can measure the relation among the heterogeneous dataset including continuous values.

In this paper, we briefly introduce related work in Section 2 and define the problem in Section 3. We represent the Three Opened Assumption’s Inconsistencies with two easy mathematical proofs in Section 4 and describe a solution for the limitation in Section 5, where we represent the Incomplete Mutual Map Transportation Framework. In Section 6, we define a new predicate function as its example: correlate and shows its preliminary evaluation. We conclude this paper in Section 7.

2. Related Work

The one of the important related works is Linked Data [1]. The Linked Data community [2] advocates a set of best principles for collaborative publishing and interlinking of structured data over the web [3]. Linked Data guidelines facilitate ad hoc re-use and the integration of semi-structured data – across the web – by consumer applications; however, thus far, systems have failed to convincingly demonstrate the potential applications for consuming the currently available Linked Data, which can be seen as a bottom-up approach to Semantic Web [4] adoption. The Linking Open Data [5] project began publishing legacy web corpora under the Linked Data principles, which were connected following datasets to each other: DBpedia [6], corporate entities such as the BBC [7], Thompson Reuters [8], the New York Times [9], and such governmental agencies as the United States [10] and the United Kingdom [11]. Ontology mapping represents the semantic associations among various ontologies (see the survey in [12]). This method was achieved using techniques such as information accessing technology [13], machine learning [14], linguistics [15], structure graph [16], [17], [18], and similarity [19]. Schema mapping, a topic of the interconnection of heterogeneous fields (see the survey in [20]), is attained using techniques such as ontology mapping and creates relationships among different attributes. Therefore, linked data and ontology mapping techniques are sometimes used in schema mapping. Gonzales et al. realized large-scale association rule generation from heterogeneous databases with missing values [21]. We continue to research other methods of heterogeneous data resources [22], [23], [24], [25]. Our proposed method dynamically computes the correlations between concepts in heterogeneous fields by interconnecting each knowledge base arranged on a knowledge grid. For the web, this method provides a framework of infinitely evolving knowledge repositories. The system dynamically creates a semantic associative network that connects user interests and concepts, provides correlation views that represent various relationships, and creates a semantic associative network, depending on the user’s context, from collective knowledge, which is created by a collaborative working environment. Therefore, this method delivers a new browsing style. We employ a link-free browsing system, called correlation browsing, which represents semantic associations that utilize the collective knowledge on a collaborative working environment by connecting various knowledge bases.

3. Problem Definition

Previous methods (as described in Section 2, including our methods) generally represent relationships as links or graph structures. Many researchers address social network analysis ([26]). We focus on human relationships. To define our problem, we show two examples of human relationships in Figures 1 and 2.
First, in Figure 1, we present an example of human relationships between the AI and DB communities that are common fields in these communities. \(a_1\) and \(b_1\) are researchers. The edges indicate relationships that represent the similarity of the research and the symmetric and transitive relationships. When someone adds symmetric and transitive relationships to \(a_3\) and \(b_3\), \(a_1\) is related to \(b_3\) because \(a_1\) is related to \(a_3\), \(a_3\) is related to \(b_3\), and \(b_3\) is related to \(b_5\). Realistically, \(a_1\) may be related to \(b_5\).

Next, in Figure 2, we illustrate another example of personal Relationships between workplace and music communities by assuming that there are no common fields. The edges represent the relationships of friendships and co-workers or co-session members, and the edges indicate the symmetric and transitive relationships. For example, \(a_3\) met \(b_1\) at a party and they became friends. In this case, we can add symmetric and transitive relationships between \(a_3\) and \(b_1\). Is it true that \(a_1\) is related to \(b_5\) when we add symmetric and transitive relationships between \(a_3\) and \(b_1\) in the graph structure? Here, \(a_1\) is related to \(b_5\). However, realistically, \(a_1\) and \(b_5\) do not share common ground without other definitions or analysis. In this case, inconsistencies occur from the previous methods described in Section 2.

The difference between the first and second examples is community positioning. Here, we consider a community to be a set and the persons are its elements. For the first example, the AI community is considered set \(A\) and the DB community is considered set \(B\). The following is the relation of sets \(A\) and \(B\):

\[
A \cap B \neq \emptyset.
\]

For the second example, the workplace community is considered set \(A\) and the music community is considered set \(B\). The following is the relation of sets \(A\) and \(B\):

\[
A \cap B = \emptyset.
\]

We represented the inconsistencies of these examples in a previous method and applied it to various fields. However, we used the results of these methods for cooperation within cases where the near fields are linking or are in the same fields. They do not completely apply to linking heterogeneous fields. The second example may also be applicable to the previous method. In this case, any relation between sets only exists implicitly. When elements are added to heterogeneous sets, their elements have the same order relation. In this case, it is implicitly true that \(A \subseteq B, B \subseteq A\) or \(A \cap B \neq \emptyset\).

When it is true that \(A \cap B \neq \emptyset\), we use the previous methods shown in Section 2. However, we do not make real-world inferences for \(A \cap B = \emptyset\).

Set theory is limited. In set theory, we have to define the transitive and order relationships in each attribute based on the relationships in each new scene. Such old computer systems as database systems and rule-based systems were designed by closed assumption. New scenes do not appear. However, current systems interconnect heterogeneous systems or the data for heterogeneous fields, which are not closed assumptions.

In the big data era, we must discover the relations for \(A \cap B = \emptyset\). We believe that discoveries lead to knowledge. Computers are discovering new relations based on opened assumptions overlooked by humans. Note that we must create a system that discovers relationships when \(A \cap B = \emptyset\).

We focus on the case of \(A \cap B = \emptyset\) because businesses require new big data analytics and it is thus important for computer science researchers to consider this issue.

4. Three Opened Assumption’s Inconsistencies with Two Easy Mathematical Proofs

In the previous section, we described related work and presented our problem. In this section, we discuss the limitations of set theory. That is, we describe the Three Opened Assumption’s Inconsistencies in detail.

First, we present two easy mathematical proofs to represent the Three Opened Assumption’s Inconsistencies in Sections 4.1 and 4.2. We then represent it in Section 4.3 using these two proofs.

4.1 Proof of Inconsistency of Order Relation Between Two Certain Sets

The preconditions of the proofs are as follows: There are two sets, \(A = \{a_1, a_2, \ldots, a_n\}\) and \(B = \{b_1, b_2, \ldots, b_m\}\), where \(A \cap B = \emptyset\). Each set defines the order relations differently.

We prove that we cannot determine the relationship between sets \(A\) and \(B\) or other relationships when we get relationship \(f\) between \(a_1 \in A\) and \(b_1 \in B\).

Proof: We prove that it is satisfied when \(b_1 = f(a_1)\) is not true by induction.

When \(i = 1\), \(b_1 = f(a_1)\) is true by the above condition.
We assume that \(b_k = f(a_k)\) is true when \(i = k\).
When \(i = k + 1\), \(b_{k+1} = f(a_{k+1})\) is not true because set \(A\) has an order relation. However, set \(B\) has another order relation. \(b_k \leq b_{k+1}\) may not be true if \(a_k \leq a_{k+1}\) is true and vice versa. Furthermore, both \(a_k \leq a_{k+1}\) and \(b_k \leq b_{k+1}\) may not be true and, although \(b_1 = f(a_1)\) is true, \(b_1 = f(a_1)\) is not. \([Q.E.D]\)

We cannot uncover the relation between each heterogeneous set when we discover or link between heterogeneous elements. It is also difficult to identify other relations with clue \(b_1 = f(a_1)\).
4.2 Proof of Inconsistency of Order Relation Between Two Certain Sets

Using the same sets A and B as in the former case, set B has order relation \( a_1 \leq b_2 \leq b_3 \leq b_4 \ldots \). Furthermore, set B has a transitive relation; if \( a_1 \leq b_2 \) and \( b_2 \leq b_3 \) is true, then \( b_1 \leq b_3 \) is true. Set A has its own order relation.

Proof: We prove that \( a_1 \leq b_3 \) is true when we obtain relation \( a_1 \leq b_1 \). To reveal the conclusion, \( a_1 \leq b_3 \) may not be satisfied. We thus show a counter example: Assume \( a_1 = (1, 5) \), \( b_1 = (2, 1) \), \( b_2 = (3, 2) \), and \( b_3 = (4, 3) \). Moreover, \( a_i \) focuses on each first element. Then \( a_1 \leq b_1 \) is true. The order relation of set B focuses on more values of each second element. Then \( b_1 \leq b_2 \leq b_3 \), and if \( b_1 \leq b_2 \) and \( b_2 \leq b_3 \) are true, then \( b_1 \leq b_3 \) is true. However, \( a_1 \leq b_1 \) is not true in the order set of set B. Like the relation of \( a_1 \) and \( b_1 \), an inconsistency occurs whose order and transitive relations of set B are not guaranteed.

Although we strictly define order and transitive relation in a certain set, an inconsistency occurs with a relation with the element outside of the set.

4.3 The Three Opened Assumption’s Inconsistencies

Until now, computer science researchers have based their ideas on closed assumptions. We freely link and interconnect each object. For example, interconnection between element \( a_i \) in set A and element \( b_j \) in set B for \( A \cap B \neq \emptyset \) remains a closed assumption. However, users, especially data intensive scientists, do not require such knowledge. They have to consider new discovery methods in an opened assumption, where \( A \cap B = \emptyset \).

Note that the inconsistencies only occur when extending the current methods introduced in Section 2. We call these inconsistencies the Three Opened Assumption’s Inconsistencies, which were proved in Sections 4.1 and 4.2.

(1) One inconsistency is that a relation does not guarantee the future.

For example, we can identify relationships among each set through data mining technology. Note that the results only represent the relationships of the present data. These relationships are not guaranteed if the system adds new records (data). Occasionally, researchers and users anticipate an uncertain value of a new record using extracted relationships. However, such usage is incorrect. Due to the insignificance of predicting uncertain values by data mining, we assume that sets A and B are attributes in the relational database and that \( a_i \) and \( b_j \) are the attribute values of each set. The data mining result is guaranteed if no updates occur. However, most tables undergo many updates. We assume \( k \) records in the database and that the numbers of each attribute value are \( k \). The system performs data mining and extracts \( b_j = f(a_i) \). This relation \( f \) is only guaranteed when there are \( k \) records in the database. If the number of records is \( k + 1 \), relation \( f \) is not guaranteed. Indexing relations, which are extracted by some methods, is meaningless for predicting uncertain or missing values.

(2) Another inconsistency is where no transitive relation is true when anyone connects links for heterogeneous fields.

With closed assumption, we create or extract relationships in a set. In this case, the transitive and order relations are true. However, they are not true when we create or extract relationships over the sets shown in Section 4.2. This phenomenon occurs when we use bridge ontology, semantic webs, linked data, etc. Each ontology in specific fields is unconsciously created in closed assumptions. These techniques connect ontologies in specific fields that are changed from closed to open assumptions. Therefore, it may become possible to use these ontologies only by connecting each element of each set. On the contrary, is the determinant possible when trying to create bridge ontology? Such determination is difficult.

(3) The third inconsistency is where no relation in heterogeneous fields can be discovered in set theory.

Should such relationships be indexed or aggregated? We might semantically discover new relationships, but how to discover new relationships is not understood. Even if part of the relations of each element of the sets is known, the relations of all of the sets are not guaranteed. Moreover, the relation is not guaranteed when a new record is entered, even if the relation of the sets was previously guaranteed. Therefore, even if we can retrieve the relationships, their discovery is impossible by inference and reasoning because only the relationships that we have discovered are effective; transitive and order relations are not true when we create or extract relationships over the sets shown in Section 4.2, as was previously described in Section 4.3(2). This is a disappointing result. However, it represents a paradigm shift from closed to opened assumption systems.

By connecting each element of each set, it may become impossible to use these ontologies. Computing some systems is very dangerous with bridge ontology and linked data. On the contrary, is the determinant in the case of trying possible for the author of bridge ontology? Determining this is very difficult. Of course, schema mapping has the same problem, since RDB has relation \( \sqsubseteq \). Discovering new relations by inference and reasoning is difficult.
As shown in Sections 4.1 and 4.2, a paradigm shift is needed because current theory cannot detect any new relationships.

One solution is our framework – the Incomplete Mutual Map Transformation Framework between set theory and the Cartesian coordinate system. This method approximates and maps from the world of modern function theory to the world of classic function theory and vice versa. By mapping the Cartesian coordinate system, the order relation is automatically decided. In addition, in the big data era, there is much continuous multi-value data. Such computation is easier in the Cartesian coordinate system than in set theory.

Until now, description logic (e.g., see [27]) systems calculate the inconsistencies in models with strong theoretical support, e.g., analyses of computational complexity. Providing relations to such works could increase reader understanding of our proposed idea. In addition, at a recent Ontology Alignment Evaluation Initiative (OAEI), some groups used a type of background knowledge (e.g., a strong ontology that can be mapped to both target ontologies) to consider such inconsistencies as hints to generate better ontology mappings. For details, see LogMap [28] and other works.

Our model maps between set theory and the Cartesian coordinate system by defining the functional predicate. We can detect various differences and relative relationships by comparisons. It is important to use the old function family on databases designed on set theory to achieve mutual mapping. By actualizing this framework, we can discover new relationships in the Cartesian coordinate system and retrieve current relationships in set theory.

5. Incomplete Mutual Map Transformation Framework for Relation Discovery

In this section, we describe the Incomplete Mutual Map Transformation Framework between set theory and the Cartesian coordinate system. The point is "incomplete." The map transformation is approximate values because the concept of set theory is wider than the Cartesian coordinate system.

In this method, users define the functional predicates. Each functional predicate has features in both set theory and the Cartesian coordinate system. Users also define the features of the function predicates in both set theory and the Cartesian coordinate system. With this definition, we can mutually map between set theory and the Cartesian coordinate system.

For discovery, we calculate in the Cartesian coordinate system and, for retrieval and any other set operation, we calculate in set theory. If we calculate in real-time, the framework can clean up the Three Opened Assumption’s...
intensity are available. Moreover, not only numerical values, but also words, are available. That is, terms are also available that are ordered by the alphabet, names, and symbols that determine the order relation.

2) Uniqueness Evaluation (UE): The elements of other sets are mapped on the axes selected in the former process. Of course, the elements of the set cannot be mapped in the selected axes. Furthermore, each value on the axis should only be one value by one element of the set. However, such cases are few. If two or more elements of the sets are mapped on the same value on an axis, we do the following:

① When two or more elements in the set are mapped by the same value on an axis, let the average of the elements be the value of the set to the value on the axis.

② Uniqueness is evaluated by calculating the variances.

In Item 2, when the variance is low, its uniqueness is high because most of the elements are centered on the average value. Conversely, when the variance is high, its uniqueness is low because each element greatly varies from the average value. If some of the elements in the set can be mapped into the axis, we should map on the axis. If there are some missing or uncertain values, we assign a value to the maximum and minimum values, and calculate the average and the variance.

3) Certainty Evaluation (CE): Some uncertain or missing values may exist when we map from set theory to the Cartesian coordinate system. In this case, we calculate the certainty evaluation by driving the rate of an uncertain value from among the numbers of the set’s elements to the map.

4) Behavior of values for representing predicates (BV): This process is crucial. We must identify the behavior of the values that are mapped in the axis for representing predicates because it is important to coordinate between the behaviors of the value and the predicate on an axis. We define such functions as disjoint, meet, overlap, coveredBy, covers, equal, contain, inside, correlate, moreThan, lessThan, alongWith, join, and domainDependedCorrelate on the axis if we coordinate between the behavior of the value and the predicate on an axis. Since these functions operate on the Cartesian coordinate system, each function can be clarified based on its norm, distance, and inner product, if it is not special. We can create functions more viscera and freely than the set theory’s function. Here, we calculated the mapped elements using the function defined above.

5.2 Features of Functional Predicates in the Set Theory

Even though we discovered some relations with the former process, we still have to resolve the results in set theory in order to use the old functions that are created in such set theories as SQL operations. This creation is not difficult. We only have to define whether our defined function satisfies such binary relations as reflexive, antisymmetric, transitive relation, and commutative. For a relation between two sets, we should also verify such binary operations as commutative, associative, and distributive properties. We operate on set theory using the discovered relations in these processes.

1) Reflexive/Irreflexive/Coreflexive Relations: We predict that the reflexive relation is almost true on the Cartesian coordinate system, where we can also verify it.

2) Symmetric/Antisymmetric/Asymmetric Relations: The symmetric relation may be equivalent to the problem of whether any inverse function exists on the Cartesian coordinate system, depending on the case. We must calculate the inverse function on the Cartesian coordinate system.

3) Transitive Relation: The transitive relation is defined by the behavior of the values for representing the predicates on the Cartesian coordinate system because we can verify it.

4) Commutative Property: The commutative property is also defined by the behavior of the values for representing the predicates on the Cartesian coordinate system because we can verify it.

5) Associative Property: The associative property is also defined by the behavior of the values for representing the predicates on the Cartesian coordinate system because we can verify it.

6) Distributive Property: The distributive property is also defined by the behavior of the values for representing the predicates on the Cartesian coordinate system because we can verify it.

6. An Example of Functional Predicate Definition and Preliminary Evaluation

In this section, we define a functional predicate called "correlate" in our framework by such functions as dependOn, disjoint, meet, overlap, coveredBy, covers, equal, contain, inside, moreThan, lessThan, alongWith, join, etc. on the same axis as this process.

“Correlate” means that data sets X and Y are correlated depending on Z. Note that, for simplicity, these are represented in correlate(X,Y|Z) when data sets X and Y are correlated depending on Z.

Table 1 shows the feature definition of "correlate,” which is one example of a functional predicate. When each absolute of evaluation value on Cartesian Coordinate System (AAE, UE, CE and BV) is higher, a condition is higher correlation.

In this section, we represent a preliminary evaluation of a predicate function called "correlate” with two actual
datasets shown in Tables 2 and 3. Table 2 shows the monthly average temperatures downloaded from the Japan Meteorological Agency website. Table 3 shows the annual averages of the price (yen) of cucumbers (5 kg) and cabbages (10 kg) from the Ministry of Agriculture, Forestry and Fisheries of Japan website.

For example, we can learn how the temperature in a particular month affected the prices of cucumbers and cabbage with the predicate function "correlate" in the Incomplete Mutual Map Transformation Framework between set theory and the Cartesian coordinate system.

Table 2. Monthly Average Temperatures in Gumma Prefecture, Japan

<table>
<thead>
<tr>
<th>Year</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>4.6</td>
<td>6.1</td>
<td>8.2</td>
<td>12.2</td>
<td>18.7</td>
<td>22.7</td>
<td>21.1</td>
<td>26.2</td>
<td>26.8</td>
<td>22.9</td>
<td>17.6</td>
<td>10.7</td>
<td>6.5</td>
</tr>
<tr>
<td>2008</td>
<td>5.6</td>
<td>2.9</td>
<td>9.1</td>
<td>13.6</td>
<td>18.0</td>
<td>22.2</td>
<td>21.2</td>
<td>25.2</td>
<td>26.8</td>
<td>22.8</td>
<td>17.2</td>
<td>10.6</td>
<td>6.8</td>
</tr>
<tr>
<td>2009</td>
<td>4.3</td>
<td>5.2</td>
<td>7.6</td>
<td>14.0</td>
<td>19.4</td>
<td>22.2</td>
<td>25.4</td>
<td>25.8</td>
<td>22.1</td>
<td>16.8</td>
<td>11.4</td>
<td>6.3</td>
<td>15.1</td>
</tr>
<tr>
<td>2010</td>
<td>4.3</td>
<td>4.8</td>
<td>7.2</td>
<td>11.2</td>
<td>18.1</td>
<td>23.5</td>
<td>27.2</td>
<td>29.1</td>
<td>24.2</td>
<td>17.1</td>
<td>11.2</td>
<td>7.2</td>
<td>15.5</td>
</tr>
<tr>
<td>2011</td>
<td>2.4</td>
<td>4.9</td>
<td>6.1</td>
<td>12.6</td>
<td>17.6</td>
<td>22.6</td>
<td>27.1</td>
<td>28.6</td>
<td>23.8</td>
<td>17.1</td>
<td>12.3</td>
<td>4.8</td>
<td>14.9</td>
</tr>
</tbody>
</table>

Each attribute seems to be sets. If the years can be an axis, then each temperature and each price are mapped and "correlate" is verified by the evaluation indexes shown in Table 1.

We have data sets such as year, month, temperature, vegetable price, cucumber price, and cabbage price. All data sets are represented in continuous values. Therefore, we can consider each data set as each axis.

We next have to create the mapping function. For this example, the year is the same attribute. Therefore, an almost complete mapping function can be created. In this paper, we focus on the year axis. In actuality we would have to process all axes; however, for the purposes of this paper, these other axes are omitted. (Try to calculate the case of the other axes.)

Table 3 shows the calculation result of the function predicate, correlate (temperature, cucumber price | year) and correlate (temperature, cabbage price | year). It is apparent that October and cucumber have a higher value. In Japan, the harvest time for cucumbers is October. It is also apparent that December and cabbage have a higher value. Cabbages harvested from December to April are called “winter cabbages” in Japan.
This process means that month, temperature, and cucumber and cabbage price are mapped and compared on the year axis. Figure 4 compares cucumber price with October’s temperature (high correlation on BV) and May’s temperature (low correlation on BV) on the year axis. The transition of cucumber price on the graph is more similar to December’s temperature than April’s. Therefore, we have discovered a relation between heterogeneous data sets. In this example, the prior method still drives this result because the year is common. However, this method can discover this relation without a common data set.

The focal point of this architecture represents not only the correlation defined as the functional predicate, but also the reason. In the example, we discovered a correlation between cucumber price and temperature in October and cabbage price and temperature in December on the year’s axis. Previous methods may uncover some correlation, but cannot present the reason. Our framework can.

This result not only substantiates the predicate function “correlate,” but also the discovery of linking heterogeneous databases by a predicate function. Our method discovers them without knowledge, descriptions, or relations with their predicates from numerical value data. The system determines the relationships with their various predicates by defining the predicate functions from various data when we define such functions as disjoint, meet, overlap, coveredBy, covers, equal, contain, inside, moreThan, lessThan, alongWith, and join. We can therefore operate in set theory because we can also define the feature of the predicate function in set theory. However, we should re-verify the “correlate” predicate function when the data are updated.

7. Conclusions

We highlighted the inconsistencies of past researches that contributed to the interconnection of such heterogeneous fields as Linked Data, Semantic Web, and our past studies. We illustrated the Three Opened Assumption’s Inconsistencies using two easy mathematical proofs of transitive and order relations for the interconnection of heterogeneous resources. We also presented the “correlate” function as an example of map transformation from set theory to the Cartesian coordinate system for the interconnection of heterogeneous things.

In our method, we can define such functions as disjoint, meet, overlap, coveredBy, covers, equal, contain, inside, moreThan, lessThan, alongWith, and join if we coordinate the behavior of the value and the predicate on an axis.

Table 4. Each Value of “Correlate” in the Cartesian Coordinate System

<table>
<thead>
<tr>
<th>Month</th>
<th>Jan’07</th>
<th>Feb’07</th>
<th>Mar’07</th>
<th>Apr’07</th>
<th>May’07</th>
<th>Jun’07</th>
<th>Jul’07</th>
<th>Aug’07</th>
<th>Sep’07</th>
<th>Oct’07</th>
<th>Nov’07</th>
<th>Dec’07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cucumber</td>
<td>0.030772549</td>
<td>0.393712691</td>
<td>0.028150432</td>
<td>0.345250399</td>
<td>0.707470032</td>
<td>0.001508122</td>
<td>0.207245927</td>
<td>0.188127809</td>
<td>0.354845675</td>
<td>0.924099011</td>
<td>0.090017819</td>
<td>0.042639156</td>
</tr>
<tr>
<td>May’08</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Cabbage</td>
<td>0.24327395</td>
<td>0.02434448</td>
<td>0.007064724</td>
<td>0.199282939</td>
<td>0.052442423</td>
<td>0.255172898</td>
<td>0.044815489</td>
<td>0.330351927</td>
<td>0.002718054</td>
<td>0.113498639</td>
<td>0.046628214</td>
<td>0.435773561</td>
</tr>
</tbody>
</table>

Figure 4. Comparing cucumber price with October’s temperature (high correlation on BV) and May’s temperature (low correlation on BV) on the year axis.

Figure 5. Comparing cabbage price with December’s temperature (high correlation on BV) and April’s temperature (low correlation on BV) on the year axis.

We have discovered a relation between heterogeneous data sets. In this example, the prior method still drives this result because the year is common. However, this method can discover this relation without a common data set.
We also introduced a preliminary evaluation of the predicate function "correlate." This result not only verifies the "correlate" predicate function, but also links heterogeneous databases by predicate function. Our method discovers relationships without knowledge or description from numerical value data. The system detects the relationships by defining the predicate function. With these relationships, we can operate in set theory because we also define the features of the predicate function in it. However, we must re-verify the predicate function "correlate" when the data are updated.

In the big data era, it is imperative that we design systems in opened assumptions. We cannot satisfy user needs with the previously designed closed assumption systems. Their extensions are also the same. Since we cannot describe the relationships of all the data, we must construct a system that discovers the relations from various data. However, there are the Three Opened Assumption’s Inconsistencies. Our framework discovers links to heterogeneous databases by predicate functions. In this preliminary evaluation, our framework is efficient for the big data era. In this paper, we provided proofs showing that heterogeneous field integration is achieved by bridging existing relationships, linking existing resources, etc. Users require new systems. We must design an on-demand relation discovery system like our framework.

References


Takafumi Nakanishi is a researcher at the Universal Communication Research Institute of the National Institute of Information and Communications Technology, Japan. Dr. Nakanishi’s research focus is cross-database systems, social media content analysis, multimedia content retrieval, and intellectual property analysis. He is currently developing an evaluation method of intellectual property and intellectual property analysis and a management system. He is an active member of the IPSJ, IEICE, DBSJ, and IPAJ (Intellectual Property Association of Japan). He is a 2nd Grade Certified Specialist of Intellectual Property Management (Administration).

Kiyotaka Uchimoto is a research manager at the Universal Communication Research Institute of the National Institute of Information and Communications Technology, Japan. Dr. Uchimoto’s research focus is corpus-based natural language processing, knowledge acquisition, and machine translation. He is an active member of the ACL, IPSJ and ANLP.

Yutaka Kidawara is a director general at the Universal Communication Research Institute of the National Institute of Information and Communications Technology, Japan. Dr. Kidawara’s research focus is digital content management, ubiquitous computing, information retrieval, and information analysis. He is an active member of the IEEE, IPSJ, IEICE, and DBSJ.
Online Handwritten Chemical Expression Recognition

Peng Tang a
Nanyang Technological University, Singapore
Siu Cheung Hui a
Nanyang Technological University, Singapore
Chi-Wing Fu a
Nanyang Technological University, Singapore
cwfu@ntu.edu.sg

Abstract

With the growing popularity of pen-based and touch-based devices such as Apple’s iPad and Samsung’s Galaxy Tablet, handwriting has become an important input method by the general public. Although handwriting recognition for textual contents and mathematical formulae are well-supported in these devices, recognizing handwritten structural chemical expressions is still challenging due to its complex spatial structure. All the existing handwritten chemical structural expression recognition approaches are simply not user-friendly as it usually requires the user to complete the entire chemical expression before the machine recognition. To tackle this problem, we propose a new approach for dynamic and progressive recognition of handwritten chemical expressions. The proposed progressive approach recognizes and displays the recognition result after each new symbol is written. By our method, the user can obtain early feedback on any recognition errors, and efficiently make the corresponding correction. To demonstrate the efficacy of our method, we have implemented it on the Apple’s iOS platform and evaluated it with several users to verify its promising performance as a real application.

Keywords: Handwritten Chemical Expression, Symbol Recognition, Structural Analysis, Handwriting Recognition, Online Sketch Recognition, Progressive Sketch Recognition

1. Introduction

Chemical expressions (chemical formulae or chemical equations) are used for presenting and formulating chemical representations and reactions. In many drug discovery projects, it is often required to search for similar chemical structural formulae of drug-like compounds that are worthy for further synthetic or biological investigation [1]. To search for relevant information of a chemical compound or chemical reaction, we need to enter the chemical expression as a query. Chemical expression could be written in simple text formats such as HNO3 and CH4, or drawn in complex 2-dimensional structures. The current input methods for chemical expressions such as ChemDraw [2] and Accelrys Draw [3] are mainly template-based. These systems rely on the traditional point-click-and-drag style of user interaction.

Compared to handwriting-based approach, the template-based approach is much more complicated, time-consuming, and non-intuitive.

With the growing popularity of pen-based and touch-based devices such as Apple’s iPad and Samsung’s Galaxy Tablet, handwriting-based input has become an important input method. Currently, the handwriting support for text contents and mathematical formulae [4,5] are widely available in most of these devices. However, due to the spatial complexity involved in recognizing structural chemical expressions, it is challenging to recognize handwritten chemical expressions with high precision and efficiency. There are two main processes for handwritten chemical expression recognition: handwritten chemical symbol recognition and chemical structural analysis. Handwritten chemical symbol recognition recognizes individual chemical symbols, whereas chemical structural analysis interprets the sequence of recognized symbols together with the spatial structural information for recognizing the whole chemical expression.

Currently, a few approaches [6,7,8] have been investigated for the recognition of handwritten chemical structural expressions. However, these approaches only perform the recognition process after the user has completed drawing his chemical expression. This will cause delay and frustration to the user if recognition error occurs as he may need to re-draw the whole expression again. To tackle this problem, we propose a progressive approach for dynamic recognition of handwritten chemical expressions. The proposed approach will recognize a handwritten chemical expression progressively while the user is drawing the expression. As such, it helps the user to identify any recognition errors early, and make immediate correction if necessary. Compared with other handwritten chemical expression recognition approaches, the proposed approach is much more efficient and user-friendly. In addition, we also aim to develop our proposed approach for the touch-based device, Apple’s iPad, under the iOS operating platform. To the best of our knowledge, currently there is no handwritten chemical expression recognition system running on iPad.

In this paper, we will discuss the proposed approach for handwritten chemical expression recognition and its performance evaluation. The rest of the paper is organized as follows. Section 2 reviews the related work on handwritten chemical expression recognition. Section 3 describes the proposed progressive approach for online handwritten chemical expression recognition. Experimental results are then presented in Section 4. Finally, Section 5 concludes the paper and discusses the directions for future extension.
2. Related Work

Handwritten chemical expression recognition consists of two main processes, namely handwritten chemical symbol recognition and chemical structural analysis.

2.1 Handwritten Chemical Symbol Recognition

Symbol recognition is a popular research area, which has been studied quite extensively for the past few decades. Currently, there are two approaches for handwritten chemical symbol recognition, namely offline recognition and online recognition. The offline recognition approach aims at recognizing chemical symbols based on the image input of the pen strokes. It is usually performed after the handwritten input image is captured. Ramel et al. [6] proposed an approach for offline recognition of handwritten chemical formulas. It accepts the image of a handwritten chemical formula. Next, it extracts the quadrilateral primitives (line vectors) from the image and form a structural graph of the chemical based on structural analysis. The symbol recognition is then performed based on the constructed structural graph.

On the contrary, the online recognition approach recognizes chemical symbols when it is written with a pen-based device, where a sensor picks up the pen-tip movements as well as pen-up/pen-down switching. This kind of data is known as digital ink and can be treated as a dynamic representation of the handwriting. Ouyang and Davis [7] proposed to use a discriminative classifier based on Support Vector Machines (SVM) [9] to recognize chemical symbols. The symbol recognizer evaluates all combinations of up to 7 sequential strokes. Each combination of strokes will be represented by a set of geometric and statistical features. Based on these features, each group of strokes are identified either as a valid symbol or invalid groups. In [8], Ouyang and Davis proposed another approach for real-time sketch recognition of chemical drawing. In this approach, the symbol recognition uses three levels of features. The first level is based on ink points. The second level is based on stroke segments by dividing strokes at corner points. The third level is based on candidate symbols identified by considering all possible sequences of up to 8 consecutive strokes. Conditional Random Field (CRF) is used to combine the features from the three levels and perform the symbol recognition.

In [10], Zhang et al. proposed an online handwritten recognition method for chemical symbols. The method is based on Hidden Markov Models (HMMs), which are increasingly being used to model characters. It transforms each sample symbol as a sequence of feature vectors or a feature matrix. Then, an HMM model is built based on the feature matrix of the training sample chemical symbols. Moreover, Zhang et al. [11] also proposed a double-stage classifier for handwritten chemical symbol recognition. The first stage is a rough classification, the SVM method is used to distinguish non-ring structure and organic ring structure symbols. The HMM method [10] is then used for refining the recognition at the second stage.

2.2 Chemical Structural Analysis

Chemical structural analysis identifies the relationships among all the recognized symbols and then constructs the chemical expression based on the identified relationships. Different techniques have been investigated for structural analysis for handwritten chemical expression recognition. Casey et al. [12] and Ramel et al. [6] proposed an image-based approach for chemical structural analysis. Quadrilateral primitives are extracted from the image of a structural formula and relationships among primitives are then classified for structural analysis.

Apart from the image-based approach, stroke-based approaches are also proposed for online structural analysis. Ouyang and Davis [7] proposed to use the local spatial context of chemical symbols and domain-specific rules for chemical structural analysis. Chang et al. [13] proposed a graph-based representation for generic chemical expressions, and formulated the structural analysis problem as a weighted directed graph search problem. Wang et al. [14] proposed a grammar spanning tree approach for chemical structural analysis. In this approach, chemical symbols are stored in a grammar-spanning tree based on a set of context-free rewriting rules for structural analysis. Ouyang and Davis [8] proposed the agglomerative clustering algorithm for structural grouping of chemical symbols. After analyzing symbol connectivity based on pairwise distance metrics such as bond-element distance, element-element distance and bond-bond distance, the agglomerative clustering algorithm iteratively merges two nearest symbols or symbol clusters for structural analysis.

3. Progressive Handwritten Chemical Expression Recognition

3.1 Proposed Approach

In this paper, we propose a progressive approach for handwritten chemical expression recognition which consists of the handwritten chemical symbol recognition and chemical structural analysis processes as shown in Figure 1. In the proposed approach, a hybrid Support Vector Machine-Elastic Matching (SVM-EM) approach is proposed for handwritten chemical symbol recognition, while a progressive structural analysis approach is proposed for chemical structural analysis.
3.2 Handwritten Chemical Symbol Recognition

In the proposed hybrid SVM-EM approach, Support Vector Machine (SVM) [9] is used for chemical symbol recognition and Elastic Matching (EM) [15] is used for the verification of the recognition results. The hybrid SVM-EM approach for handwritten chemical symbol recognition consists of the following five processes: Stroke Partitioning, Stroke Preprocessing, SVM Recognition, Elastic Matching Verification and User Feedback.

Before discussing the five chemical symbol recognition processes, we first define pen stroke as follows.

**Definition 1 (Stroke).** A stroke is a sequence of \( n \) two-dimensional points \( (p_1, p_2, ..., p_m) \):

\[
s = (p_1, p_2, ..., p_m)
\]

where \( p_i = (x_i, y_i), 1 \leq i \leq m \), \( p_1 \) is the pen-down point, \( p_m \) is the pen-up point and \( p_2, ..., p_{m-1} \) are the pen-move points. \( m \) is the number of points in the stroke. The stroke information are captured by the pen-based device when user writes on the device.

1) **Stroke Partitioning.** Stroke partitioning aims to group strokes correctly into symbols since a chemical symbol may comprise more than one stroke. The main challenge of stroke partitioning is that it does not have any obvious rules to group the strokes that belong to a symbol. In the proposed approach, when there is a new stroke written, this new stroke will be grouped with some previously written strokes to form different stroke groups. We assume that the users always complete all the strokes of one symbol before writing the next symbol, and they do not make any corrections to the previously written symbols. Therefore, we only group consecutively written strokes. Since each chemical symbol contains at most four strokes, the most recent strokes will be grouped together up to the last 4 previous strokes. The invalid stroke groups will be filtered out by spatial distance checking and stroke intersection checking. Spatial distance checking filters out invalid stroke groups by checking the spatial distance between two consecutive strokes. Stroke intersection checking filters out invalid stroke groups which include only one of the two intersected strokes.

2) **Stroke Preprocessing.** After the user has written a stroke on the drawing panel, the sequence of stroke points is captured. However, the stroke point list is usually very large and unstructured. Stroke preprocessing aims to structure and standardize the stroke points. As such, it will help reduce the possibility of recognition errors due to noisy data in the partitioned strokes. Stroke preprocessing is further divided into three steps: Size Normalization, Smoothing and Sampling. Before discussing the preprocessing steps, we define bounding box as follows.
**Definition 2** (Bounding Box). The bounding box $B$ of a symbol $S$ is the smallest rectangle which encloses $S$. The edges of the rectangle are in parallel with the coordinate axes.

Size normalization normalizes a symbol into a standard size because the handwritten symbols from users are in various sizes. Size normalization is performed based on the bounding box of each symbol. Smoothing removes the outliers in the sequence of stroke points which can be caused due to unsteady handwriting or uneven surface of the device. In this research, we have used average smoothing in which the mid-point of every three consecutive points is replaced by the average of the three points. Sampling aims to sample and redistribute a certain number of stroke points evenly for the valid partitioned strokes. In this research, the number of sampling points is set to 50 points. Figure 3 shows an example for the stroke preprocessing process.

3) **SVM Recognition.** SVM recognition is the kernel process that recognizes the partitioned strokes into a chemical symbol. After recognition, a list of candidate symbols, sorted by their SVM confidence levels, will be generated. The SVM confidence level ranging from 0 to 1 indicates the accuracy of symbol matching. There are two steps in SVM recognition. *Feature Extraction* extracts the various kinds of symbol features from the partitioned strokes while *SVM Classification* performs symbol classification based on the extracted features.

In feature extraction, the extracted symbol features include the number of strokes, stroke point coordinates, horizontal angle and turning angle. The number of strokes is the most basic and necessary information used for recognizing chemical symbols. The stroke point coordinates include the $X$ and $Y$ coordinates of the sampled stroke points for the partitioned strokes. Same as the number of strokes, the coordinates of stroke points are also important features.

**Definition 3** (Horizontal Angle). Considering two consecutively written stroke points $P_1$ and $P_2$, the horizontal angle $\phi$ is defined as the angle between the stroke line connecting $P_1$ and $P_2$ and the positive direction of horizontal axis as illustrated in Figure 4. Each horizontal angle represents the slope of a stroke line. Horizontal angle ranges from $-180'$ to $180'$.

In this research, the horizontal angles are divided into 12 groups with $30'$ for each group. And the horizontal angle feature is measured by the percentage of horizontal angles falling into a group.

**Definition 4** (Turning Angle). Considering two horizontal angles $\phi_1$ and $\phi_2$ which are formed by a sequence of three consecutively written stroke points $P_1$, $P_2$ and $P_3$, the turning angle $\theta$ is defined as the difference between the two horizontal angles $\phi_1$ and $\phi_2$ as illustrated in Figure 5. It measures the curvature of the sequence of stroke points. Turning angle ranges from $0'$ to $180'$.

Similar to horizontal angles, turning angles are divided into 18 groups with $10'$ for each group. The turning angle feature is also measured by the percentage of turning angles falling into each group.

After the symbol features are extracted from the partitioned strokes, the SVM classification step generates a list of candidate symbols that match with the partitioned strokes. In this research, we use the LibSVM [16] to train a linear SVM Classifier and perform SVM classification. To train the SVM classifier, symbol features are extracted from a set of sample handwritten symbols collected from different users. The training set consists of the most commonly used symbols in 2-dimensional chemical expressions including 10 Arabic numerals, 52 alphabetical letters, and 18 bond and operator symbols.

4) **Elastic Matching Verification.** After SVM recognition, elastic matching checks the similarity between the candidate symbols and the partitioned strokes by calculating the elastic distance which is defined as the point-to-point distance between the sequence of stroke points of the candidate symbol and the partitioned strokes. The stroke point sequence of each candidate symbol is retrieved from the Chemical Elastic Symbol Library which is created to store the stroke points of the handwriting samples of all the chemical symbols.

Different users may write a symbol in different sizes, stroke orders or stroke directions. For stroke size, it will be standardized in the preprocessing process. For stroke order and stroke direction, all the possible stroke sequences and stroke directions of the symbols are stored in the symbol library. Due to the limited number of possible stroke
combinations of each symbol, it is feasible to collect all the samples. Therefore, to calculate the elastic distance for a candidate symbol, the distance between the partitioned strokes and each sample of the candidate symbol is calculated. And the minimal elastic distance of all the symbol samples will be the elastic distance for that symbol. Algorithm 1 gives the algorithm of the elastic matching verification process.

Algorithm 1 Elastic Matching Verification Algorithm

Input:
- \( S_p \) – The partitioned and preprocessed strokes
- \( C_{svm} = \{c_0, c_1, ..., c_i, ..., c_m\} \) – The list of candidate symbols from SVM recognition
- \( L \) – The elastic chemical symbol library

Output:
- \( C_{em} \) – The list of candidate symbols after elastic matching

Process:
1. \( D \leftarrow \emptyset \)
2. for \( c_i \in C_{svm} \) do
   a. \( S \leftarrow \text{getSamples}(c_i, L) \)
   b. distance \( \leftarrow \emptyset \)
   c. for \( s_i \in S \) do
      i. \( d \leftarrow \frac{\sum_{j=1}^{n} D(p(s,j), p(s_p,j))}{n} \)
      ii. distance \( \leftarrow \text{min}(\text{distance}, d) \)
   end for
   d. \( d_i \leftarrow \text{distance} \)
   e. \( D \leftarrow D \cup d_i \)
3. end for
4. \( C_{em} \leftarrow \text{sort}(C_{svm}, D, \text{ascending}) \)
5. return \( C_{em} \)

The distance function for elastic distance is defined as follows:

\[
\text{Distance}(s, s_p) = \sum_{j=1}^{n} D(p(s,j), p(s_p,j))
\]

where \( s \) is the candidate symbol and \( s_p \) is the partitioned strokes of the input symbol. \( p(s,j) \) is the \( j \)-th stroke point in \( s \) and \( p(s_p,j) \) is the \( j \)-th stroke point in \( s_p \). \( D(p(s,j), p(s_p,j)) \) is the Euclidean distance of the two stroke points \( p(s,j) \) and \( p(s_p,j) \). \( n \) is the number of the stroke points in each symbol after stroke preprocessing. The list of candidate symbols will be sorted in ascending order according to the elastic distance. The candidate symbol with the minimal elastic distance will be ranked at the top of the candidate symbol list.

Figure 6 shows an example of the elastic matching verification process. The candidate symbol which minimizes the elastic distance will be listed at the top of the ranked candidate symbol list.

5) User Feedback. After elastic matching verification, the best matching symbol will be treated as the recognition result. However, the list of the ranked candidate symbols is returned to the user instead of just returning the best matching symbol. If the best matching symbol is not the symbol that the user has written, the user can select the correct symbol from the candidate symbol list. If the symbol that the user has written does not occur in the candidate symbol list, the user can select the undo operation and rewrite the strokes or symbol again.

6) Overall Algorithm. Algorithm 2 gives the algorithm for our proposed chemical symbol recognition approach.

Algorithm 2 Chemical Symbol Recognition

Input:
- \( s_{last} \) – The last drawn stroke
- \( S = \{s_0, s_1, ..., s_n\} \) – The list of previously drawn strokes

Output:
- \( c_r \) – The recognized symbol
- \( C = \{c_0, c_1, ..., c_m\} \) – The ranked list of candidate symbols in descending order of recognition confidence

Process:
1. \( S_{par} \leftarrow \text{strokePartitioning}(s_{last}, S) \)
2. \( S_{pre} \leftarrow \text{preprocessing}(S_{par}) \)
3. \( C_{svm} \leftarrow \text{SVMRecognition}(S_{pre}) \)
4. \( C_{em} \leftarrow \text{EMVerification}(C_{svm}) \)
5. \( c_r \leftarrow c_0 \)
6. return \( c_r, C \)

3.3 Chemical Structural Analysis

After chemical symbol recognition, the list of recognized symbols can be analyzed to construct a valid chemical expression. Unlike textual inputs, chemical symbols are arranged in a spatial two-dimensional structure, possibly with different symbol sizes. Therefore, the chemical structural analysis process is very challenging even when all the chemical symbols can be recognized correctly. The proposed progressive chemical structural analysis approach [17] consists of the following four steps: Related Symbol Identification, Symbol Relationship Identification, Chemical Expression Tree Update and Displaying Recognition Result.

1) Chemical Expression Tree. After analyzing the spatial properties of structural chemical expressions, eight
possible spatial relationships between two neighboring symbols can be identified in a chemical expression [17]. These relationships include left, right, superscript, subscript, above, under, pre-superscript and pre-subscript. The eight relationships can be further classified into row relationship and parent-child relationship. They are defined as follows.

**Definition 5 (Row Relationship).** Two chemical symbols have row relationship if they are aligned to each other horizontally or they have the left/right relationship.

**Definition 6 (Parent-child Relationship).** Two chemical symbols have parent-child relationship if one symbol has another symbol as its argument with the superscript, subscript, above, under, pre-superscript or pre-subscript relationship. The symbol is the parent symbol of its argument symbol and the argument symbol is the child symbol. The parent symbol occupies a dominant position over its child symbol.

For example, in the expression ‘CH₄ + 2O₂’, the symbol pairs (C, H), (H, +), (+, 2) and (2, O) have row relationships, whereas the symbol pairs (H, 4) and (O, 2) have parent-child relationships. Taking the symbol pair (H, 4) as an example, ‘H’ is the parent symbol of ‘4’ because ‘H’ has a subscript relationship with ‘4’ and ‘H’ occupies the dominant position over ‘4’.

With the chemical symbols and its relationships, we can construct a Chemical Expression Tree (CET) which consists of three types of tree nodes: root node, symbol node and relation node. They are defined as follows.

**Definition 7 (Root Node).** A root node is the root of a Chemical Expression Tree. It represents a chemical expression.

**Definition 8 (Symbol Node).** A symbol node represents a recognized chemical symbol.

**Definition 9 (Relation Node).** A relation node represents a parent-child relationship.

![Figure 7](image)

Figure 7 gives an example of the Chemical Expression Tree for the chemical equation ‘CH₄ + 2O₂ → CO₂ + 2H₂O’. In Figure 7, the root node is located at the top of the tree; the un-shaded nodes are symbol nodes, whereas the shaded nodes are relation nodes. All symbols represented by sibling symbol nodes have row relationship, and the non-sibling symbol nodes connected through a relation node form a parent-child relationship.

2) Related Symbol Identification. For chemical structural analysis, it is very important to determine the related symbol of the last recognized symbol obtained from the chemical symbol recognition process. Before discussing the related symbol identification process, we define baseline, related symbol and previous neighbor as follows.

**Definition 10 (Baseline).** A baseline of a structural expression is a virtual horizontal line on which the symbols lie. The symbols located next to each other on the same baseline have the row relationship. In addition, these symbols also have the same parent symbol.

**Definition 11 (Related Symbol).** A symbol A is a related symbol of the last recognized symbol B if A is written before B chronologically. And the two symbols satisfy either one of the following two conditions:

- A and B are two adjacent symbols on the same baseline of an expression.
- A and B have parent-child relationship which is indicated by a relation node R in CET.

**Definition 12 (Previous Neighbor).** The previous neighbor of a symbol is the symbol which was written just before it chronologically.

For example, in the chemical expression ‘SO₄²⁻+Ba²⁺ → BaSO₄’, there are five baselines, which are ‘SO+Ba → BaSO’, ‘2⁻’, ‘4’, ‘2+’, and ‘4’. Although ‘2⁻’ and ‘2+’ are aligned on the same horizontal line, their baselines are different since they have different parent symbols. In this research, we use baseline to help identify the related symbol of the last recognized symbol.

![Figure 8](image)

Algorithm 3 gives the proposed approach for related symbol identification. If the last recognized symbol is assigned to an existing baseline, the symbol that is closest to it in the same baseline will be its related symbol. Otherwise, it identifies the closest symbol to the last recognized symbol. To do this, it identifies all the symbols along the path from the root to the previous neighbor of the last recognized symbol (including the previous neighbor). These symbols will be put into a list of candidate symbols. If the candidate symbol list is not empty, it constructs an acute angle for
each candidate symbol by connecting the centers of the last recognized symbol, the closest symbol and the candidate symbol as shown in Figure 8. If the angle is greater than a prescribed threshold, then the candidate symbol will be removed from the candidate symbol list. In practice, we use 60 degree as the threshold, which is determined empirically. If the candidate symbol list is empty, the closest symbol is chosen as the related symbol. Otherwise, it checks the positions of the candidate symbols in the Chemical Expression Tree. The symbol which is located at the lowest level in the CET is then chosen as the related symbol.

Algorithm 3 Related Symbol Identification

Input:
- \( B \) – The last recognized symbol
- \( S = \{s_0, s_1, ..., s_i, ..., s_n\} \) – The chronologically ordered list of recognized symbols
- \( L = \{l_0, l_1, ..., l_i, ..., l_m\} \) – The elastic chemical symbol library
- \( T \) – The current Chemical Expression Tree

Output:
- \( A \) – The Related Symbol

Process:
1. \( l_B \leftarrow \text{getBaseline}(B, L) \)
   - If \( l_B \neq \text{null} \)
     - \( A \leftarrow \text{getClosestSymbol}(B, l_B, S) \)
   - Else
     - \( A \leftarrow \text{getClosestSymbol}(B, S) \)
     - \( \{C\} \leftarrow \text{getCandidateSymbols}(B, S, T) \)
     - If \( |C| > 0 \)
       - For each \( c_i \in \{C\} \)
         - \( \text{angle} \leftarrow \text{getAcuteAngle}(B, S, c_i) \)
         - If \( \text{angle} > 60^\circ \)
           - \( \{C\} \leftarrow \{C\} \setminus c_i \)
       - End if
     - End for
     - If \( |C| = 0 \)
       - \( A \leftarrow \text{getSymbolInLowestLevel}(\{C\}, T) \)
     - End if
   - End if
2. Return \( A \)

In Figure 9, it shows the handwritten expression ‘\((\text{NH}_4)_2\)’, where the user has just finished writing the symbol ‘2’. After the symbol ‘2’ is recognized, our proposed approach finds out that it does not belong to any existing baseline. Therefore, the closest symbol ‘\(’\) and the candidate symbol list that contains only the symbol ‘4’ are identified. Since the acute angle of ‘2’, ‘\(’\), and ‘4’ is greater than 60 degree, ‘4’ will be removed from the candidate symbol list. As a result, the closest symbol ‘\(’\) is considered to be the related symbol of ‘2’.

3) Symbol Relationship Identification. A geometrical approach based on the bounding box is proposed for Symbol Relationship Identification. As mentioned earlier, there are eight possible relationships between the last recognized symbol, say B, and its related symbol, say A, possibly left, right, superscript, subscript, above, under, pre-superscript or pre-subscript, relative to B. These relationships are identified based on the following rules:

- **Row Relationship**: If the difference between the Y-coordinates of the centers of A and B is less than 1/4 of their maximum width or height, and if the center of B is located on the left/right of the center of A, then B and A have the (left/right) row relationship.
- **Superscript/Subscript Relationship**: If more than 80% of B is located above/below the center of A and more than 20% of B lies on the right of A, then B has the superscript/subscript relationship with A.
- **Pre-superscript/Pre-subscript Relationship**: If more than 80% of B is located above/below the center of A and more than 20% of B lies on the left of A, then B has the pre-superscript/pre-subscript relationship with A.
- **Above/Under Relationship**: If more than 80% of B is located above/below the center of A and no more than 20% of B lies on the left/right of A, then B has the above/below relationship with A.

For example, in the handwritten expression shown in Figure 9, the last recognized symbol ‘2’ is 100% below the center of the related symbol ‘\(’\), and it also lies on the right hand side of the symbol ‘\(’\). So the last recognized symbol ‘2’ has a subscript relationship with its related symbol ‘\(’\).

The symbol relationship identification process computes the relationships based on the bounding box. However, as the size of the bounding box for some symbols such as ‘1’ and ‘4’ are not in square shape, the error rate could be high if we just use the bounding box to determine the symbol relationship. To resolve this problem, we transform the size of the bounding box of a non-square symbol to a square box size before the symbol relationship identification process.

4) Chemical Expression Tree Update. After identifying the symbol relationship for the last recognized symbol, this step updates the CET accordingly. If there is no related symbol, i.e., the last recognized symbol is the first symbol written by the user, the symbol will be appended as the first child node of the root node in the CET. On the other hand, if the last recognized symbol and its related symbol have the row relationship, then the last recognized symbol will be appended as the next sibling node of the related symbol node in the CET. Otherwise, a new relation node is created and appended to the related symbol node. The last
recognized symbol is then appended as the first child of this newly created relation node in the CET.

5) Displaying the Recognition Result. The last step of our proposed approach for chemical structural analysis is to convert the updated Chemical Expression Tree into a standard chemical representation such as Chemical Markup Language (CML) [18] or SMILES [19] for displaying the recognition result progressively. CML and SMILES are the two most commonly-used representations for chemical expressions. Since SMILES is easy to understand, it is currently adopted for displaying the recognition result to the user. Whenever there is an updated Chemical Expression Tree, a conversion is performed based on the updated CET and the corresponding SMILES is displayed to the user. In this way, the user can view the most updated recognition result immediately after he has written a new symbol. During the conversion process, multiple related symbols are also grouped together to form a meaningful chemical element. For example, the symbols ‘C’ and ‘a’ are grouped into a chemical element ‘Ca’ for Calcium.

3.4 System Implementation

We have implemented the proposed handwritten chemical expression recognition approach based on the Apple’s iOS platform. Figure 10 shows the user interface of the Handwritten Chemical Expression Recognition System. As shown in the figure, the top 10 candidate symbols are displayed for the last handwritten symbol. The user can select the correct symbol by clicking the corresponding candidate symbol. In addition, the user can also select the undo or clear option, and then rewrite the symbol. The recognition results are displayed in the SMILES representation.

4. Performance Evaluation

To evaluate the performance of the proposed progressive approach for handwritten chemical expression recognition, we have conducted experiments with 10 users as participants. The evaluation procedure is given as follows. First, we introduce the Handwritten Chemical Expression Recognition System to the users. Then, the users spend about 10 minutes to familiarize themselves with the system and try a few simple expressions which are different from those used in the testing.

4.1 Handwritten Chemical Symbol Recognition

Table I shows the test data on the chemical symbols used in the experiment. Each participant is required to write all the test chemical symbols for 3 times. Table II shows the performance results based on the SVM, Elastic Matching (EM) and Hybrid SVM-EM approaches. The performance of SVM and Elastic Matching is evaluated for comparison purpose. Precision@1 (P@1) is used for evaluating the performance. In addition, the performance results of the proposed hybrid SVM-EM approach based on Precision@3 (P@3) and Precision@5 (P@5) are also given. Precision@n reports the recognition accuracy that the correct symbol is in the top-n candidate symbols. As shown in Table II, the proposed hybrid SVM-EM symbol recognition approach has achieved the average recognition accuracy of 89.7% which performs better than that of using only the SVM or EM. The performance results based on Precision@3 and Precision@5 for the proposed hybrid approach are 94.1% and 94.6% respectively.

<table>
<thead>
<tr>
<th>Category</th>
<th>Symbols</th>
<th>Number of Symbols</th>
<th>Number of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digits</td>
<td>0 1 2 3 4 5 6 7 8 9</td>
<td>10</td>
<td>300</td>
</tr>
<tr>
<td>Alphabetical Character</td>
<td>ABCDEFGHIJKLMNOPQRSTUVWXYZ</td>
<td>52</td>
<td>1560</td>
</tr>
<tr>
<td>Bonds Operators</td>
<td>a = e = s → 1 d / A^+ / ^(-)()[ ]</td>
<td>16</td>
<td>480</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>78</td>
<td>2340</td>
</tr>
</tbody>
</table>

Table II. Performance Results for Chemical Symbol Recognition

<table>
<thead>
<tr>
<th>Category</th>
<th>SVM</th>
<th>EM</th>
<th>Hybrid SVM-EM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P@1</td>
<td>P@1</td>
<td>P@1</td>
</tr>
<tr>
<td>Digits</td>
<td>87.30%</td>
<td>81.30%</td>
<td>91.30%</td>
</tr>
<tr>
<td>Alphabetical Characters</td>
<td>83.10%</td>
<td>75.20%</td>
<td>88.60%</td>
</tr>
<tr>
<td>Bonds Operators</td>
<td>89.60%</td>
<td>78.10%</td>
<td>91.90%</td>
</tr>
<tr>
<td>Average</td>
<td>85.10%</td>
<td>76.70%</td>
<td>94.10%</td>
</tr>
</tbody>
</table>
4.2 Chemical Structural Analysis

Table III. Test Data on Chemical Expressions

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
<th>No. of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Equation</td>
<td>CH₄+2O₂→CO₂+2H₂O</td>
<td>60</td>
</tr>
<tr>
<td>Bidirectional</td>
<td>2SO₃+O₂→2SO₃</td>
<td>40</td>
</tr>
<tr>
<td>Ionic Equation</td>
<td>Ag⁺ +Cl⁻→AgCl</td>
<td>60</td>
</tr>
<tr>
<td>Complexes</td>
<td>[Co(H₂NCH₂CH₂NH₂)₂]₃(SO₄)</td>
<td>40</td>
</tr>
<tr>
<td>Nuclear Equation</td>
<td>¹³He⁺ ⁴He⁺→²⁵He+2p</td>
<td>40</td>
</tr>
<tr>
<td>Simple Bonds</td>
<td>H – C ≡ C = H</td>
<td>40</td>
</tr>
<tr>
<td>Structural Formula</td>
<td>H – C – H + 2O₂ ↔ C = O + 2H₂O</td>
<td>40</td>
</tr>
<tr>
<td>Others</td>
<td>⁰Al⁺⁺e⁻→¹⁰Mg⁺⁺νe</td>
<td>60</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>380</td>
</tr>
</tbody>
</table>

Table III shows the test data on different typical chemical expressions used in the experiment. The test chemical expressions can be classified into the following categories: basic equations, bidirectional equations, ionic equations, complexes, nuclear equations, simple bonds, structural formulae and others. In the experiment, each user is required to write all the expressions based on the test chemical expressions. Each expression can be written only once. In addition, we also assume that all the symbols can be recognized correctly as user feedback mechanism is supported in the chemical symbol recognition process. As such, we can focus on evaluating the performance based on the proposed structural analysis approach.

Table IV. Performance Results for Chemical Structural Analysis

<table>
<thead>
<tr>
<th>Category</th>
<th>Related Symbol Identification</th>
<th>Symbol Relationship Identification</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Equations</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Bidirectional Equations</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Ionic Equations</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Complexes</td>
<td>92.90%</td>
<td>92.90%</td>
<td>92.90%</td>
</tr>
<tr>
<td>Nuclear Equations</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Simple Bonds</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Structural Formulae</td>
<td>96.70%</td>
<td>96.30%</td>
<td>96.30%</td>
</tr>
<tr>
<td>Others</td>
<td>90.30%</td>
<td>87.10%</td>
<td>87.10%</td>
</tr>
<tr>
<td>Average</td>
<td>97.30%</td>
<td>96.80%</td>
<td>96.80%</td>
</tr>
</tbody>
</table>

The performance results of the proposed chemical structural analysis approach are given in Table IV. The average accuracy for related symbol identification and symbol relationship identification are 97.3% and 96.8% respectively. The overall accuracy for the proposed chemical structural analysis approach is 96.8%. The accuracy for basic equations, bidirectional equations, ionic equations, nuclear equations and simple bonds are 100% as the related symbols and symbol relationships in these expressions are relatively simple. Note that if an error occurs in the related symbol identification process, it may further affect the performance of the symbol relationship identification and the overall structural analysis approach. The failure cases in the experiments are due to several factors such as inconsistent symbol distance and symbol sizes. Both the related symbol identification and symbol relationship identification processes rely on the symbol distance for its performance. Handwritten chemical expressions with good symbol distance can enhance the accuracy. And, inconsistent symbol sizes can adversely affect the performance of the chemical structural analysis process.

5. Conclusion

In this paper, we have proposed an effective approach for online recognition of handwritten chemical expressions. In the proposed approach, a hybrid SVM-EM approach is proposed for chemical symbol recognition while a progressive approach is proposed for chemical structural analysis. The proposed progressive handwritten recognition approach is more natural, efficient and intuitive for inputting chemical expressions. The performance of the proposed approach has been evaluated which has shown promising results with high accuracy. The proposed handwritten chemical recognition approach has been implemented on Apple’s iPads which are one of the most popular touch-based devices. Currently, we are extending our proposed approach for the recognition of more complex handwritten structural expressions involving ring structures and connected bonds.

References


Peng Tang received the B.Eng. degree in Electrical and Electronic Engineering with First Class Honours from Nanyang Technological University, Singapore, in 2010. He is currently pursuing the PhD degree in Computer Engineering at Nanyang Technological University. His research interests are in chemical expression recognition and retrieval.

Siu Cheung Hui is currently in the Division of Information Systems, School of Computer Engineering since 1990. He received his B.Sc. degree in Mathematics in 1983 and D. Phil in 1987 from the University of Sussex, UK. He worked in IBM China/Hong Kong Corporation as a system engineer from 1987 to 1990. His current research interests include data mining, Web mining, information retrieval, natural language processing, mathematical data mining, intelligent tutoring systems and Semantic Web.

Chi-Wing Fu joined the school of computer engineering at the Nanyang Technological University as an assistant professor from 2008. He received his B.Sc.(1st Hon) and M.Phil. in Computer Science and Engineering from the Chinese University of Hong Kong in 1997 and 1999, respectively, and his PhD in Computer Science from Indiana University in Bloomington in December, 2003. In 2005, he received the IEEE Transactions on Multimedia Prize Paper Award (an annual award from the IEEE Signal Processing Society) for his paper published in the transaction in 2003. His research interests include tile-based modeling and rendering methods, image-based rendering, texture synthesis, and large-scale astrophysical visualization. He has published papers in top venues, including ACM SIGGRAPH, ACM SIGGRAPH ASIA, ACM CHI (SIGCHI), and IEEE Visualization.
Semantic Specialization: Specialization Using Explicit Specification and Semantic Information

Teruhisa Hochin\textsuperscript{1} and Hiroki Nomiya\textsuperscript{2}
Kyoto Institute of Technology, Japan

Abstract

This paper formally describes the semantic specialization. This specialization makes it possible for a shape graph, which corresponds to a relation schema in the relational data model, to have shape graphs and edges which are different from those of the original shape graphs, but are semantically related to them. Viewpoints (relationship lattices, respectively) are introduced as lattices of concepts for shape graphs (edges) of shape graphs. The specialized shape graphs (edges) are specified as common descendants of original shape graphs (edges) in viewpoints (relationship lattices). The mappings between the elements of the original shape graphs and the elements specialized from them are specified in the correspondences. In order to enable to specify only the correspondences truly required, the concept of correspondence extension is introduced. It is shown that usability of the semantic specialization can be improved by using the correspondence extension. It is also shown that the inner specialization, which is the multiple inheritance of the elements in a shape graph, can be specified.

Keywords: Specialization, semantic, data model, graph, inheritance, explicit specification.

1. Introduction

In recent years, various kinds of knowledge have been represented, gathered, and used around us according to the advances of computers and computer networks. Wikipedia is an encyclopedia collaboratively created over the Internet. It gathers the knowledge of many people. The conceptual descriptions of web resources have been represented in the Resource Description Framework (RDF), which is a kind of semantic network. By using these descriptions, web resources could effectively be manipulated. These descriptions represent the knowledge of web resources.

The contents of multimedia data has also been represented with directed labeled graphs, which could be captured as a kind of semantic network. Petrakis et al. have proposed the representation of the contents of the medical images by using directed labeled graphs [1]. Uehara et al. have used the semantic network in order to represent four components and the relationships between them [3]. Contents of video data are represented with a kind of tree structure in XML [4].

We have also proposed a graph-based data model, the Directed Recursive Hypergraph data Model (DRHM), for representing the contents of multimedia data [5-7]. It incorporates the concepts of directed graphs, recursive graphs, and hypergraphs. An instance graph is the fundamental unit in representing an instance in the form of a graph. A collection graph is a graph having instance graphs as its components. A shape graph represents the structure of the collection graph. Moreover, a schema graph has been introduced to represent specialization and generalization relationships in DRHM [7]. The schema graph enables us to systematize and reuse knowledge.

Moreover, the semantic generalization has been proposed [8]. In the semantic generalization, the elements of the shape graphs, which are not the same, but are semantically related, can be generalized. The element generalized is the lowest common ancestor of the elements in a hierarchy of concepts. Viewpoints are introduced as hierarchies of concepts. Relationship trees are also introduced as hierarchies of relationships for edges. The semantic generalization has been extended for a common ancestor of the elements to become a semantically generalized element [10]. Viewpoints were extended to have lattice structure. Relationship lattices were introduced instead of relationship trees [10].

We have also proposed the semantic specialization [9]. In the semantic specialization, elements can be inherited from the original shape graphs as different elements. Although these are different from those of the original ones, these are semantically related to them. Owing to the semantic specialization, the specialized elements could have appropriate names while those elements are not independent of the elements of the original shape graphs, but are related to them. The semantically related elements are handled as if these were of the original shape graphs because the specialized elements are related to the original ones. Therefore, when the original shape graphs are the targets of retrieval, the specialized elements could also be the targets of the retrieval. The definition of the semantic specialization is, however, descriptive [9]. Although the procedure may be simple, theoretical aspect is not precisely described. Data types are not specifically described, either [9]. The definition had better include the treatment of data types.

\textsuperscript{1}Department of Information Science
Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan
hochin@kit.ac.jp

\textsuperscript{2}Department of Information Science
Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan
nomiya@kit.ac.jp
This paper formally describes the semantic specialization. The treatment of data types is included in the definition. The concept of correspondence extension is introduced in order to improve usability of the semantic specialization. It is shown that inner specialization, which is the multiple inheritance of the elements in a shape graph, is possible.

This paper is organized as follows: Section 2 explains DRHM. Section 3 explains the semantic specialization with an example. And then, it is formally described in Section 4. Section 5 improves usability of the semantic specialization. Section 6 describes inner specialization. The proposed framework is compared with the conventional ones in Section 7. The merits of the semantic specialization are also described. Lastly, Section 8 concludes this paper.

2. Directed Recursive Hypergraph Data Model

The structure of DRHM is informally described through examples. The formal definition is included in the previous work [6]. In DRHM, the fundamental unit in representing data or knowledge is an instance graph. It is a directed recursive hypergraph. It has a label composed of its identifier, its name, and its data value. It corresponds to a tuple in the relational model. Please refer to our paper [6] for the formal definition.

Consider the representation of the contents of the picture shown in Fig. 1. In this picture, a butterfly is on flowers. Figure 2 represents the contents of this picture in DRHM. In Fig. 2, an instance graph is represented with a round rectangle. For example, g1, g11, g12, and g13 are instance graphs. An edge is represented with an arrow. When an arrow has more than one element as a head or tail of the arrow, the elements are surrounded with a broken curve. For example, g11 is connected to g12 and g13 by the edge e11. An instance graph may contain instance graphs and edges. For example, g1 contains g11, g12, g13, and e11. These are called the constructing elements of g1. As a constructing element can include another element, the order of the inclusion can be considered. The instance graph g11 is called the first constructing elements of g1. If g11 contains another shape graph, say g111, g111 is called the second constructing element of g1. The i th constructing elements of an instance graph g is denoted as \( V_{ce}(g) \). Please note that \( V_{ce}(g) \) contains edges as well as instance graphs.

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. It corresponds to a relation in the relational model. An example of a collection graph is shown in Fig. 3. A collection graph is represented with a dashed dotted line. It has a unique name in a database. The name of the collection graph shown in Fig. 3 is "Picture." The instance graph g1 is the one shown in Fig. 2.
The instance graph $g_2$ is for 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*. It corresponds to a relation schema in the relational model. The collection graph, whose structure the shape graph represents, is called its *corresponding collection graph*. Figure 4 shows the shape graph for the collection graph "Picture" shown in Fig. 3. It represents that an instance graph "picture" includes an instance graph "object," and an instance graph "object" is connected to "object" by an edge "pos."

The formal definition of the shape graph is presented here for the use in defining the semantic specialization. Definitions of the other graphs are included in the previous work [6].

**Definition 1** A shape graph $sg$ is a 9-tuple $(nm_{sg}, V_s, E_s, L_{vs}, \varphi_{vs}, \varphi_{es}, \varphi_{connects}, \varphi_{comps})$, where $nm_{sg}$ is the name of the shape graph $sg$, $V_s$ is a set of shape graphs included in $sg$, $E_s$ is a set of shape edges, $L_{vs}$ is a set of labels of the shape graphs, $Le_s$ is a set of labels of the shape edges, $\varphi_{vs}$ is a one-to-one mapping from the set of the shape graphs to the set of labels of the shape graphs $(\varphi_{vs} : V_s \rightarrow L_{vs})$, $\varphi_{es}$ is a one-to-one mapping from the set of the shape edges to the set of labels of the shape edges $(\varphi_{es} : E_s \rightarrow L_{vs})$, $\varphi_{connects}$ is a partial mapping representing the connections between sets of shape graphs $(\varphi_{connects} : E_s \rightarrow 2^{Vs \times Vs})$, and $\varphi_{comps}$ is a partial mapping representing the inclusion relationships $(\varphi_{comps} : V_s \cup \{sg\} \rightarrow 2^{Vs \cup Es})$. A label of a shape graph or a shape edge is called a *shape label*. It is a triple of an identifier, a name, and a set of data types.

A *schema graph* defines specialization or generalization relationships between shape graphs. A schema graph uses shape graphs as its nodes. A specialization (generalization, respectively) relationship is represented with an arrow (a broken arrow). An example of a specialization relationship is shown in Fig. 5. It is shown that the shape graphs "Student" and "Teacher" are specialized from the shape graph "Person." The shape graphs "Student" and "Teacher" have the elements of the shape graph "Person." In the specialization, "Person" is called the original shape graph, and the shape graphs "Student" and "Teacher" are called the specialized ones. The description of the generalization relationships is omitted because it is not related to the topics of this paper. The direction of the edge of the generalization relationship is opposite to that of the specialization one [7].

Shape graphs form a layer structure through specialization relationships and generalization ones. This structure is called the *shape graph lattice*. It is permitted for the collection graphs corresponding to the shape graphs only at the lowest layer to have instance graphs. Let the shape graph "S" be one level higher than the shape graph "A" through a specialization relationship. For example, "Person" is one level higher than "Student" and "Teacher" shown in Fig. 5. The shape graph "S" has to be connected to the shape graph whose name is "S" with a generalization relationship as well as "A." The shape graph "S" has the same elements as those of "S." This shape graph "S" is called an *exception shape graph* on "S." The collection graph corresponding to the shape graph "S" has the instance graphs which are not included in the collection graphs whose shape graphs are at the same level in the shape graph lattice as that of the shape graph "A."

Moreover, the *data type tree* has been introduced in order to generalize data types [7]. It is a tree representing the hierarchy of data types. An example of a part of a data type tree is shown in Fig. 6. The data type *Number* is a generalized one of the data types *Int*, *Float*, *Double*, and so on. The root of the tree is *Value*, which is for all of data types. The data type of the element, whose original elements have different data types, is the one which is of the lowest common ancestor of these data types in a data type tree.

Specialization permits the multiple inheritance. That is, more than one shape graph can be original shape graphs for a specialized one. An example of the multiple inheritance is shown in Fig. 7. The shape graph "TA" has two shape graphs "Student" and "Teacher" as the original shape graphs. In this case, shape graph "TA" has four elements named "name." Three elements are those inherited from the shape graphs "Person," "Student," and "Teacher." The other is the specific element of the shape graph "TA."

![Figure 5. An example of specialization relationships.](image)

![Figure 6. An example of a datatype tree.](image)
3. Description of Semantic Specialization

Let us consider the situation that we have the shape graph "Ball_game" shown in Fig. 8. This shape graph represents the contents of pictures of ball games by using the notations of conceptual graphs [15]. Let us create the shape graph "Tennis" shown in Fig. 9 by using the shape graph "Ball_game." Please note that the element "tennis_player" in the shape graph "Tennis" corresponds to the element "player" in the shape graph "Ball_game." Similarly, the element "tennis_ball" in the shape graph "Tennis" corresponds to the element "ball" in the shape graph "Ball_game." These elements would like to be the specialized ones from the shape graph "Ball_game." This is, however, impossible in the conventional specialization.

Semantic specialization makes this kind of specialization possible. In the semantic specialization, viewpoints are used. A viewpoint has a lattice structure. It represents general – concrete relationships of concepts. Examples of viewpoints are shown in Fig. 10. Four viewpoints are shown in Fig. 10. For example, the concept "player" is more general than the concepts "soccer_player" and "tennis_player." A viewpoint may have only one node. The concept "scene" is an example of this viewpoint. This viewpoint is called a trivial viewpoint.

In the semantic specialization, relationship lattices are also used. A relationship lattice is shown in Fig. 11. The relationship "relationship" is more general than the other relationships: "connect," "inst," "agnt," and "target."
Correspondences are specified in the semantic specialization. The form of a correspondence is a quadruple \( <OE, se, vp, dt> \), where \( OE \) is a set of elements of the original shape graphs, \( se \) is an element of a specialized shape graph, \( vp \) is a viewpoint or a relationship lattice, and \( dt \) is a data type.

The correspondences of the semantic specialization of the example described here are shown in Fig. 12. The fourth element “String” means the data type of the character string. The first five correspondences are for shape graphs, while the other three ones are for shape edges. In the correspondences \( c1, c2, c6, \) and \( c7 \), the names of the elements of the specialized shape graph are the same as those of the original one. These elements are inherited from the original shape graph as in the conventional specialization. The correspondences are called trivial ones. The first items of the correspondences \( c5 \) and \( c8 \) are empty sets. These are for the specific elements of the specialized shape graph as in the conventional specialization. The names of the elements of the specialized shape graph are different from those of the original one in the correspondences \( c3 \) and \( c4 \). These elements are related to those of the original shape graph in the viewpoints specified, i.e., "player" and "ball," which are shown in Fig. 8. These are the elements inherited from the original shape graph through the semantic specialization.

Correspondences guarantee the legality of specialized shape graphs. They also represent the mapping from the original shape graphs to the specialized one. Specifying these correspondences is, however, considered to be cumbersome. In our example, the correspondences \( c1, c2, c6, \) and \( c7 \) are not required to be specified because these are the same as the elements of the original shape graphs. In Section 5, we will examine the omission of these trivial correspondences.

The schema graph including the shape graphs named "Ball_game" and "Tennis" is shown in Fig. 13. The relationship between an original element and a semantically specialized one is represented with a thin arrow. The thin arrows for edges are omitted because of simplicity. These relationships are obtained from correspondences.

The semantic specialization permits the multiple inheritance as in the conventional specialization. That is, more than one shape graph can be original shape graphs for a specialized one. In the example shown in Fig. 7, the shape graph "TA" has two shape graphs "Student" and "Teacher" as the original shape graphs. In this case, for example, the correspondence \( \langle\{\text{Student.person.name, Teacher.person.name}\}, \text{TA.person.name, name, String}\rangle \) is required for the element "name," where the existence of the viewpoint "name" is assumed.

4. Definition of Semantic Specialization

4.1 Viewpoints and Relationship Lattices

We begin with the definition of a viewpoint. It is defined as follows.

Definition 2 A viewpoint is a lattice whose nodes represent concepts. A parent node represents more general concept than a child node. There are no two nodes representing the same concept. A viewpoint having only one node is called a trivial viewpoint.

Next is a relationship lattice. The following is the definition of the relationship lattice.

Definition 3 A relationship lattice is a lattice whose nodes have the distinct names of relationships. A parent node represents more general relationship than a child node.

4.2 Preliminaries

In defining the semantic specialization, the notations shown in Table 1 are used. In Table 1, an element means a shape graph or edge.

The atomic formula \( \text{ComDesc}(la, L, lbl) \), where \( la \) is a lattice, \( L \) is a set of labels in the lattice \( la \), and \( lbl \) is a label, returns true if the label \( lbl \) is a common descendant of all of the labels in \( L \) in the lattice \( la \). If not, it returns false.

4.3 Semantic Specialization

By using the definitions of viewpoints and relationship lattices, the semantic specialization is defined as follows.
Definition 4 Let consider $SG_{init}$, $T$, $P_{init}$, $P_{term}$, $W$, $Corp$, $Q_{init}$, $Q_{term}$, $R$, and $Cor_e$ such that $SG_{init}$ is a set of original shape graphs, $T$ is a data type tree, $P_{init}$ is a set of paths of the elements of original shape graphs, $P_{term}$ is a set of paths of the elements of a specialized shape graph, $W$ is a set of viewpoints, $Corp : 2^{P_{init}} \times P_{term} \rightarrow (W, T)$ is a set of correspondences for shape graphs, $Q_{init}$ is a set of the edges in original shape graphs, $Q_{term}$ is a set of the edges in a specialized shape graph, $R$ is a set of relationship lattices, and $Cor_e : 2^{Q_{init}} \times Q_{term} \rightarrow (R, T)$ is a set of correspondences for edges. The semantic specialization creates a shape graph $sg_{term} = (nm_{sp}, V_s, E_s, L_v, L_e, \varphi_{var}, \varphi_{con}, \varphi_{com})$ satisfying all of the following conditions:

1) $\forall c_i \in Corp, ComDesc(vp(c_i), nm(tail(se(ci))), nm(tail(\text{OE}(ci))))$
2) $\forall c_i \in Corp, ComDesc(T, dt(tail(se(ci))), dt(tail(\text{OE}(ci))))$
3) $L_v = \{ l | nm(l) = nm(tail(se(ci))), dt(l) = dt(c_i), c_i \in Cor_e \}$
4) $V_s = \{ v | nm(\varphi_{var}(v)) = nm(tail(se(ci))), c_i \in Cor_e \}$
5) $\forall v_i \in V_s, \exists l \in L_v, (\varphi_{con}(v_i) = l)$
6) $\forall c_i \in Corp, \exists v_i \in V_s (\varphi_{comp}(\text{tail2}(se(ci))) = v_i)$
7) $\forall c_i \in Corp, ComDesc(rl(c_i), nm(tail(se(ci))), nm(tail(\text{OE}(ci))))$
8) $\forall c_i \in Corp, ComDesc(T, dt(tail(se(ci))), dt(tail(\text{OE}(ci))))$
9) $L_e = \{ l | nm(l) = nm(tail(se(ci))), dt(l) = dt(c_i), c_i \in Cor_e \}$
10) $E_s = \{ e | nm(\varphi_{con}(e)) = nm(tail(se(ci))), c_i \in Cor_e \}$
11) $\forall e_i \in E_s, \exists l \in L_e, (\varphi_{com}(e_i) = l)$
12) $\forall c_i \in Cor_e, \exists e_i (\varphi_{con}(\text{tail}(\text{OS}(ci)))) = e_i$
13) $\forall c_i \in Cor_e, \exists e_i (\varphi_{comp}(\text{tail2}(se(ci))) = e_i$

Then, the original shape graphs are connected to $sg_{term}$ with edges in the schema graph. A set of the exception shape graphs $SG_{init} \text{exc}$ corresponding to $SG_{init}$ is created. Each original shape graph is connected to the exception shape graph corresponding to it, and being a member of $SG_{init} \text{exc}$ with an edge in the schema graph.

The latter two steps are the same as those of the conventional specialization [7].

The condition 1 (the condition 7, respectively) says that a specialized shape graph (edge) must be a common descendant of the original shape graphs (edges) in the viewpoint (relationship lattice), or the same as the original ones. Similarly, the condition 2 (the condition 8, respectively) says that the data type of a specialized shape graph (edge) must be the data type of original shape graphs (edges) or the one corresponding to a common descendant of the data types of original shape graphs (edges) in $T$. The conditions 3 to 6 (the conditions 9 to 13, respectively) describe the shape graphs (edges) created by semantic specialization. Although many conditions are used, the creation of shape graphs and edges is straightforward: the shape graphs and the edges specified as the second elements of correspondences are created.

In the previous work [7], the function $SS_{basic}$, where $SS$ stands for "Semantic Specialization," has been introduced to create a specialized shape graph. This function confirms the correspondences given. It checks whether the shape graphs (the edges and the data types, respectively) specified as the original and the specialized ones in correspondences exist in a viewpoint (a relationship lattice and a data type tree), or not. This function also checks that specialized shape graphs (edges, respectively) are common descendants of the original shape graphs (edges) in the viewpoints (relationship lattices), or the same ones as the original ones, and that the data types of specialized shape graphs (edges) are those of original shape graphs (edges), or the ones corresponding to common descendants of the data types of original shape graphs (edges) in the data type tree. When these checks are passed, specialized shape graphs (edges) are created.

5. Improving Usability of Semantic Specialization

Until here, all of the correspondences of the elements of a specialized one are assumed to be specified. An example of a correspondence of a shape graph in the conventional specialization is a quadruple $\langle\text{Person.person.name}\rangle$. Teacher.person.name, name, String$. This specification is cumbersome because the name of the element of the specialized shape graph is the same as that of the original one. This name could easily be obtained from the one of the original shape graph.

The function $SS_{ext}$ has been introduced [9] to identify the elements of the original shape graphs, which are not specified in the correspondences, to create the correspondences for these elements, and to invoke the function $SS_{basic}$. That is, this function creates the
correspondences for the elements not specified in the correspondences.

The algorithm of the creation of correspondences of the function $SS_{ext}$ consists of two steps. The first step is for the multiple inheritance. When an element, which is not specified in the correspondences, of each original shape graph appears in a viewpoint, the elements constitute a set of elements of the original shape graphs, which is the first term of a correspondence. The greatest common descendant of the elements in a viewpoint becomes the element of the specialized shape graph, which is the second term of a correspondence. The viewpoint used is the third term of a correspondence. The greatest common descendant of the data types of the elements in a data type tree becomes the corresponding data type. The viewpoint used is the third term of a correspondence. The greatest common descendant of all of the labels is the viewpoint, the elements constitute a set in the lattice $\mathcal{L}$. For the success of the correspondence extension, it is required that the greatest common descendant of the elements is uniquely decided. If there are two or more greatest common descendants, correspondences could not successfully be extended. When a viewpoint is a tree, correspondence extension always succeeds because there is no multiple inheritance. In this case, $Cor_{p}^{extM}$ and $Cor_{e}^{extM}$ are empty sets. The conditions a1 and a3 become meaningless.

The function $SS_{ext}$ confirms this condition. That is, it checks whether there are two or more greatest common descendants, or not. If so, it stops the procedure.

By using the extended correspondences $Cor_{p}^{ext}$ ($Cor_{e}^{ext}$, respectively), $P_{init}$ and $P_{term}$ ($Q_{init}$ and $Q_{term}$) are easily extended such that $P_{init}^{ext} = OE(Cor_{p}^{ext})$ and $P_{term}^{ext} = se(Cor_{p}^{ext})$ ($Q_{init}^{ext} = OE(Cor_{e}^{ext})$ and $Q_{term}^{ext} = se(Cor_{e}^{ext})$). By using $P_{init}^{ext}$ and $P_{term}^{ext}$, $Cor_{p}^{ext}$, $Q_{init}^{ext}$, $Q_{term}^{ext}$, and $Cor_{e}^{ext}$, a semantically specialized shape graph is created as shown in Definition 4. In this way, the function $SS_{ext}$ can be realized by using the correspondence extension.

### 6. Inner Specialization

In the semantic specialization, the multiple inheritance can be applied to the elements of a shape graph [11], which correspond to attributes in the relational model. An example of this kind of inheritance is shown in Fig. 14. The shape graph "ExerciseClass" is the one specialized from the shape graph "SchoolClass." The element "TA" of "ExerciseClass" is specialized from the elements "teacher" and "student" of "SchoolClass." The correspondence for this shape graph "TA" is $\langle$SchoolClass.class.teacher, SchoolClass.class.student$, ExerciseClass.class.TA, person, String$\rangle$, where "person" is the viewpoint whose structure is of the schema graph shown in Fig. 7. This correspondence is shown as the thin arrow from the element "teacher" to the one "TA" and the one from the element "student" to the one "TA." This kind of inheritance is called the inner specialization. In Fig.

![Figure 14. An example of an inner specialization.](image-url)
14, the element "teacher" ("student," respectively) of the shape graph "SchoolClass" is also connected to the element "teacher" ("student") of the shape graph "ExerciseClass" with a thin arrow. These correspondences, which are shown as thin arrows in Fig. 14, are required because the elements "teacher" and "student" of the shape graph "SchoolClass" are used by the inner specialization of "TA," and these are not implicitly inherited to the shape graph "ExerciseClass." If these correspondences are not specified in invoking the function SSext, the elements "teacher" and "student" will not appear in the specialized shape graph "ExerciseClass."

7. Related Works

7.1 Specialization

Specialization is an important mechanism in conceptualizing the real world [12, 13, 14]. It can be used to define possible roles for members of a given type. Specialization has been introduced in many data and knowledge models, and programming languages. The entity-relationship (ER) model supports ISA relationships for specialization and generalization [12]. The IFO model introduces two kinds of ISA relationships: specialization and generalization relationships [13].

In the conventional specialization, the attributes of a type/class are inherited downward to the specialized type/class. For example, when the type "Dog" is defined as a specialized one of the type "Animal," all of the attributes of the type "Animal" are inherited to the type "Dog." That is, the type "Dog" has all of the attributes of the type "Animal." The type "Animal" is called the supertype, while the type "Dog" is called the subtype. Moreover, the subtype "Dog" could have its own attributes.

The specification of the conventional specialization can be said to be implicit. That is, the attributes of the supertype are automatically inherited to the subtypes. Those names are the same as those of the supertype. On the other hand, the specification of the semantic specialization is said to be explicit. The attributes of the supertype are explicitly specified. Those names can be different from those of the supertype.

The conventional specialization can be realized only by specifying the correspondences for the specific elements of a specialized shape graph. This specification is the same as that in the conventional specialization. The burden of this specification is equal to that of the conventional specialization. In the semantic specialization, the elements, which are semantically related to those of the original shape graphs, can be added as the inherited ones. This kind of inheritance is not supported in the conventional specialization. The semantic specialization is said to be more powerful than the conventional one. The conventional specialization is considered to be a special case of the semantic one.

7.2 Retrieval

Let us consider the situation that the shape graph "Tennis" shown in Fig. 9 is created through the conventional specification. The elements "tennis_player" and "tennis_ball" are defined as the specific elements of the specialized shape graph "Tennis." In this case, the shape graph "Tennis" has also the elements "player" and "ball" as the inherited elements. Please note that the shape graph "Tennis" has two elements on the player: "player" and "tennis_player." As the element "tennis_player" is the one that a user defined, the information on the player is inserted into this element. There are two cases whether the information is inserted into the element "player." The first case, say Case A, is that the information is inserted into it. The second case, say Case B, is that the information is not inserted into it. The same thing can be said for the elements "ball" and "tennis_ball."

When we retrieve some information on the shape graph "Ball_game," the targets of the retrieval are the shape graphs "Ball_game," "Tennis," and "Ball_game." Some retrieval conditions on the element "player" and/or the element "ball" may be specified in the retrieval. In Case A, appropriate information is retrieved because data are inserted into the elements "player" and "ball" as well as "tennis_player" and "tennis_ball." This is, however, cumbersome because the same data must be inserted into both of the elements "player" and "tennis_player" ("ball" and "tennis_ball," respectively). As the element "tennis_player" ("tennis_ball," respectively) is also treated as the element "player" ("ball") in the semantic specialization, there is not such cumbersomeness. That is, what is needed is inserting data into the element "tennis_player" ("tennis_ball," respectively). These data are treated as if those were stored in the elements "player" ("ball")

In Case B, those conditions are meaningless for the shape graph "Tennis" because no value is stored in the elements "player" and "ball." Values are stored only in the elements "tennis_player" and "tennis_ball." This may result in false dismissal. In the semantic specialization, the element "tennis_player" ("tennis_ball," respectively) could be the target of the retrieval because it is managed with being related to the element "player" ("ball") of the original shape graph "Ball_game." This is the merit of the semantic specialization.

7.3 Inner Specialization

The semantic specialization enables the elements of a specialized shape graph to have the names different from
Although the semantic specialization was formally described in this paper, the semantic generalization, which has been proposed [8] and extended [10], has not formally been described. Formal description of the semantic generalization is in future work. Viewpoints and relationship lattices are very important in the semantic generalization and the semantic specialization. Showing the feasibility of viewpoints and relationship lattices is also in future work. The correspondence extension does not always succeed as described in Section 5. On the other hand, it always succeeds if a viewpoint is a tree. There may be several conditions for the success of the correspondence extension. Clarifying these conditions is in future work. The semantic specialization and the semantic generalization have two major characteristics. One is the explicit specification of the element specialized or generalized. For this purpose, the correspondence is introduced. The other is the usage of the conceptual networks. In this paper, viewpoints and relationship lattices are used as the conceptual networks. The name of the "semantic" specialization originates from the usage of the conceptual networks. Although we use a kind of conceptual network in this paper, we could take a method not using it. In this case, we could consider the specialization only with the explicit specification. This specialization, which may be named explicit specialization, is considered to be more general framework than the semantic specialization. Clarifying the concept of the explicit specialization and clarifying the relationships between the semantic specialization and the explicit one are also in the future work.

Acknowledgment

This research is partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B), 23300037, 2011-2014.

References

Sons, Ltd., 2002.


Teruhisa Hochin is a professor in the department of Information Science at Kyoto Institute of Technology. His research interests include graph-based, multimedia, scientific, and flexible databases, extensibility of DBMS, and Kansei retrieval. He is a member of IEEE, ACM, Institute of Electronics, Information and Communication Engineers (IEICE), Information Processing Society of Japan (IPSJ), Database Society of Japan (DBSJ), Japan Society for Archæological Information (JSAI), and Japan Society of Kansei Engineering (JSKE) as well as ACIS. He has given presentations at national and international conferences, symposiums, and workshops. He has reviewed papers for IEICE Transactions on Information and Systems, IPSJ Journal, IPSJ TOD, DBSJ Journal, Journal of JSAI, Kansei Engineering International Journal, Transactions of JSK, and Journal of Communication and Computer.

Hiroki Nomiya is an assistant professor in the department of Information Science at Kyoto Institute of Technology. His research focus is multimedia data engineering, image recognition and machine learning. He has been engaged in the research of multimedia data retrieval, especially emotional video scene retrieval using facial expression recognition based on machine learning. He has given presentations at national and international conferences and workshops. He is an active member of the Information Processing Society of Japan and the Japanese Society for Artificial Intelligence. He has reviewed papers for IEICE (the Institute of Electronics, Information and Communication Engineers) Transactions on Information and Systems.
Adapting CRM Systems for Mobile Platforms: An MDA Perspective
Federico Améndola¹
Liliana Favre²
Universidad Nacional del Centro de la Provincia de Buenos Aires, Comisión de Investigaciones Científicas de la Provincia de Buenos Aires, Argentina

Abstract

Mobile phones have become as powerful as any desktop computer in terms of applications they can run. However, the software development does not take advantage of the whole potential of this kind of devices. Enterprises are now adopting mobile technologies for numerous applications. A current problem in software engineering is the modernization of useful legacy systems to mobile platforms. In this context, we describe a reengineering process that integrates traditional reverse engineering techniques such as static and dynamic analysis with Model Driven Development (MDD), MDA (Model Driven Architecture) in particular. We describe a case study that shows how to move CRM (Customer Relationship Management) applications from desktop to mobile platforms. We validated our approach by using the open source application platform Eclipse, EMF (Eclipse Modeling Framework), EMP (Eclipse Modeling Project), the Atlas Transformation Language (ATL) and the Android platform.

Keywords: mobile computing, reverse engineering, reengineering, Model Driven Architecture, Customer Relationship Management

1. Introduction

Nowadays mobile devices come with their users all the time and everywhere. Among other novel features, mobile devices contain global positioning sensors, wireless connectivity, built-in web browsers and photo/video/voice capabilities that allow providing highly localized, context aware applications [7]. Mobile phones have become as powerful as any desktop computer in terms of applications they can run. However, the software development in mobile computing is still not as mature as it is for desktop computer and the whole potential of mobile devices is wasted [9].

Enterprises are now adopting mobile technologies for numerous applications to increase their operational efficiency and meet new customer demands. A current problem is the adaptation of legacy systems for mobile technologies. On the one hand, legacy systems resume key knowledge acquired over the life of an enterprise and, if they are business critical there is a high risk in replacing them. On the other hand, mobile technologies have changed the way in which enterprises do business and create value. A number of solutions can be considered such as redevelopment, which rewrites existing applications, or migration, which moves the existing system to a more flexible environment while retaining the original system data and functionality. A good solution should be to restore the value of the existing software, extracting knowledge and exploiting investment in order to migrate to new software that incorporates the new mobile technologies.

Traditional reverse engineering techniques can help in the software migration to mobile applications. They are related to the process of analyzing available software with the objective of extracting information and providing high-level views on the underlying code [4].

The rapid proliferation of different mobile platforms has forced developers to make applications tailored for each type of device. To achieve interoperability with multiple platforms the software community needs to adapt development approaches. Model Driven Development (MDD) is considered a promising approach to meet these challenges. It refers to a range of development approaches based on the use of software models as first class entities; the most well-known realization of MDD is the OMG standard Model Driven Architecture (MDA) [13]. The outstanding ideas behind MDA are separating the specification of the system functionality from its implementation on specific platforms, managing the software evolution from abstract models to implementations increasing the degree of automation of model transformations and achieving interoperability with multiple platforms. Models play a major role in MDA which distinguishes at least Platform Independent Model (PIM) and Platform Specific Model (PSM). The essence of MDA is the Meta Object Facility Metamodel (MOF) that allows different kinds of software artifacts to be used together in a single project [15].

The objective of this paper is to describe a reengineering process that allows moving existing desktop application to mobile platforms achieving interoperability with multiple platforms. Our research aims to simplify the creation of applications for mobile platforms by integrating traditional reverse engineering techniques, such static and dynamic analysis, with MDA. We analyze a case study on
modernization of a desktop CRM (Customer Relationship Management). To check the technical feasibility of our approach we choose the Eclipse open-source application platform, EMF (Eclipse Modeling Framework), EMP (Eclipse Modeling Project), the Atlas Transformation Language (ATL) and, the Android platform [2] [3] [10].

The paper is organized as follows. Section 2 presents background and related work. Section 3 describes the proposed reengineering process. Next, Section 4 includes a case study related to the modernization of desktop CRM applications. Finally, conclusions are included in Section 5.

2. Background and Related Work

Improvements in the development of business applications for mobile devices are emerging in recent years. In [22], authors present a new taxonomy of enterprise mobile applications. They classify mobile applications into five categories: mobile broadcast (applications that broadcast different kind of content to a large group of mobile users), mobile information (applications that primarily present user-request information, where the flow is usually from the application to the user), mobile transaction (applications that facilitate and execute transactions such as buy and sell goods and services, place and track orders and make electronic payments), mobile operation (applications that primarily support the operational aspects of a business without direct interaction with costumer and client) and mobile collaboration (applications that foster collaboration among employers and various functional units in an enterprise and with other enterprises of interest and business partners).

In [7], authors express that the fundamental challenges of mobile application software engineering involve creating user interfaces for different kinds of mobile devices providing reusable applications across multiple mobile platforms, designing context aware applications and handling their complexity and specifying requirements uncertainly. Authors remark in [19] that a critical aspect of developing future applications for mobile devices will be ensuring that the application provides sufficient performance while maximizing battery life.

Complex mobile applications require applying software engineering processes to assure the development of secure, high quality applications. Although developing mobile software is similar to software engineering for other embedded applications, mobile applications present some additional requirements. Reference [24] provides an overview of important software engineering research issues related to the development of applications that run on mobile devices. The author remarks that there is a mobile “angle” to almost every aspects of software.

Targeting an application for multiple markets, devices or operators has become a crucial problem. Today, mobile developing requires programmers that must know the very differences between the different platforms. Reference [12] presents a review of key concepts on Android, iPhone and Qt.

Cross-platforms development tools are uncommon yet. Some efforts are underway in this direction, for instance, haXe emerged as a general purpose programming language designed for helping developers that can use a single language and a set of libraries to deploy applications on multiple platforms effectively [6].

The proliferation of mobile devices over the last years generates the need to adapt existing desktop applications to mobile platforms. Software reengineering is a promising approach to support the modernization of these systems. Software reengineering starts from an existing implementation and requires an evaluation of every part of the system that could be transformed or implemented anew from scratch. This definition distinguishes the following main phases: “the examination and the alteration of a subject system to reconstitute it in a new form” and “the subsequent implementation in a new form” [8]. In other words, reengineering includes some form of reverse engineering followed by some form of forward engineering.

Reverse Engineering is the process of analyzing available software artifacts such as requirements, design, architectures, code or byte code, with the objective of extracting information and providing high-level views on the underlying system. Reverse engineering does not involve changing the source legacy systems, but understanding them to help reengineering processes that are concerned with their re-implementing.

The success of reengineering depends on the existence of CASE tools. In the context of MDD, the more relevant advances are linked to the Eclipse Modeling Framework (EMF) [10]. EMF was created for facilitating system modeling and the automatic generation of Java code. It started as an implementation of MOF, EMOF (Essential MOF) in particular. The EMF metamodel comparable to EMOF is ECORE. EMF has evolved starting from the experience of the Eclipse community to implement a variety of tools and to date is highly related to MDD. For instance, commercial tools such as IBM Rational Software Architect, Spark System Enterprise Architect or Together are integrated with Eclipse-EMF [5].

The MMT (Model-to-Model Transformation) Eclipse project is a subproject of the top-level Eclipse Modeling Project that provides a framework for model-to-model transformation languages. Transformations are executed by transformation engines that are plugged into the Eclipse Modeling infrastructure. The main transformation engines
developed in the scope of that project are ATL (Atlas Transformation Language) [3] and the OMG standard QVT [17].

ATL is a model transformation language in the field of MDD that is developed on top of the Eclipse platform. It is a hybrid language that provides a mix of declarative and imperative constructs.

Few MDA-based CASE tools support any of the QVT languages. As an example, IBM Rational Software Architect and Spark System Enterprise Architect do not implement QVT. Other tools partially support QVT, for instance Together allows defining and modifying transformations model-to-model (M2M) and model-to-text (M2T) that are QVT-Operational compliant. To date, the QVT declarative component is in its “incubation” phase and provides only editing capabilities to support the QVT language. Reference [23] analyzes performance in model transformations through experiments with ATL and QVT; the performance of the transformation engines of three languages (ATL, QVT and QVT Relations) is compared concluding that ATL is the fastest among the three engines.

OMG is involved in the definition of standards to successfully modernize existing information systems referred as Architecture-Driven Modernization (ADM) [1]. It depends on the existence of CASE tools that make a significant impact on the automation of processes involved in the modernization. The Eclipse-MDT MoDisco open source project is considered by ADM as the reference provider for implementations of several of its standards [14]. It is a reusable and extensible model-based framework applications. To date, MoDisco approach only support reverse engineering of class diagrams.

3. A Reengineering Process for Migrating Legacy Applications

We propose a reengineering process for modernizing desktop applications to mobile platforms (Fig. 1). First, the information is extracted out of the source desktop application. Second, this information is analyzed in order to take adequate modernization decisions and finally, the information is used to implement new mobile applications. These steps are supported by metamodels to describe existing systems, discoverers to automatically create models of these systems and, tools to understand and transform complex models created out of existing systems. Next, we summarize the description of the process into three steps: reverse engineering, model transformation and implementation.

Fig.1. The reengineering process.

3.1 Reverse Engineering

The proposed process starts from a source system and the application of reverse engineering. Reverse engineering techniques aim to support the understanding of a program by using source code as the main source of information about the organization and program behavior, and extracting a set of potentially useful views, expressed by models. Different techniques are involved to recover this information and generally, are based on two main types of analysis: structural or static analysis, and behavioral or dynamic analysis [20]. Traditionally, static and dynamic analyses have been viewed as two different techniques; today there are complementary approaches.

Static analysis examines software documentation (e.g., the source code text) and reasons over all possible behaviors that might arise at run time without executing program. Static analysis is based on classical compiler optimization techniques and abstract interpretation. The latter operates by constructing a model of the state of the program that is easier to manipulate (although it loses some information), and then analyzing how the program reacts to this state [11].

Dynamic analysis is based on observing actual, run-time program executions through testing and profiling. Extracting dynamic information is especially important when considering object-oriented software. This is due to the dynamic nature of object-oriented programs. Creating objects, deleting objects / garbage collector and dynamic binding can become very difficult (and sometimes impossible) understanding the behavior of the program by examining only its source code.

Reference [11] provides a comparison of static and dynamic analysis from the point of view of their synergy and duality. Author argues that static analysis is conservative and sound. Conservatism means reporting weak properties that are guaranteed to be true, preserving soundness, but not be strong enough to be useful. Soundness guarantees that static analysis provides an accurate
description of the behavior, no matter on what input or in what execution environment. Dynamic analysis is precise due to it examines the actual run-time behavior of the program; however the results of executions may not generalize to other executions. Also, reference [11] argues that whereas the chief challenge of static analysis is choosing a good abstract interpretation, the chief challenge of performing good dynamic analysis is selecting a representative set of test cases. A test can help to detect properties of the program, but it can be difficult identify whether results of a test are true program properties or properties of a particular execution context. The combination of static and dynamic analysis can enrich reverse engineering process. There are different ways of combination, for instance performing first static analysis and then dynamic one or perhaps, iterating through static and dynamic analysis. Likewise, the definition of appropriate heuristics may guide the search for information on the traces generated during the dynamic analysis.

Static and dynamic analysis support reverse engineering process (first stage of the reengineering process) and allow extracting artifacts in a high abstraction level that describe the application being analyzed.

3.2 Model Transformation

With the aim of reducing the effort and avoid the generation of errors when performing a model transformation, it has been attempted to increase the degree of automation of transformation processes. This automation has led to define different ways of carrying out the transformations, for example by general purpose languages, intermediate representations or transformation-specific languages. The last alternative proposes the use of specific processing languages to define how to generate a target model which conforms to a target metamodel, from a source model that conforms to a source metamodel.

At this point it is necessary to consider the dependencies of the recovered software artifacts on the technologies applied to implement the system under analysis. These dependencies should not cause an impact to the artifacts that describe the new system to be implemented. To avoid these situations is proposed the integration of reverse engineering techniques with MDD, MDA in particular. MDD aims interoperability between platforms and technologies independence proposing that all devices involved in a development process are represented from MOF.

The transformation model step allows representing the new system to be implemented. A metamodeling technique is used in this step. MOF metamodels are used to describe the transformations at model level. We consider PIM expressed in UML [21]. For each transformation, source and target metamodels are specified. A source metamodel defines the family of source models to which transformation can be applied. A target metamodel characterizes the generated models. There exists specific transformation languages that provides a way to specify how generate a target model that conforms to a target metamodel from a source model that conforms to a source metamodel, for example, we can mention QVT or ATL transformation languages. As a result of this step, models of target applications related to different mobile platforms are created.

3.3. Implementation

Next, in the implementation step, target applications are generated. Tools supporting forward engineering in the context of MDA are needed. A current problem in developing mobile applications is to deploy an implementation for multiple platforms. Developers prefer to implement an application once and deploy it to multiple platforms with minimal effort.

This step is guided by the artifacts recovered and the platform independent models constructed from such artifacts. Because of this, the quality and expressiveness of the artifacts will have a strong impact on the resulting implementation process.

Currently, there are no tools supporting a complete reengineering process as proposed by this work. However, various tools are available to deal with it. A more detailed description is included in the following sections.

4. A Case Study: Moving from Desktop CRM to Mobile Platforms

4.1 Application Domain: Customer Relationship Management

The emergence of mobile devices in business applications has improved care and has streamlined the customer relationship, besides greatly simplify the exchange of information between a client and a consultant. In this section we exemplify our approach with the modernization of a CRM software system developed to run over desktop computers. A CRM manages company interactions with current and future customers. Interactions are supported and guided by creating dynamic customer profiles that register information such as contracted services and products, frequent contact channels, and commercial transactions and their associated responses. Having rich customer profiles and good customer segmentation is the condition to achieve powerful CRM solutions.

With the advent of smartphones, CRMs have evolved from client-server applications to large Web applications (such as the case of Salesforce.com). Mobile CRM tries to
complement the existing CRM systems in the enterprise to make them mobile. We propose to analyze how to move CRMs to a mobile platform, Android platform in particular. Developing software for mobile devices requires more large effort compared to software development for desktop computers and servers. While mobile device support advanced features they have limited resources, such as battery capacity, screen size, use of primary memory and availability of development libraries.

We validate our approach by using the open source application platform Eclipse, EMF (Eclipse Modeling Framework), EMP (Eclipse Modeling Project). We select a set of appropriate tools that will be described in the following sections.

4.2 Description of the Source Application

The application that will be used as a case study is called SellWin [18]. It is a simple sales-oriented CRM that centers the data management on what it call opportunities. SellWin allows managing customer data, system users and sale opportunities. To illustrate the migration process, the analysis in this case study, will prioritize entities related to managing customer data. From the technical point of view, we can mention that SellWin is an open source application implemented entirely using Java. It uses Swing programming interface for the user interface and JDBC for database connections. The simple client-server architecture of the application follows a component-oriented design separated in different modules: Domain, Database, Server and User Interface.

Choosing SellWin is due to its simple features and the possibility of having an open source CRM system. Furthermore, SellWin lacks adequate documentation to understand its design, which allows us to analyze the strengths and weaknesses of the application of reverse engineering techniques for understanding its functionality. Following, the steps that need to be executed to transform the CRM application to the Android platform are described.

4.3. Application of Techniques for Recovering Designs

As mentioned above, we consider that only the source code is the repository of information for recovering the system design. Because of this, the first stage consists of applying different techniques of reverse engineering to generate UML diagrams.

4.3.1. Static Analysis

The initial step had to do with the recovering of class diagrams to detect relationships between the various components that make up the main modules. As an example, we show the class diagram of the Customer Management (Fig. 2). The purpose of this diagram is to visualize the relationships between the various modules. As we can see,
the user interface module is unrelated to the database, and the access to data is provided by the server module, with which it maintains a direct association via a defined interface. Moreover, the user interface is the only one that has direct associations with the domain, since both the server and database, have only registered dependencies according to the methods of the interface of each class.

The explorer tool integrated with the Eclipse development environment, called UML ObjectAid [16], was used in this step. ObjectAid is a free tool for working with class diagrams but, it restricts access to sequence diagrams using a special license.

4.3.2. Dynamic Analysis

The classes that make up the application and their relationships were detected by static analysis. The dynamic analysis is applied to detect how classes interact in resolving functionality provided. In this case information is retrieved using two techniques: execution traces, and detection of memory status.

To obtain and analyze execution traces of an application, we select the Eclipse Test and Performance Tools Platform (TPTP) [10]. It provides an open platform supplying powerful frameworks and services that allow software developers to build unique test and performance tools. TPTP allows executing instances of the application and registering the invocations. While the result is not a classic sequence diagram (for example, control statements are not detected) it is a good approximation to detect objects involved in each specific functionality and method invocation sequences. It also allows you to see how the components of the user interface components interact directly with the domain but not with the database components. To access the data provided by these components, BizServices class, that represent the server component, is used.

Other dynamic analysis technique that was used in the process is memory snapshots. This analysis seeks to recover what is the current value of each of the attributes of the objects created during the execution of the application. This information is important not only to successfully deploy the application on the target platform, but in the modeling stage, as described in the next section. To detect the state of the memory was used a commercial tool that can be freely used for a limited time called YourKit Java Profiler [25]. This tool allows running the application and capturing the information of the objects that were created in memory.

This information is very important not only to successfully implement the application of the target platform, but also useful for modeling stage. For example, Fig. 3 shows a simplified execution trace obtained when the screen that lets us manage the system contacts is initialized.

4.4. Model Transformation

This step integrates the artifacts recovered by the reverse engineering analysis techniques previously described, with the ideas behind MDD to achieve the platform independence representation involved so far.

MDA aims at the development of software systems based on the separation of business and application logic from underlying platform technologies, facilitating technology independency and interoperability between platforms. All artifacts involved in a development process are represented by means of metamodeling techniques, MOF metamodeling in particular. Our goal is to generate platform independent models from the results of the preceding analysis steps. The consistency of resulting models is evaluated by expressing the model translation in ATL.

This stage of the translation process was supported by the Eclipse Modeling Project (EMP) which provides tools for both defining metamodels and transformation rules, and executing the translation process. ATL mainly focuses on the model-to model transformations which can be specified by means of ATL modules.

The Android platform provides a version of the Java language that is different to the version provided by environments of standard execution (Java Runtime Environment). One of main differences of this version is the way of constructing graphic interfaces. It does not provide frameworks such as Swing or AWT but its own component libraries called widgets. Considering the above-mentioned, we present examples of translation centered on the
components of the user interface module, which require substantial changes. Fig. 1 shows the relation between models, source and target metamodels, and model transformations.

Following, we describe a simplified Java/JSwing metamodel that includes classes (and attributes) used for the construction of Client Management screen (Fig. 4). On the other hand, Fig. 5 shows a simplified Java/Android target metamodel and the concrete model of the application to implement screens of client management. The main difference between the source and target metamodels is that interface controls do not provide the same functionality for all cases. In some cases, due to technological constraints and characteristic of the target platform, it is necessary create equivalent functionality using different widgets. One such case may be the JTable class, which implements a data table, which has no equivalent functionality in Android and will be implemented by combining other controls.

In other cases, we can also see restrictions that are configured from attributes of a control, becoming associations between widgets. For example, to set a maximum size for the number of characters that can be entered in an edit control (for class JTextFields, attribute column); it is represented in Android by means of the association between the class editText with a filter of input of length (class LenghtFilter and the configuration of its attribute nMax). These considerations will be present at the moment of establishing translation rules in ATL. ATL mainly focuses on the model to model transformations which can be specified by means of ATL modules. An ATL module is composed of the following elements:

- a header section that defines the names of the transformation module and the variables of the source and target metamodels.
- an optional import section that enables to import some existing ATL libraries
- a set of helpers that can be used to define variables and functions.
- a set of rules that defines how source model elements are matched and navigated to create and initialize the elements of the target models.

Following, we present some of the ATL rules that allow the translation between the two metamodels:

```atl
module SwingToAndroid;

create OUT : JavaAndroid from IN : JavaSwing;

helper context JavaSwing!Component def: getVisibility():
JavaAndroid!Visibility =
if self.visible = true then
  #VISIBLE
else
  #INVISIBLE
endif.
```

Fig. 4. Java/JSwing Metamodel.

Fig. 5. Java/Android Metamodel.
helper context JavaSwing!Component def getWidth(s: JavaSwing!Dimension): Integer =
    if s.oclIsUndefined() then
        0
    else
        s.width
    endif;

helper context JavaSwing!Component def getHeight(s: JavaSwing!Dimension): Integer =
    if s.oclIsUndefined() then
        0
    else
        s.height
    endif;

rule ComponentToView {
    from jc: JavaSwing!Component
to
tv: JavaAndroid!View (visibility <- jc.getVisibility(),
    id <- jc.name,
    enabled <- jc.enabled,
    width <- jc.width,
    height <- jc.height,
    mLeft <- jc.x,
    mTop <- jc.y,
    mMinHeight <- jc.getHeight(jc.minimumSize),
    mMinWidth <- jc.getWidth(jc.minimumSize))
}

rule ContainerToViewGroup extends ComponentToView {
    from jc: JavaSwing!Container
to
tv: JavaAndroid!ViewGroup (mChildren <- jc.component,
    mChildrenCount <- jc.ncomponents )
}

rule JComboBoxToSpinner extends ComponentToViewGroup {
    from jc: JavaSwing!JComboBox
to
tv: JavaAndroid!Spinner
}

rule JComponentToViewGroup extends ContainerToViewGroup {
    from jc: JavaSwing!JComponent
to
tv: JavaAndroid!ViewGroup
}

rule JLabelToTextView extends JComponentToViewGroup {
    from jc: JavaSwing!JLabel
to
tv: JavaAndroid!TextView(mText <- jc.text )
}

rule JTextFieldWithColumnsToEditText extends JComponentToViewGroup {
    from jc: JavaSwing!JTextField(jc.columns > 0)
to
tv: JavaAndroid!EditText(enabled <- jc.editable,
    filters: JavaAndroid!LengthFilter (mMax <- jc.columns))
}

rule JTextFieldToEditText extends JComponentToViewGroup {
    from jc: JavaSwing!JTextField(jc.columns = 0)
to
tv: JavaAndroid!EditText (enabled <- jc.editable )
}

rule JButtonToButton extends JComponentToViewGroup {
    from jc: JavaSwing!JButton
to
tv: JavaAndroid!Button(mText <- jc.text )
}

rule JScrollPaneToScrollView extends JComponentToViewGroup {
    from jc: JavaSwing!JScrollPane
to
tv: JavaAndroid!ScrollView
}

rule JPanelToRelativeLayout extends JComponentToViewGroup {
    from jc: JavaSwing!JPanel
to
tv: JavaAndroid!RelativeLayout
}

rule AddressPanelToAddressLayout extends JPanelToRelativeLayout {
    from jc: JavaSwing!AddressPanel
to
tv: JavaAndroid!AddressLayout (isModified <- jc.modified,
    mNameLabel <- jc.nameLabel,
    mAddrLabel <- jc.addrLabel,
    mAddrExtLabel <- jc.addrExtLabel,
    mCityLabel <- jc.cityLabel,
    mPhoneLabel <- jc.phoneLabel,
    mCountryLabel <- jc.countryLabel,
    mLastNameField <- jc.lastNameField,
    mFirstNameField <- jc.firstNameField,
    mMField <- jc.miField,
    mAddrField <- jc.addrField,
    mAddrExtField <- jc.addrExtField,
    mCityField <- jc.cityField,
mCountryField <- jc.countryField,
mZipField <- jc.zipField,
mZip4Field <- jc.zip4Field,
phoneNumberField <- jc.phoneField,
stateSpinner <- jc.stateCombo
)

Because the main source metamodel components are related to each other by a hierarchy, the rules also have the same structure. Therefore, the first rule describes how to transform the parent metaclass of the source class `Component` into the parent metaclass of the target class `View`.

The transformation is performed for each attribute in an almost direct way, except for attributes that need to be invoked from the previously defined helpers. The rules specify that, after determining that it is necessary to link metamodel objects, it should first try to solve each of the objects before linking objects of different metamodels. In this specific case, it determines that the components found in `Component`, are firstly evaluated to see if there are rules that define which elements should be transformed and thereafter, they are assigned to `mChildren`. Because of this, we define rules to transform each of the possible elements found in `Component` to their equivalent in the Android model.

These rules are evaluated by using information recovered through dynamic analysis of memory state. Next, the transformation is executed on the ATL virtual machine and an equivalent Android metamodel is created.

4.5 Target Application: Android Platform

From the design recovered by reverse engineering techniques and the transformation process created using metamodeling concepts, an equivalent design on the Android platform is created. By using this design we can complete the migration of the application under study. The main difficulties in the new implementation are associated with the particular features of platforms, primarily the size of the screen available to build the user interface, and methods of use of the input devices available which differ significantly from those found in a classical computer. Fig. 6 and Fig. 7 show the source screen of client and the target screen on the mobile device respectively.

5. Conclusions

This research integrates traditional reverse engineering techniques and MDA to simplify the modernization of legacy desktop applications to mobile platforms. We exemplify our approach with a case study on modernization of a legacy CRM software system to mobile platforms.

![Fig. 6. Original screen of client management.](image1)

![Fig. 7. Resulting screen of client management.](image2)
tools. This is one of the most important complications when attempting migrate legacy system logic into a new application. Similarly, poor documentation tools and development in metamodeling and transformation also cause inconvenience.

Beyond these difficulties we believe that the case study illustrates the acceptable feasibility of the proposed reengineering process. Future activities in reverse engineering should push towards a tight integration of dynamic analysis and human feedback into automatic reverse engineering techniques. The idea is to learn from expert feedback to automatically produce results.

A current problem is the development of an application for various mobile platforms. One limitation of our proposal is that it currently can handle Android Platform only. Our idea is to use different layers of abstraction that can map a “write once” application into native executable programs that will run on multiple platforms. Future projects are complementary to the limitations previously mentioned. We foresee to integrate haXe with MDD approaches.

References

Information for Authors

Scope

The aim of the International Journal of Computer & Information Science is to present information on well defined theoretical results and empirical studies, to describe current and prospective practices and techniques, to provide critical analysis of the use of CASE tools and to present broad issues in the fields of Computer Science and Engineering, Software Engineering and Information Science and Engineering. A primary aim of the journal will be to make the research accessible and useful to practitioners through carefully written methodology section and discussions of research results in ways that relates to practitioners.

Submission Policy

1. The IJCIS does not accept manuscripts or works that have been published in a journal, but accepts substantially revised and expanded versions of papers that were previously presented at conferences. In this case, the author should inform the Editor-in-Chief of the previous publication when the paper is submitted.

2. Submissions, referencing and all the correspondence will be conducted by e-mail.

3. Manuscripts should be in English and are subject to peer review before selection.

Procedure for Submission for Review

1. Send your complete text of manuscript to the Editor-in-Chief (e-mail: lee@cps.cmich.edu) via e-mail as an attachment document either in Word, PDF or PostScript format.

2. It may be useful for non-English speakers to have a native English speaker read through the article before submission.

3. Your e-mail message (to the Editor-in-Chief) should include the title of your paper, author(s) information (department/section, university/company, phone/fax numbers, e-mail address), the corresponding author’s complete address (mailing address), and area(s) of the paper.

Style of Manuscript to be Submitted for Review

1. The text should be double-spaced, two columned, single-sided in Times 12-point type on 8.5-inch x 11-inch paper with margins of 1 inch on all four sides. It should not exceed 30 double-spaced pages. All pages must be numbered sequentially.

2. Provide a 200-250 word abstract, 5-7 keywords. Your abstract should briefly summarize the essential contents.

3. All figures/tables must be captioned, numbered and referenced in the text.

4. An acknowledgement section may be presented after the conclusion section.

5. References should appear at the end of the paper with items referred to by numbers in square brackets.

Journal Papers: author’s first initials, followed by last name, title in quotation marks, journal (in Italics), volume and issue number, inclusive pages, month and year. For e.g.:


Papers of Proceedings: author’s first initials, followed by last name, title in quotation marks, proceedings / symposium / workshop name (in Italics), location, month, and year, and inclusive page numbers. For e.g.:


Books: author’s first initials, followed by last name, title (in Italics), edition, location: publisher, year. For e.g.:


Accepted Manuscript Submission

All accepted papers are edited electronically. You are required to supply an electronic version of the accepted paper. The detailed guidelines for preparing your final manuscript will be sent upon article acceptance. It is also available on the ACIS website at http://acisinternational.org.

Page Charges

After a manuscript has been accepted for publication, the author will be invoiced a page charge of $400.00 USD for up to 10 pages if he/she is not a member of ACIS and thereafter an additional $30 per printed page to cover part of publication after the receipt of your final manuscript (camera ready).