Schema Integration Methodology and its Verification  
by use of Information Capacity

Irene KWAN                                      Joseph FONG  
Computer Science & Information Systems Dept.   Computer Science Department  
The Brunel University, London, England           City University of Hong Kong, Hong Kong  
E-mail: irene@oodb.cse.ttit.edu.tw                         E-mail : csjfong@cityu.edu.hk

Abstract - In the past decade a considerable amount of research has been done on schema integration and translation. Until recently, most of the work has largely neglected the proof of correctness in transforming schemas. Anomalies could arise due to errors in the transformation rules. Invalid transformations can produce incomplete or inconsistent pictures that may cause damage to organisations. In this paper, we present a Schema Integration Methodology, with proof of correctness, based on the use of Information Capacity. We present our methodology as a set of steps for schema translation tasks, verify their correctness according to operational goals and derive the information capacity of the original schema in its pre-transformed and post-transformed conditions. If the information capacity of the original schema is equivalent to or dominated by the transformed schema, then information is preserved after integration. Our correctness criteria is based on the assumption that these goal(s) could be practically pursued in an operational sense and are not solely mathematical proofs.

Key words : schema integration, information capacity equivalence, correctness criteria, schema transformation, information completeness

1 Introduction

There are many schema translation methods proposed as part of database reengineering methodologies [1]. Unnecessary anomalies can be avoided if these transformations could be shown to be correct. In this paper, we suggest a schema transformation methodology and apply the verification concept based on the correctness criteria proposed by Miller [2]. We identify the operation goals implicit in each step based on the pre and post conditions of each case, as well as the extent of goal(s) that each transformation rule could achieve and be justified as a valid translation. The second section describes Miller’s verification concept and criteria. The third section covers the three steps Schema Integration Methodology with rules, examples and verifications. Section four illustrates the methodology with a case study and section five summarises the main conclusions.

2 Verification of Schema Integration Rules by Use of Information Capacity

Information capacity equivalence and dominance is used as a basis for verifying the transformed schemas without information loss for schema integration. This is important because users expect no information loss after schema integration. We apply Miller’s theory on verifying the information capacity equivalence for our schema integration methodology.

A classification of generic integration and translation tasks based on their operation goals is defined by Rosenthal and Reiner [3]. There are different levels of operational goals (G1, G2, G3, G4) in using database systems, ranging from queries to update. Our method is to prove that the state of the database is the same for both source schemas and integrated schema. In our method, each translation rule and relative information capacity requirements of the pre and
post conditions of original and transformed schemas is first derived and then examined to find out whether the information capacity preserving mapping is achieved. The transformation rule is proved to be correct if all the goals are met. However, for most integration tasks, information capacity equivalence of the schemas is not required; rather it is sufficient to identify dominance of either the original or the transformed schemas [2]. In practice, the notions of dominance and equivalence are most useful if the associated mappings are required to capture meaningful semantic correspondence between schemas. In our approach, verification of the information capacity between schemas is done by applying the operational goals on the assumption that the goal(s) could be technically and practically achieved in an operational sense. Hence, our work includes informal If-Then rules to explain each integration process. This is followed by more formal verification using set notation to define the pre-integrated condition and post-integrated condition in terms of its information capacity requirements the extent to which each transformation achieves its goals.

In this section, we explain the use of information capacity and operational goals in evaluating our schema integration rules. It is valuable to analyse whether the original schemas and the transformed schema after the mapping function are in a dominance or equivalence domain. To achieve different levels of goals would require the mapping function to extend the information capacity. There are five classifications of information capacity which a mapping function can hold.

1. **Functional**
   Let A and B be sets. The mapping is Functional if \( f : A \rightarrow B \), \( \forall a \in A \), \( \exists b \in B \cdot f(a) = b \)

2. **Injective**
   If the inverse of the binary mapping relation is also Functional, then the function is injective.
   \( f^{-1} : B \rightarrow A \), \( \forall b \in B \), \( \exists a \in A \cdot f(b) = a \)

3. **Total**
   If the Functional binary mapping relation is defined on every element of A, then the function is Total and it is an information capacity preserving mapping between the instances of two schemas S1 and S2.
   (Note: S1 denote the original schema and S2 denotes the transformed schema.)
   \( f : I(S1) \rightarrow I(S2) \) where \( I(S_n) \) denotes the set of all (data) instances of schema \( S_n \) and \( S_1 \) dominates \( S_2 \) (i.e. \( S_2 \subseteq S_1 \)), that is \( \forall I(S1) \in S1, \exists I(S2) \in S2 \)

4. **Surjective**
   If the inverse of the Total function is injective, then it is an information capacity preserving mapping between the (data) instances of two schemas S1 and S2. This Total and Injective function is called a Surjective function.
   \( f^{-1} : I(S2) \rightarrow I(S1) \) where \( S_2 \) dominates \( S_1 \) (i.e. \( S_1 \leq S_2 \)), that is \( \forall I(S2) \in S2, \exists I(S1) \in S1 \)

5. **Bijection**
   If the mapping function meets all the above four properties, it is an information equivalence preserving mapping.
   \( f : I(S1) \rightarrow I(S2) \) where \( I(S_n) \) denotes the set of all (data) instances of schema \( S_n \)

Hence, S1 and S2 are equivalent via f.

That is \( \forall I(S1) \in S1, \exists I(S2) \in S2 \wedge \forall I(S2) \in S2, \exists I(S1) \in S1 \wedge S1 \equiv S2 \)
Schema integration provides a global view of multiple schemas. It involves using a bottom up approach to integrate existing databases into one by considering pairs of databases. In the content of our methodology, it can be simplified in the following algorithm:

Begin
For each existing database do
IF its conceptual schema does not exist
THEN reconstruct its conceptual schema in EER model by reverse engineering;
For each pair of existing databases’ EER models of schema A and schema B do
begin
Resolve the semantics conflicts between the schema A and schema B; /* step 1 */
Merge the entities between the schema A and schema B; /* step 2 */
Merge the relationships between the schema A and schema B; /* step 3 */
end
end

As shown in the above algorithm, schema integration consists of a set of tasks[4]. It requires the achievement of at least the third level of operational goals (G3) to ensure information completeness. We verify our schema integration rules by means of a two-steps correctness proof. First, we derive from each integration rule, the pre-integration condition and post-integration conditions in terms of information capacity requirements. Second, we identify the operational goal(s) implicit in each of these integration processes and the extent of goal(s) for each mapping function could achieve a valid transformation. The four levels of operational Goals (G1, G2, G3 and G4) for systems involving two schemas and their relative information capacity requirements are:

**G1** targets to make a query via S1 the data stored under S2, where S1 is a view of S2, that is, providing a logical view of S1 on the physical database of S2. Hence, a Total function is required at instance level for achieving this minimum operational goal. Since the query q on I(S1) is mapped to the unique query q on I(S2), the function does not have to be information preserving to achieve G1. The information capacities of S1 and S2 may be incommensurate.

**G2** is to achieve G1, that is, querying through S1 the entire database stored under S2 where S1 is a view of S2, plus also viewing through S1 the entire database stored under S2. At the minimum, a Total Injective function is required to achieve G2 since the function needs no loss of information where information preserving mapping is required to achieve G2. S1 must dominate S2.

**G3** is to achieve G2, plus updating through S1 the data stored under S2. Hence, to achieve G3, at the minimum, a Total Injective and Surjective function is required. An update u that changes instance of S1, i1 to a new instance i1’, that is u(i1) = i1’ which i1’ should determine a unique instance of S2. As a result, f must be Surjective. Since the function f must be information preserving mappings to achieve G3, S2 must dominate S1.

**G4** is to querying through S1 the data stored under S2 and also through S2 the data stored under S1. It is a bi-direction and information preserving mapping in both direction to achieve G4. S1 and S2 must be equivalent in both directions to allow updates be done through both S1 and S2. Hence, f is a Bijection function and to achieve G4, S1 and S2 must be equivalent.
3 Methodology for Schema Integration and its Verification in Schema Integration Steps

In our approach, a successful schema integration process should require information capacity of the original schemas to be equivalent or dominated by the transformed schemas. To achieve this, we must prove that each proposed integrated process can fulfill up to the third level of its operational goal (G3) to ensure information completeness. The following three major steps must be followed in its sequence. However, the sequence of sub-steps in each major step is immaterial.

Step 1. Resolve conflicts among conceptual schema in EER models.

Sub-step 1.1 Resolve conflicts on synonyms and homonyms

<table>
<thead>
<tr>
<th>Rule</th>
<th>IF A.x and B.x have different data types or sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>THEN x in A and B may be homonyms, let users clarify x in A and B</td>
</tr>
<tr>
<td></td>
<td>ELSE IF x ≠ y, and A.x and B.y have the same data type and size</td>
</tr>
<tr>
<td></td>
<td>THEN ((x, y) may be synonyms, let users clarify (x, y));</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Eg.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[Diagram of EER model before and after transformation with synonyms and homonyms]</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1 EER model with synonyms and homonyms

<table>
<thead>
<tr>
<th>Pre-cond.</th>
<th>∃ Entities (A, B) • ∀ x ∈ (Attribute (A) ∩ Attribute (B))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Homonyms ⇒ (A.x = B.x) ∧ (Role (A.x) ≠ Role (B.x))</td>
</tr>
<tr>
<td></td>
<td>Synonyms ⇒ ∃ (x, y), x ∈ attribute (A) ∧ y ∈ attribute (B)</td>
</tr>
<tr>
<td></td>
<td>(A.x ≠ B.y) ∧ (Role (x) = Role (y))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Post-cond.</th>
<th>Homonyms ⇒ let (A.x ≠ B.x) ∧ (Role (A.x) ≠ Role (B.x))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Synonyms ⇒ let (A.x = B.y) ∧ (B.y = A.x) ⇒ A.x = B.y</td>
</tr>
</tbody>
</table>

| Proof                                     | This step is subject to the users input during transformation process. Role, by definition, is the functional usage of an entity [5]. However, to define role, in the case of synonyms, either A.x or B.x dominates one another in its data type and size, which has its information capacity preserved. The only trigger here is the user identification of its semantics equivalence. Similarly, once user has identified that the attributes are of homonyms, the data types and its size can be re-defined into different data structure. This translation process is bi-directional and the information capacity is preserved. |
Sub-step 1.2 Resolve conflicts on data types

| Rule | IF \( x \in (\text{attribute}(A) \cap \text{entity}(B)) \) THEN entity A’ ← entity B such that cardinality (A, A’) ← n:1 ELSE IF \( x \in (\text{keys}(A) \cap \text{entity}(B)) \) THEN entity A’ ← entity B such that cardinality (A, A’) ← 1:1 ELSE IF \( (x \subset \text{keys}(A)) \cap \text{entity}(B)) \) THEN entity A’ ← entity B such that cardinality(A, A’) ← m:n |

Eg.

<table>
<thead>
<tr>
<th>Case 1</th>
<th>Schema A</th>
<th>Schema B</th>
<th>Transformed schema A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loan</td>
<td>Customer</td>
<td>Customer name ➔ type</td>
<td></td>
</tr>
<tr>
<td>Contract</td>
<td>n</td>
<td>A'</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 2</th>
<th>Schema A</th>
<th>Schema B</th>
<th>Transformed schema A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loan</td>
<td>Customer</td>
<td>Customer name ➔ type</td>
<td></td>
</tr>
<tr>
<td>Contract</td>
<td>1</td>
<td>A'</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 3</th>
<th>Schema A</th>
<th>Schema B</th>
<th>Transformed schema A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loan</td>
<td>Customer</td>
<td>Customer name ➔ type</td>
<td></td>
</tr>
<tr>
<td>Contract</td>
<td>m</td>
<td>A'</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2 EER model with data types conflicts in 3 cases

Pre-cond. | Case 1: \( \exists \) Entity (A) \& Entity(B) • A.x = Entity(X) where \( x \in (\text{Attribute}(A) \cap \text{Entity}(B)) \)
Case 2: \( \exists \) Entity (A) \& Entity(B) • A.x = Entity(X) where \( x \in (\text{Key}(A) \cap \text{Entity}(B)) \)
Case 3: \( \exists \) Entity (A) \& Entity(B) • A.x = Entity(X) where \( x \in (\text{Component-Key} (A) \cap \text{Entity}(B)) \)

Post-cond. | Case 1: let A’ = (Entity(B), R(A,A’)) • R(A,A’) ← n:1
Case 2: let A’ = (Entity(B), R(A,A’)) • R(A,A’) ← 1:1
Case 3: let A’ = (Entity(B), R(A,A’)) • R(A,A’) ← m:n

Proof case 1

Case 1 conflict occurs when an attribute appears as an entity in another schema. Case 2 conflict occurs where a key appears as an entity in another schema and Case 3 conflict occurs when a component key appears as an entity in another schema. To verify case 1, since the translation process has preserved the information capacity in both the original schema A and schema B into the transformed schema A = (A, R(A,A’), A’), the transformed schema A has proved to dominate original schemas. The transformation process is information preserved. This transformation mapping between schema A and schema B to resolve conflicts on data types has satisfied goals G1, G2, and G3 since schema B remains its original structure. Unless schema B is applied together with transformed schema A to recover its original schema, bi-directional at instance level is not feasible. The verification of case 2 and case 3 is similar for all cases which are transforming entity with attributes as entity in another schema. The only difference is the cardinality between the created entity A’ and the original entity.
Sub-step 1.3 Resolve conflicts on key

| Rule | IF \( x \in (\text{key}(A) \cap \text{candidate_keys}(B)) \)  
|      | THEN let users clarify \( x \) in A and B

Eg.

![EER models with key conflicts](image)

Pre-cond. \( \exists \) Entities(A,B) \( \bullet x \in (\text{key}(A) \cap \text{candidate key}(B)) \)

Post-Cond. Either Entity (B) \( \leq \) Entity (A) or vice versa.
IF Entity (A) dominates Entity (B)  
THEN key = x.A  
ELSE key = x.B

Proof

G1 The conflict exists where a key appears as a candidate key in another schema. The verification of this rule is subject to the users input. User will have to decide on whether Schema B dominates Schema A. If so, schema A will take the key of schema B as its own key or vice versa. Hence, this translation process is information capacity preserved and bi-directional.

Sub-step 1.4 Resolve conflicts on cardinality

| Rule | IF (entity(A_1) = entity(B_1)) \( \land \) (entity(A_2) = entity(B_2)) \( \land \) (cardinality(A_1, A_2) = 1:1) \( \land \) (cardinality(B_1, B_2) = 1:n)  
|      | THEN cardinality(A_1, A_2) \( \leftarrow \) 1:n;  
|      | ELSE IF (entity(A_1) = entity(B_1)) \( \land \) (entity(A_2) = entity(B_2)) \( \land \) (cardinality(A_1, A_2) = 1:1 or 1:n) \( \land \) (cardinality(B_1, B_2) = m:n)  
|      | THEN cardinality(A_1, A_2) \( \leftarrow \) m:n;

Eg.

![EER model with cardinality conflicts](image)

Pre-cond. \( \exists \) cardinality (A1,A2) and cardinality (B1,B2)  
(entity (A1)= entity(B1)) \( \land \) (entity(A2)= entity(B2))
\[ \Rightarrow \text{cardinality } (A_1, A_2) \neq \text{cardinality } (B_1, B_2) \]

**Post-cond.**
Cardinality \((A_1, A_2) = \text{cardinality } (B_1, B_2) \leftarrow \text{Higher Cardinality}

**Proof**
Conflict exists where identical entities are of different cardinality in two schemas. The verification of this step is subject to which schema has higher cardinality. Schema with higher cardinality naturally dominates the other schema with identical entities. Hence, higher cardinality will override the lower cardinality conflicts. This translation process is therefore information capacity equivalent and is bi-directional with feasible recovery of original schema from transformed schema.

**Sub-step 1.5 Resolve conflicts on weak entities** [5]

**Rule**
\[
\begin{align*}
&\text{If } ((\text{entity}(A_1) = \text{entity}(B_1)) \land (\text{entity}(A) = \text{entity}(B)) \land ((\text{key}(A_2) = \text{key}(B_2))=0) \\
&\land ((\text{key}(B_1)) \cap \text{key}(B_2)) \neq 0)
&\text{then } \text{Key}(A_2) \leftarrow (\text{Key}(A_1) + \text{Key}(A_2))
\end{align*}
\]

**Eg.**

![EER model with weak entity conflict](image)

**Pre-cond.**
Role(Entity \((A_2)) = \text{Role(Entity}(B_2))

\[ \exists \text{Entities } (A_1, A_2) \land \text{Entities } (B_1, B_2) \land R(A) \land R(B) \]
\[ ((\text{entity}(A_1)=\text{entity}(B_1)) \land ((\text{key}(A_1) \land \text{key}(A_2))=0) \land (\text{key}(B_1)) \cap \text{key}(B_2)) \neq 0) \]

**Post-cond.**
Key}(A_2) \leftarrow (\text{Key}(A_1)+\text{Key}(A_2))

**Proof**
Conflict occurs when a strong entity appears as a weak entity in another schema. The verification of this resolution step is subject to the inter-dependence between entities. Schema has weak entity which is similar to another strong entity in another schema but with an additional key component from its strong entity. The former dominates the latter. Hence, weak entity overrides the strong entity by transforming the strong entity to weak entity for consistency. This translation process is bi-directional and information capacity equivalent.

**Sub-step 1.6 Resolve conflicts on subtype entities**

**Rule**
\[
\begin{align*}
&\text{IF } ((\text{entity}(A_2) \subseteq \text{entity}(A_1)) \land (\text{entity}(B_1) \subseteq \text{entity}(B_2)) \land (\text{entity}(A_1) = \text{entity}(B_1)) \land \\
&\text{(entity}(A_2) = \text{entity}(B_2)))
&\text{THEN begin entity } X_1 \leftarrow \text{entity } A_1
\end{align*}
\]
entity X₂ ← entity A₂

\text{cardinality}(X₁, X₂) ← 1:1

end;

Eg.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig6}
\caption{EER model with subtype conflict}
\end{figure}

Pre-cond. \exists \text{Entities}(A₁, A₂) \text{and Entities}(B₁, B₂)\bullet

\text{(Entity (A₂) ⊂ Entity (A₁)) \land (Entity (B₁) ⊂ Entity (B₂))}

Post-cond.

\text{Entity X₁ ← Entity (A₁), Entity (B₁)}

\text{Entity X₂ ← Entity (A₂), Entity (B₂)}

\text{Cardinality(X₁,X₂) ← 1:1}

Proof

\text{Conflict exists where a subtype entity appears as a super type entity in another schema. The verification of this step is to identify the overlapping of two identical entities in bi-directional in two different schemas. A₁ isa A₂ in one schema and A₂ isa A₁ in another schema. This translation process is transformed into schema with 1:1 cardinality. This translation process is obvious to be bi-directional and information capacity equivalent with no information loss.}

In step 2 and step 3, the transformation processes are totally based on its pre-condition without users’ interference during the integration process.

\textbf{Step 2 Merge Entities}

\textbf{Sub-step 2.1 Merge Entities by Union}

<table>
<thead>
<tr>
<th>Role</th>
<th>IF ((domain(A) \cap domain(B)) \neq 0) THEN domain(X) ← (domain(A) \cup domain(B))</th>
</tr>
</thead>
</table>

Eg.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig7}
\caption{Merge EER models by union}
\end{figure}

Pre-cond. \text{(Domain (A) \cap Domain (B)) \neq 0}

Post-cond. \text{Domain (x) ← (Domain (A) \cup Domain (B))}
Proof

G1: This step is Functional since \( A \cup B \) (i.e. \( A \cup B \)) is without duplicating the overlapping. Hence, \( \forall (a \lor b) \in (A \cup B), \exists x \in X \cdot f(a \lor b) = x \). It is Injective since there is a one to one mapping between elements of domain \( A \cup B \) and elements of domain \( X \). Hence, \( \forall x \in X, \exists a \in A \land b \in B \cdot f(x) = (a, b) \). It is Total since \( A \cup B \) is dominated by \( X \). Hence, \( \forall (I(a) \lor I(b)) \in (A \cup B), \exists I(x) \in X \cdot f(I(a \lor b)) = I(x) \Rightarrow (A \cup B) \leq X \). It is Surjective since \( A \cup B \) is an union of \( A \) and \( B \) without duplicating the overlapping and there is a one to one mapping between every instance of domain \( A \cup B \) and every instance of domain \( X \) and vice versa. Hence, \( \forall I(x) \in X, \exists (I(a) \cap I(b)) \in (A \cup B) \Rightarrow S(A,B) \equiv S(X) \). This transformation mapping between schema \( A \) union schema \( B \) to schema \( X \) has satisfied all the required properties of \( f \).

Sub-step 2.2 Merge Entities by Generalisation

| Role | IF ((domain(A) \cap domain(B)) \neq 0) \land ((I(A) \cap I(B)) = 0) THEN begin entity \( X_1 \leftarrow \) entity \( A \) 
entity \( X_2 \leftarrow \) entity \( B \) 
domain(X) \leftarrow domain(A) \cap domain(B) 
(I(X_1) \cap I(X_2)) = 0 
end 
ELSE IF ((domain(A) \cap domain(B)) \neq 0) \land ((I(A) \cap I(B)) \neq 0) THEN begin entity \( X_1 \leftarrow \) entity \( A \) 
entity \( X_2 \leftarrow \) entity \( B \) 
domain(X) \leftarrow domain(A) \cap domain(B) 
(I(X_1) \cap I(X_2)) \neq 0 
end; |

Example

Case 1

Case 2

Figure 8 Merge EER models by generalisation
Sub-step 2.3 Merge Entities by Subtype Relationship

<table>
<thead>
<tr>
<th>Pre-cond.</th>
<th>Post-cond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1: ((\text{Domain}(A) \cup \text{Domain}(B) \neq 0) \land (I(A) \cap I(B)) = 0)</td>
<td>Case 1: Entity ((X_1)) ← Entity ((A))</td>
</tr>
<tr>
<td>Case 2: ((\text{Domain}(A) \cap \text{Domain}(B) \neq 0) \land (I(A) \cap I(B)) \neq 0)</td>
<td>Entity ((X_2)) ← Entity ((B))</td>
</tr>
<tr>
<td></td>
<td>Entity ((X)) ← (Entity ((X_1)) ∧ Entity ((X_2)))</td>
</tr>
<tr>
<td></td>
<td>Domain ((X)) ← Domain ((X_1)) \cap Domain ((X_2))</td>
</tr>
<tr>
<td></td>
<td>(\exists (I(X_1) \cap I(X_2)) = 0 \land (x \in I(X_1) \land x \notin I(X_2)) \Leftrightarrow (y \in I(X_2) \land y \notin I(X_1)))</td>
</tr>
<tr>
<td>Case 2: ((\text{Domain}(A) \cup \text{Domain}(B) \neq 0) \land (I(A) \cap I(B)) \neq 0)</td>
<td>Entity ((X_1)) ← Entity ((A))</td>
</tr>
<tr>
<td></td>
<td>Entity ((X_2)) ← Entity ((B))</td>
</tr>
<tr>
<td></td>
<td>Entity ((X)) ← (Entity ((X_1)) ∧ Entity ((X_2)))</td>
</tr>
<tr>
<td></td>
<td>Domain ((X)) ← Domain ((X_1)) \cap Domain ((X_2)) (&amp; (I(X_1) \cap I(X_2)) \neq 0)</td>
</tr>
</tbody>
</table>

Proof

**Case 1:** Disjoint Generalisation - Entities with the same attributes appear in two schemas, but instance of the first entity in one schema cannot appear as instance of the second entity in another schema. It is Functional since \(A \cup B\) is an union of \(A\) and \(B\) without duplication in overlapped instance. Hence, \(\forall a \lor b \in (A \cup B), \exists x \in X \Rightarrow f(a \lor b) = x\). It is Injective since there is a one to one relationship between elements of either domain \(A\) or \(B\) and elements of domain \(X\) and vice versa due to the creation of a predicate field in the Entity \(X\). Hence, \(\forall a \lor b \in (A \cup B), \exists x \in X \Rightarrow f(a \lor b) = x\). It is Total since \(X\) is an union of domain \(A\) and domain \(B\) with no duplicating instance in both \(A\) and \(B\). There is a one to one mapping between every unique instance of domain \(A\) or \(B\) and every unique instance of domain \(X\). Hence, \(\forall I(a) \lor I(b) \in (A \cup B), \exists I(x) \in X \Rightarrow f(I(a) \lor I(b)) = I(x)\) and \(X\) dominates \((A \cup B)\) \(\Rightarrow (A \cup B) \leq X\). It is Surjective since there is a bi-directional relationship between the original schemas and the transformed schema due to the existence of a predicate field in the transformed schema. This results to a one to one relationship between every instance of domain \(A\) or domain \(B\) and every instance of domain \(X\) and vice versa. It is able to recover the instance of \(x\) which is derived from either \(X_1\) or \(X_2\). Hence, \(\forall I(x_1) \lor I(x_2) \in X, \exists I(a) \lor I(b) \in (A \cup B) \Rightarrow f(I(x_1)) = I(a)\lor I(b)\). This transformation mapping has satisfied the four required properties of \(f\). Hence, \(X\) is equivalent to \((A \cup B)\) without information loss and information capacity is preserved in its translation process.

**Case 2:** Overlap Generalisation - Entities with the same attributes appear in two schemas, but instance of the first entity in one schema can appear as instance of second entity in another schema. It is Functional since \(A \cup B\) is an union of \(A\) and \(B\) with overlapping in instance. Hence \(\forall a \lor b \in (A \cup B), \exists x \in X \Rightarrow f(a \lor b) = x\). It is Injective since \(A \cup B\) is an union of Entity \(A\) and Entity \(B\) without duplication in instance and there is a one to one mapping between elements of domain \(A\) and \(B\) to domain \(X\) and vice versa. Hence, \(\forall x \in X, \exists (a \lor b) \in (A \cup B) \Rightarrow f^{-1} x = (a \lor b)\). It is Total since \(X\) is an union of domain \(A\) and domain \(B\) with duplicating instance in both \(A\) and \(B\). There is a one to one mapping between every unique instance of domain \(A\) and \(B\) and every unique instance of domain \(X\). Hence, \(\forall I(a) \lor I(b) \in (A \cup B), \exists I(x) \in X \Rightarrow f(I(a) \lor I(b)) = I(x)\). Thus, \(X\) dominates \((A \cup B) \Rightarrow (A \cup B) \leq X\). It is Surjective since there is a bi-directional relationship between the original and transformed schemas by introducing a predicate field in the translated schema. This results to a one to one relationship between every instance of domain \(A\) and \(B\) and every instance of domain \(X\). It is able to recover the instance of \(x\) which is derived from either domain \(A\) or \(B\). Hence, \(\forall I(x) \in X, \exists I(a) \lor I(b) \in (A \lor B) \Rightarrow f(I(a) \lor I(b)) = I(x)\). This transformation mapping between schema \(A\) and \(B\) and schema \(X\) has satisfied the four required properties of \(f\). Hence, \(X\) does not only dominate \((A \cup B)\), but \(X\) is also equivalent to \((A \cup B)\) after schema is transformed.

Sub-step 2.3 Merge Entities by Subtype Relationship
<table>
<thead>
<tr>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF domain(A) (\subset) domain(B) THEN begin entity X(_1) (\leftarrow) entity A entity X(_2) (\leftarrow) entity B entity X(_1) isa entity X(_2) end;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Eg.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="EER models" /></td>
</tr>
</tbody>
</table>

**Pre-cond.**

| Domain (A) \(\subset\) Domain (B) |

**Post-cond.**

| Entity (X1) \(\leftarrow\) Entity (A) Entity (X2) \(\leftarrow\) Entity (B) Entity (X1) \(\subset\) Entity (X2) \(\Rightarrow\) Entity (X1) isa Entity (X2) |

**Proof**

<table>
<thead>
<tr>
<th>G1</th>
</tr>
</thead>
</table>

This step is a Functional mapping since A is related to B with a relation of A as a subset of B. Hence, \(\forall a\forall b \in (A,B,R(A,B))\), \(\exists x \in X \bullet f (A,B, R(A,B)) = x\). It is Injective since domain A is a subset of domain B. There is a bi-directional mapping between elements of A and B to X1 and X2. Any element that does not exist in domain B will be in domain A only and any element that exists in domain B will be also in domain A. Hence, \((\forall b \in B, \exists a \in A) \land (\forall x \in X \bullet f (A,B, R(A,B)) \rightarrow I(x))\). Note: \(\neg\exists\) implies do not exist. It is Total since X is an union of A and B, there is a one to one relationship between every unique instance of domain A or B and every unique instance of domain X1 and X2. Hence, \(I(a)\lor I(b) \in (A,B,R(A,B))\), \(I(x) \bullet f : I(A,B,R(A,B)) \rightarrow I(x)\). Thus, A dominates B and X2 dominates X1 \(\Rightarrow (A \cup B) \leq X\). It is Surjective since it is a bi-directional relationship between the original and translated schemas. There is a one to one relationship between every instance of domain A and every instance of domain X1 and between every instance of domain B and every instance of domain X2. It is able to recover the instance of X which is derived from either A or B. The practical recovery search logic is that any element that does not exist in domain B will be in domain A only and any element that exists in domain B will be also in domain A. Hence, \((\forall a \in A, \exists b \in B) \land (\forall x \in X \bullet f : I(A,B,R(A,B)) \rightarrow I(x))\). This transformation has satisfied all the required properties of \(f\). |

<table>
<thead>
<tr>
<th>G2</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>G3</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>G4</th>
</tr>
</thead>
</table>

**Sub-step 2.4 Merge Entities by Aggregation**

<table>
<thead>
<tr>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF relationship B (\rightarrow\rightarrow) entity A /*MVD (\rightarrow\rightarrow) means multi-value dependency/ THEN begin aggregation X(_i) (\leftarrow) (entity B(_i), relationship B, entity B(_i)) entity X(_i) (\leftarrow) entity A cardinality (X(_i), X(_i)) (\leftarrow) 1:n end;</td>
</tr>
</tbody>
</table>
Eg.

![Diagram of EER models](image)

Figure 10 Merge EER models by aggregation

<table>
<thead>
<tr>
<th>Pre-cond.</th>
<th>Post-cond.</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality (B1,B2) ← 1:n</td>
<td>Entity (X2) ← Entity (A)</td>
<td>This step is Functional since X provides a view to A and B. Hence, ( \forall (a,b) \in (A,B), \exists x \in X \bullet f(A,B)=X ). It is Injective since X is an aggregation of B1, B2 and R(B). Entity A and Entity B and their relationships are preserved in the transformed schema X. There is a bi-directional one to one mapping between elements of A, (B1,B2 R(B)) and (X1, X2, R(X)) by introducing a common key field. For example: Loan# in entity Loan Security is in both the original schema A and B and transformed schema X. Hence, ( \forall (a,b) \in (A,B), \exists x \in X \land (\forall x \in X, \exists (a,b) \in (A,B) \Rightarrow \exists x \in X \bullet f^{-1} (X) \rightarrow (A,B) ). It is Total since X is an aggregation of Entity B1 and Entity B2 and their relationship R(B). There is a one to one mapping between every unique instance of domain (A,B) and every unique instance of domain X. Hence, ( \forall (b1) \in B1 \land I(b2) \subseteq B2 \land I(x1) \subseteq X ) ( I(b1 \land b2) \equiv I(x1) ). Entity X1 dominates entity (B1 and B2) ( \Rightarrow ) entity(X1 and B2) ( \leq X ) to ensure there is no information loss during transformation. It is Surjective since there is a bi-directional relationship between the two schemas enhanced by the existence of a common key attributes between every instance of domain (B1,B2) and every instance of domain X1. It is able to recover the instance of x which is derived from either B1 or B2. X1 dominates the (B1and B2) to ensure that information is preserved after schema is transformed and X is proved to be equivalent to (A,B).</td>
</tr>
</tbody>
</table>

| Rule | IF \( I(B) \subseteq I(A_1) \lor I(B) \subseteq I(A_2) \) THEN begin entity X_y ← entity B entity X_1 ← entity A_1 entity X_2 ← entity A_2 categorisation X_y ← (entity X_1 , entity X_2) \( (I(X_1) \subseteq I(X_c)) \lor (I(X_2) \subseteq I(X_c)) \) /* X_c is subtype to X_1 or X_2 */ | |
Eg.

Figure 11 Merge schemas into Categorisation

Pre-cond. ∃ Entity(A1) ∧ Entity(A2) • Entity(A1) ≠ Entity(A2)

Post-cond. Entity (X2) ← Entity (B)

Entity (Xc1) ← Entity (A1) ∧ Entity (Xc2) ← Entity (A2)

Entity (X1) ← Entity(Xc1, Xc2)

∃ (I(x2) ⊂ I(xc1)) ∨ (I(x2) ⊂ I(xc2))

Proof

This step is Functional since ∀ a ∈ (A1 ∧ A2), ∃ xL ∈ X • f (A1 ∧ A2) = X1. Hence, X provides a view to Schema A and Schema B. The function is Injective since X1 is a union of A1 and A2. Information capacity of Entity A1 and Entity A2 are preserved in the transformed schema X1. There is a bi-directional one to one relationship between Entities of (A1,A2,B) and (X1,X2) by introducing a common key field Loan # in entities Loan Contract, Mortgage Loan and Commercial Loan, in both the original and the transformed schema during the translation process. Hence, xL ∈ X1, ∃ aL ∈ A1 ∧ aL ∈ A2 ⇒ ∃ xL ∈ X1 • f⁻¹ (X) = (A1,A2, B). It is Total since X1 is a categorisation of Entity A1 and Entity A2. There is a one to one mapping between every unique instance of domain A1 or A2 and every instance of domain X1. Entity X1 dominates Entity (A1, A2) ⇒ Entity(A1, A2) ≤ Entity X1. Entity X2 dominates Entity B ⇒ Entity B ≤ Entity X2 to ensure that there is no information loss during transformation. It is Surjective since there is a bi-directional relationship due to the creation of a common key field defined in the original and transformed schemas at the instance level. This results to a one to one relationship between every instance of domain (A1,A2 ) and every instance of domain X1. It is able to recover the instance of x1 which can be derived from either A1 or A2. Hence, ∀ l(x1)∈X1, ∃ l(a1)∧ l(a2)∈(A1,A2)) ∧ (∀ l(x2)∈X2 ∃ l(b)∈B)•f⁻¹ : l(x)→ l(a1,a2,b) is well proved. Schema (X) is equivalent to Schemas (A1,A2,B) ⇒ X ≡ (A1,A2,B).

Sub-step 2.6 Merge Entities by Implied Binary Relationship

Rule

IF x ∈ (attribute(A) ∩ key(B))

THEN begin entity X1 ← entity A

entity X2 ← entity B

cardinality (X1, X2) ← n:1

end
ELSE IF ((attribute(A) ∩ key(B)) ≠ 0) ∧ ((attribute(B) ∩ key(A)) ≠ 0)

THEN begin entity X₁ ← entity A
entity X₂ ← entity B
cardinality (X₁, X₂) ← 1:1
end;

Eg.

Case 1

Pre-cond. ∃ Entity A, Entity B • (attribute (A) ∩ key(B)) ≠ 0

Post-cond. Entity X₁ ← Entity A
Entity X₂ ← Entity B
Cardinality (X₁,X₂) ← n:1

Proof

Case 1 condition is that an identical data item appears in different data types in two schemas and case 2 condition is that two identical data items appear in different data types in two schemas. The two cases under this step have similar proof. Here, case 1 is taken as an example to demonstrate our proof. The mapping function in case 1 is Functional since (∀ (a,b) ∈ (A ∧ B), ∃ x ∈ X) ∧ (∃ a ∈ A ∧ a = key(B) ⇒ R(A,B) • f (A,B)=X. Hence, X provides a view to A and B. The function is Injective since X is a function of (A,B, R(A,B)), and that ∃ x ∈ X, ∀ a ∈ A ∧ ∀ b ∈ B ⇒ ∃ x ∈ X • f⁻¹ (X)=(A,B). Hence, it is a bi-directional mapping between entities (A,B) and entity X. It is a Total function since there is a mapping between every unique instance of entity A and B and every instance of entity X. That is, ∀ ( I(a)∈A ∧ I(b)∈B ), ∃ I(x)∈X • f I(a,b)=I(x). Hence, entity X dominates entity (A, B)⇒ entity A ≤ entity X and entity B ≤ entity X to ensure that there is no information loss during transformation. It is Surjective since it is a bi-directional mapping between the two schemas at the instance level. There is a common field of entity key to enable relationship built at each pair of instance in entity (A,B) and instance in entity X. It is able to recover the instance of entity X which is derived from entity(A, B). Hence, ∀ I(a)∧ I(b) ∈ (A,B), ∃ I(x)∈ X • f⁻¹ I(x) = I(a,b) is proved and Schema (A,B) ≡ Schema X

Step 3 Merge relationships

Sub-step 3.1 Merge relationships by subtype relationship [6][7]

| Rule | IF (entity(A₁) = entity(B₁)) ∧ (entity(A₂) = entity(B₂)) ∧ (participation(A₁, A) = total) ∧ (participation(B₁, B) = partial) |
Case 1
THEN begin
  entity X₁ ← entity A₁
  entity X₂ ← entity A₂
  entity X₃ isa entity X₁
  relationship X ← entity(X₃, X₂)
  participation(X₁, X) ← total
end

Case 2
ELSE IF (entity (A₁)=entity(B₁)) ∧ (entity(A₂) = entity(B₂)) ∧((relation(A) ∩ relation(B)) ≠ 0)
THEN begin
  entity X₁ ← entity A₁
  entity X₂ ← entity A₂
  entity X₃ isa entity X₂
  relationship X₁ ← Relationship A
  relationship X₂ ← Relationship B
end

Eg. Case 1

Figure 13 Merge EER models by subtype relationship

Pre-cond. Case 1 : ∃ (R(A),R(B)) • (Entity(A₁)=Entity(B₁) ∧ Entity(A₂)=Entity(B₂) ∧
((Relationship(A) ∩ Relationship(B))≠0)
Post-cond. Case 1 : (Entity (X₁)∈ Entity(A₁)) ∧ (Entity(X₂)∈ Entity (A₂))
  • (Entity(X₃) ⊂ Entity (X₂)) ∧ (Entity (X₄)⊂Entity(X₂))

Proof
G1
G2
G3
Proof G4
Case 1: Two relationships A, B are in the same role with different level of participation. The verification of this step is to identify the participation of two identical schema A and B with different level of participation but with the same role. The schema with total participation will naturally dominate the schema with partial participation to ensure no information loss after transformation. As the higher level of participation has absorbed the lower level of participation in the transformed schema with a new entity and relationship created, no alteration of data semantics is necessary. It is a Functional mapping since (∀ (a,b)∈ (A∧ B),∃ x∈X) ∧ (∃a∈ A ∧ a =key(B) ⇒ R(A,B) • f (A,B) = X. Hence, X provides a view to A and B. It is Injective since X is a function of (A,B, R(A),R(B)) and (∀ x∈X, ∃ a∈A ∨ ∃ b∈B ∧ (∀ x∈X, ∃ a∈A ∨ ∃ b∈B) ∧ (∀ x∈X, ∃ a∈P(A) ∨ ∃ b∈B) • f⁻¹ (X) = (A,B) (Note : P~ partial). It is Total such that there is a mapping between every unique instance of entity A and B and every unique instance of entity X since ∀
(I(a) ∈ A ∧ I(b) ∈ B), ∃ I(x) ∈ X • f I(a,b) = I(x). Hence, entity X dominates entity (A, B) ⇒ entity A ≤ entity X and entity B ≤ entity X to ensure that information capacity of entity A and entity B are preserved in the transformed schema X. It is Surjective since it is a bi-directional mapping between the two schemas at the instance level. There is a common field of entity key to enable relationship built at each pair of instance in entity (A,B) and in entity X. It is able to recover the instance of entity X which is derived from entity(A, B). As a result, ∀ I(a) ∨ I(b) ∈ (A,B), ∃ I(x) ∈ X • f⁻¹ I(x) = I(a,b) is proved and Schema (A,B) ≡ Schema X.

Eg.
Case 2

<table>
<thead>
<tr>
<th>Pre-cond.</th>
<th>Case 2 : (Entity (A1)=Entity (B1) ∧ Entity(A2)=Entity(B2)) ∧ (Participation (A1,A)=Partial) ∧ (Participation (B1,B) = Total)</th>
</tr>
</thead>
</table>
| Post-cond.| Case 2 : let Entity X3 ⊂ Entity X1  
Entity X2 ← Entity A2, Entity X1 ← Entity A1  
Participation (X3,X2) ← Total |

| Proof | Case 2 : Two relationships have different semantics but with intersecting relationship. The verification of this step is to identify two relationships have different semantics but with intersecting relationship. The schema which has overlapping relationships of different kinds of semantics would naturally dominate these schemas by assigning an overlap generalisation relationship to its intersecting schemas. Hence, information about its original semantics and relationships should both be preserved. This implies a bi-directional transformation process at instance level without loss of information. It is a Functional mapping since ∀ (a,b) ∈ (A ∧ B), ∃ x ∈ X ∧ (∃ a ∈ A ∧ b ∈ B) • x ∈ X ⇒ f (A,B) = X, X provides a view to A and B. The function is Injective since X is a function of (A,B,R(A),R(B)) and ∀ x ∈ X, ∃ a ∈ A ∧ ∀ b ∈ B ⇒ ∃ x ∈ X • f⁻¹ (X) = (A,B). Hence, there is a bi-directional mapping between entities (A,B) and entity X. It is Total since there is a mapping between every unique instance of entity A and B to entity X and ∀ (I(a) ∈ A ∧ I(b) ∈ B), ∃ I(x) ∈ X • f I(a,b) = I(x). Hence, entity X dominates entity (A, B) ⇒ entity A ≤ entity X and entity B ≤ entity X to ensure that there is no information loss during transformation and information preserved. It is Surjective as the bi-directional mapping between the schemas A, B and X at the instance level is supported. There is a common field : Customer # as entity key in entity Customer to enable that relationship is built at each pair of instance in entity (A,B) and instance of entity X. It is feasible to recover the instance of entity X which is derived from entity(A, B). Hence, ∀ I(a) ∨ I(b) ∈ (A,B), ∃ I(x) ∈ X • f⁻¹ I(x) = I(a,b) is proved. Schema (A,B) = |

---

Figure 14 Merge EER models by overlap generalisation
Sub-step 3.2 Absorbing Lower degree Relationship into a Higher degree Relationship

**Rule**

IF ((relationship(A) ⊃ relationship(B) ∧ (degree(A) > degree(B)) ∧ (entity(A1)=entity(B1)) ∧ (entity(A2)=entity(B2)))

THEN begin relationship(X) ← relationship(A)

entity X1 ← entity A1
entity X2 ← entity A2
entity X3 ← entity A3

end;

**Eg.**

<table>
<thead>
<tr>
<th>Schema A</th>
<th>Schema B</th>
<th>Schema X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customers</td>
<td>Loan</td>
<td>Contracts</td>
</tr>
</tbody>
</table>

Figure 15 Merge EER models by absorbing relationships

**Pre-cond.**

∃ R(A), R(B) • (relationship(A) ⊃ relationship(B)) ∧ (Degree A > Degree B)

**Post-cond.**

R(X) ← R(A) with higher degree relationship

**Proof**

This step is to identify the inconsistent degree level of two identical entities in different schema A and B. The schema with higher degree naturally dominates the schema with lower degree to ensure that there is no information loss after transformation. This translation process is to absorb the schema with lower degree relationship by the schema with higher degree relationship. This translation process is bi-directional and information capacity is preserved without information loss for the higher degree of relationship has absorbed the lower degree of relationship in the transformed schema. It is Functional mapping since (∀ (a,b)∈ (A∧ B), ∃ x∈ X) ∧ (∃a∈A ∧ b∈B) • ∃ x∈ X) ⇒ f (A,B) = X. X provides a view to A and B. The function is Injective , ∀ x∈X, ∃ a∈A ∧ ∀ b∈B ⇒ ∃ x∈X • f⁻¹ (X) = (A,B). It is a bi-directional mapping. It is Total for X is a function of (A,B, R(A,B)). There is a mapping between every unique instance of entity A and B and every unique instance of entity X. Hence, (∀ I(a)∈A ∧ I(b)∈B), ∃ I(x)∈X • f I(a,b) = I(x). Hence, entity X dominates entity (A, B) ⇒ entity A ≤ entity X and entity B ≤ entity X. Information capacity of schema A and B are preserved in the transformed schema X. It is Surjective, since it is a bi-directional mapping between the schemas A, B and X at the instance level. It is feasible to recover the instance of entity X which is derived from either entity A or entity B by referring to the participation of relationship between (X1,X) and (X2,X). The mapping rule has satisfied all the required properties of f.

4 Case study
A bank has existing databases with different schemas: one for a Mortgage Loan Customer, one for an Auto Loan Customer, one for Loan Contract and one for an Index Interest Rate. They are used by various applications in the bank. However, there is a need to integrate them together for an international banking loan system. The following is the four source schemas shown in Figure 16. In applying the algorithm of our methodology, the relevant steps are used in this case study as follows:

![EER models of the Loan System](image)

**Figure 16 EER models of the Loan System**

In the first iteration, in step 1.1, there are two synonyms: Loan_status and Balance_amt such that the Loan_status can be derived from the Balance_amt. As a result, we can replace Loan_status by Balance_amt with a stored procedure to derive Loan_status value from Balance_amt. In step 2.2, the intermediate integrated schema will be merged with the index rate schema. There is an overlapping generalisation between the two schemas such that a loan must be on fixed or indexed interest rate. Thus, by joining the integrated schema and the index rate schema with overlap generalisation, the two schemas can be integrated.

In the second iteration, in step 2.6, there is an implied relationship between the Loan Contract schema and (Mortgage loan) Customer segment such that ID# used as attribute in loan schema but as an entity key in customer schema. Thus, we can derive cardinality from the implied relationship between these entities, and integrate the two schemas into one EER model.

In the third iteration, in step 2.6, there is an implied relationship between the Loan Contract schema and (Auto loan) Customer segment and integrate the two schemas into one EER model. In step 3.1, the relationships between the loan contract and the two customer entities can be merged into an overlap generation as shown in Figure 17.
Figure 17 Integrated loan system schema

5 Conclusion

We have presented a three step Schema Integration Methodology with proof of its schema integration rules in terms of information dominance and equivalence in the transformation processes. We have applied the correctness criteria which depends on the use of information capacity concept and the extent of operational goals that each translation tasks can achieve. The processes can achieve up to the third level of goal or beyond, and are claimed to be a valid transformation with information capacity preserved. All the translation tasks in our proposed methodology cover all the basic conceptual database design aspects[8], and have technically satisfied up to at least the third level of these goals in an operational sense.

In conclusion, we have justified the correctness of our proposed schemas integration rules by (1) comparing information capacity between original schema and translated schema to ensure that there is no information loss in our transformation processes and (2) most of these steps are capable of being reverse to recover the original schema via the translated schema. (3) In addition to the proposed schema integration rules, we have also demonstrated that the correctness criteria by use of information capacity, as proposed by Miller, is a theoretically sound verification method for practical schema integration processes.

A research prototype was built on IBM486 by C++ and Paradox Engine to automate our schema integration rules [9]. The validity of these schema integration steps and rules are reinforced by this working prototype. Our next step is to investigate the possibility of integrating the new data design based upon the user's new data requirements on top of on existing integrated database[10].

Constraint rules in stored procedure
If balance_amt > 0
THEN loan_status = 'outstanding'
ELSE IF balance_amt = 0
THEN loan_status = 'paid_off'
Acknowledgement

The authors would like to thank the referees for their valuable comments and suggestions.

Reference


