A DATABASE WITH AN EXPLICIT SEMANTIC REPRESENTATION

Norio Katayama[1], Atsuhiro Takasu[2], and Jun Adachi[2]

[1] Department of Electrical Engineering, University of Tokyo, Hongo 7-3-1, Bunkyo-Ku, Tokyo 113, Japan.

[2] R&D Department, National Center for Science Information Systems, Otsuka 3-29-1, Bunkyo-Ku, Tokyo 112, Japan.

E-mail: {katayama, takasu, ja}@nacsis.ac.jp

ABSTRACT

In this paper, we present a database with an explicit semantic representation. A semantic representation is a mechanism to describe the semantics of data. A new type of database, called the S-object database, is proposed for this purpose. In conventional databases, the semantics of data is determined by the data's position in data structures. The S-object database, however, explicitly manages a semantic representation having a network structure. This approach improves the flexibility of a semantic representation; thereby making the S-object database especially suitable for storing data having complex structures and various semantics, which arise in novel database applications such as full-text databases. The major advantages of the S-object database are: (1) a semantic representation and the physical data are managed separately, (2) physical data are stored based on abstract data types, (3) a semantic representation has a network structure similar to hypertexts and semantic networks, and (4) queries and integrity constraints are described with predicate logic as in deductive databases. The S-object database accomplishes (1) – (4) by integrating the concepts of abstract data types, hypertexts, and deductive databases.

1 INTRODUCTION

This paper presents an "S-object (semantic object) database" which can explicitly represent semantic aspects of data; thereby providing the management capability for handling data with complex structures and various semantics. The S-object database integrates the concepts of abstract data types, hypertexts, and deductive databases; hence making it especially suitable for a full-text database and other novel database applications.

Section 1 discusses the explicit management of a semantic representation of data and compares our S-object database with related works. The applied "S-object model" is presented in Section 2, and the database's system structure is explained in Section 3. Section 4 describes the associated notation and management of constraints which are defined to preserve the consistency of a semantic representation, while Section 5 outlines query processing. Future work and conclusions are respectively contained in Section 6 and 7.

1.1 Semantic Representation of Data

As shown in Figure 1, when data is stored in a database, its meaning must also be simultaneously registered; otherwise the data's meaning will not be known when retrieved. For example, let us consider the small relational database depicted in Figure 2, where two data exist: "12345" and "J. Smith". In this case, the meanings of these data are determined by where they are contained in the table, i.e., "12345" is the student's identification (ID) number and "J. Smith" the student's name. Such data can be represented in the form of the semantic network [DK79, Leh92] as shown in Figure 3.

Therefore, a database contains mechanisms to either explicitly or implicitly represent the meanings of data. We term this representation as a *semantic representation of data*. In conventional databases, e.g., relational or object-oriented ones, data structures are specified in a fixed form of relations and objects, where the meaning of data is determined by their position within a data structure; hence these structures play the role of a semantic representation.

This approach has distinct advantages for achieving high performance using fixed data structures. On the other hand, limitations arise when handling data with complex structures or various semantics.

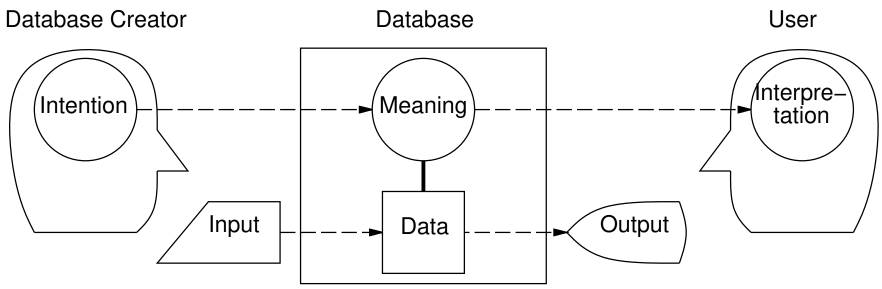


FIGURE 1: Data Registration and Retrieval Process

Student

StudentID	Name
12345	J. Smith

FIGURE 2: Student Database

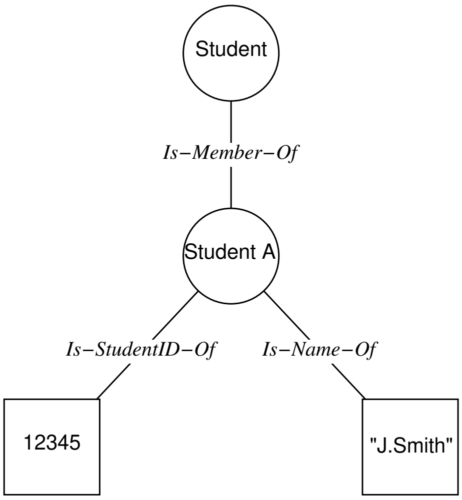


FIGURE 3: Network Representation

1.2 Explicit Management of Semantic Representations

Today's database applications are widely varied. One new application involves the use of full-text databases which store a document's primary information such as text and images, as well as secondary information such as title, authors, and abstract. In this application, stored data have the following characteristics:

– Various document structures

  Documents consist of various components (chapters, sections, abstract, etc.) having non-uniform structures (Figure 4).

– Various semantic relationships

  An example of semantic relationships among documents, magazines, and research fields is depicted in Figure 5.

It is difficult to fix data structures to this type of data, and therefore it is impractical to apply conventional methods such as relational and object-oriented databases.

Based on these considerations, we designed the S-object database which manages semantic representation explicitly. This database is comprised of semantic and data repositories (Figure 6), i.e., the semantic repository contains a network-structured semantic representation, whereas the data repository contains physical data.

Advantages and disadvantages of explicit semantic representation management in comparison to conventional approaches are summarized as follows:

Advantages

– Increased flexibility for handling a semantic representation.

– Separate management of physical data and a semantic representation.

– Queries can be described based on a semantic representation instead of on data structures; thereby making query descriptions easier and their semantics clearer.

– Integrity constraints can be described based on a semantic representation; thus making the descriptions and management of integrity constraints easier.

Disadvantages

– Due to the flexibility of a semantic representation being high, inconsistent or invalid representations may be stored; therefore management of the consistency of a semantic representation is necessary.

– Regarding effects on the resultant system efficiency, employing a flexible semantic representation creates management difficulties because data structures are not fixed.

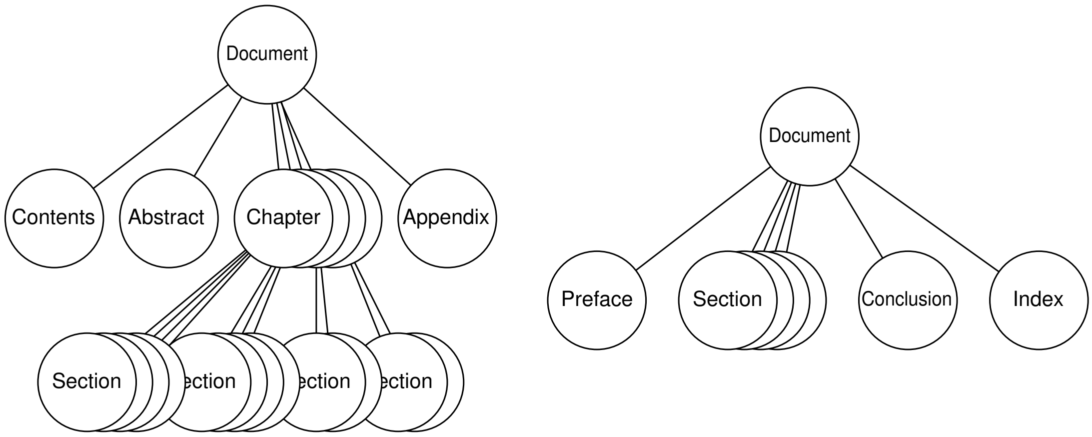


FIGURE 4: Document Structures

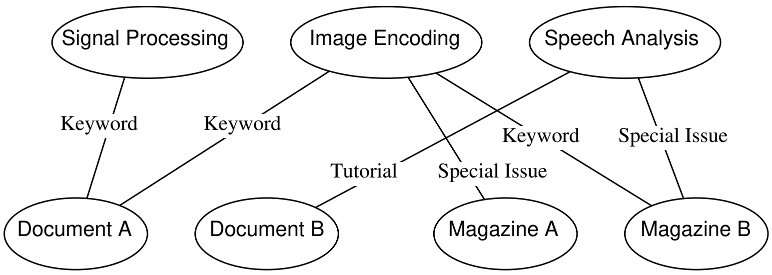


FIGURE 5: Semantic Relationships

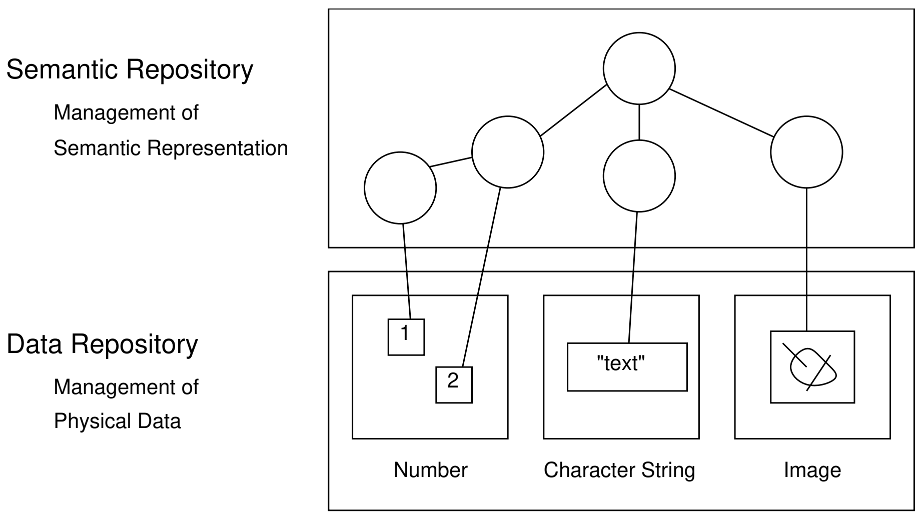


FIGURE 6: Database with Explicit Semantic Representation

Consequently, the S-object database is ideally appropriate for data having complex structures and/or various semantics. Conversely, relational databases and object-oriented databases are more suitable when the stored data has a uniform structure.

1.3 Design Policy of the S-Object Database

Design policy is outlined as follows:

- Information to be stored is classified into physical data or a semantic representation.
- Physical data are managed based on abstract data types, with each physical datum being encapsulated. This is accomplished by defining the type of numerics, character string, text, image, etc.
- The semantic representation is managed using the S-object model (Section 2).
- The management of the consistency of the semantic representation is realized by applying a predicate logic-based method used in deductive databases.

1.4 Related Works

Object-Oriented Database

The difference between an object-oriented database and the S-object database is the separation of information into physical data and a semantic representation.

In object-oriented databases, the following class definitions of objects are declared [ABD+89, Kim90]:

(1) Attributes to play the role of internal data.

(2) Composite attributes to refer to other objects.

(3) Methods to define operations for manipulating objects.

(1) and (3) define characteristics inherent to the objects, while (2) defines relationships among objects. Therefore, two types of declarations are mixed into one class definition, i.e., one is a declaration of the properties which are inherent to each object, and the other a declaration relevant to the relationships among the objects.

In comparison, in the S-object database, the properties inherent to physical data are encapsulated using abstract data types; and other properties, containing relevant semantic relationships among the data, are separated from the "inherent properties" and managed as a network-structured semantic representation. In addition, data consistency is managed using predicate logic, which is not employed in an object-oriented database.

Deductive Database

As in deductive databases, the consistency of a semantic representation is managed in the S-object database with predicate logic [Ari90, GMN84]. Although this is a common property, a deductive database primarily manages the derivation of data, whereas the S-object database instead uses predicate logic as the framework for describing integrity constraints of a semantic representation.

A deductive object-oriented database [YY92] has recently been studied, being a deductive database that introduces an object-oriented paradigm into predicate logic. However, the separation of information is not present.

Hypertext Database

Hypertext databases [TNH+91] have also been investigated, with the main objective being the integration of hypertext and database techniques. The S-object database contains a similar property in that both have a network structure. On the other hand, in the hypertext database the physical data and semantic representation are not managed separately and predicate logic is not applied to manage semantic consistency.

Semantic Data Model

The semantic data model [PM88] is a general term applied to data models developed for representing semantics of data, e.g., the entity-relationship model and the functional data model are generally considered as such models. Because the S-object model is designed to represent semantics of data, it could be considered as a semantic data model, yet it differs because: (1) physical data and a semantic representation are managed separately, (2) in our semantic representation, an entity, abstract concept, and physical datum are treated uniformly, and (3) consistency is managed with predicate logic.

2 S-OBJECT MODEL

2.1 Components

The S-object model is a data model representing semantic aspects of data, having a network structure constructed similarly to the conventional semantic network (Figure 7). This model is comprised of S-objects, P-data, S-relations, S-links, and S-constraints.

An *S-object* represents a semantic unit in the database, being a node within the network structure, i.e., the smallest unit of a semantic representation. An *S-relation* is a semantic relationship among S-objects, producing *S-links* among S-objects which are links in the network. An *S-constraint* is a restriction against a particular semantic representation, having the role of preserving its consistency.

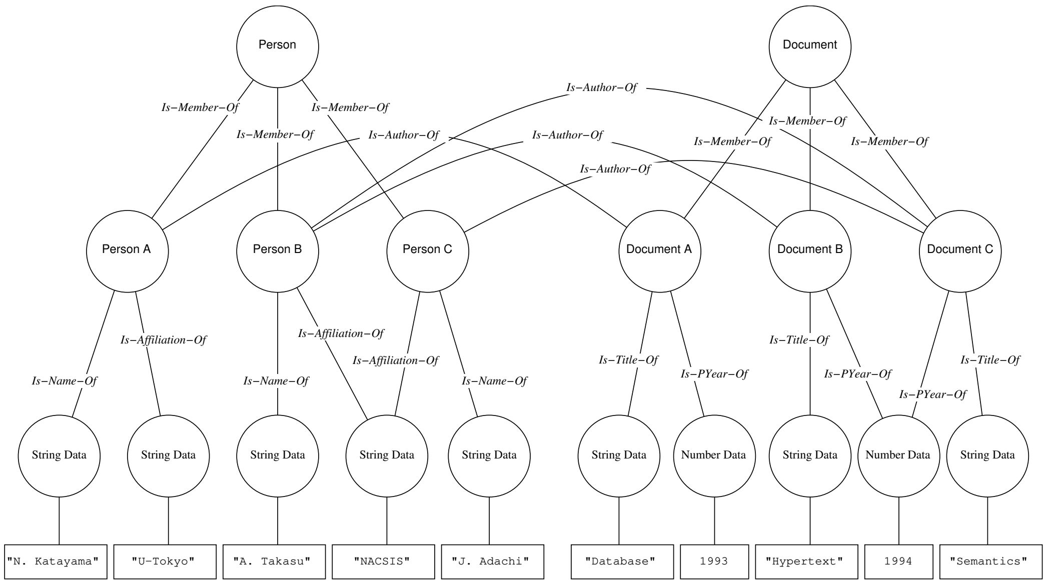


FIGURE 7: Semantic Representation with the S-Object Model

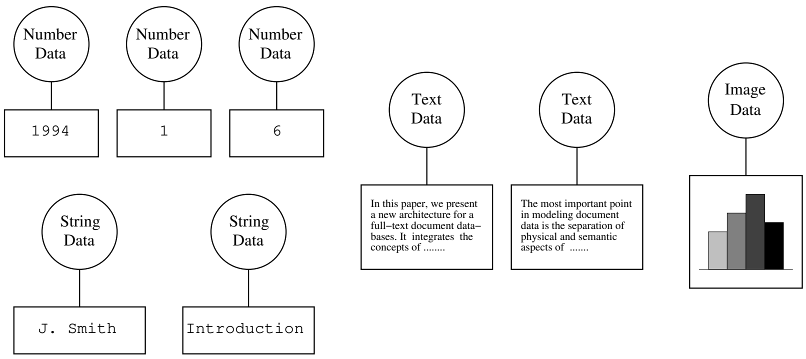


FIGURE 8: Pairs of P-data and S-object

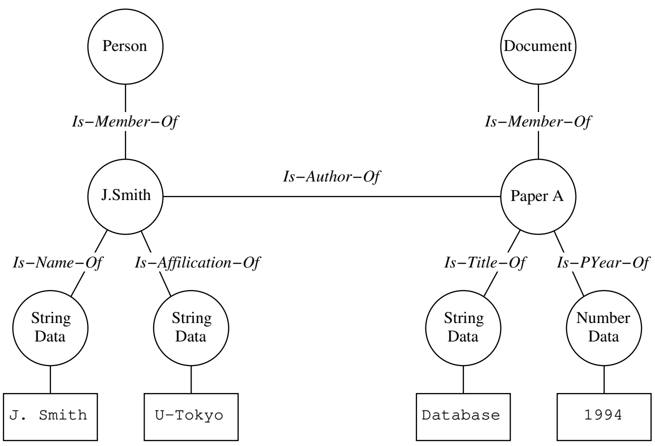


FIGURE 9: S-link Between an Author and his Document

2.2 S-Object

The role of an *S-object* is to uniquely represent a database semantic unit. These units can be classified as follows:

- Real World Entities
  e.g., individual documents, persons, and pages
- Conceptual Entities
  e.g., sets (documents, persons, etc.) and abstract concepts (signal processing, image encoding, etc.)
- Physical Data
  e.g., numerical, character string, text, and image data

When the above semantic units are stored in a database, S-objects are inserted to represent them within a semantic representation network. Since an S-object is just a node, it should not be confused with the objects in an object-oriented database.

The uniqueness of an S-object should be especially noted, namely, an S-object which represents a semantic unit must be unique, and conversely, a semantic unit which is represented with an S-object must be unique. Without this concept being true, a semantic representation built using the S-object model would not be unique.

2.3 P-Data

A *P-data* is a physical data. It is encapsulated based on the concept of abstract data types. An abstract data type is a subset of the object-oriented data model. It is a data type which is defined with two types of definitions: one is the definition of a data structure which provides an internal representation of the type, and the other is the definition of operations on the data structures. Each P-data has an S-object which takes the semantic role for the P-data. The mapping between a P-data and its corresponding S-object is one-to-one (Figure 8). With this mechanism, P-data can be treated as an encapsulated component of the semantic representation.

2.4 S-Relations

The role of an *S-relation* is to uniquely represent the semantic relationship among S-objects. For example, consider that the author of "Paper A" is "J. Smith " (Figure 9). Now, let ⟨`NAME`⟩ stand for an S-object whose name is `NAME`, then, between ⟨`Paper A`⟩ and ⟨`J. Smith`⟩ exists a relationship in which "⟨`J. Smith`⟩ is the author of ⟨`Paper A`⟩". This relationship, i.e., "X is the author of Y", is described with a predicate as

```
Is-Author-Of [ Author = X, Document = Y ]
```

where `Is-Author-Of` is a predicate name identical to the name of the S-relation. Inside the brackets is a list of terms. Each term has a label, i.e., `Author` and `Document`. The number of terms for an individual S-relation is assumed to be constant, and must be greater than zero.

An *S-link* among S-objects is a link in a network structure. It can be practically represented by setting values to the terms of an S-relation, e.g., an S-link representing "⟨`J. Smith`⟩ is the author of ⟨`Paper A`⟩" is described as follows:

```
Is-Author-Of [ Author = ⟨J. Smith⟩,
               Document = ⟨Paper A⟩ ]
```

2.5 S-Constraints

A semantic representation can be constructed in a network structure with S-objects, S-relations, and S-links, yet when updated, it may become semantically inconsistent. Therefore, to preserve the consistency of a semantic representation, *S-constraints* need to be defined. Because S-relations are represented with predicates in the S-object model, S-constraints can be described with predicate logic (Section 4).

3 S-OBJECT DATABASE SYSTEM

The structure of the S-object database system is depicted in Figure 10. The semantic repository manages S-objects, S-relations, and S-constraints; whereas the data repository stores physical data and manages functional S-relations.

3.1 S-Object Manager

This module manages the creation and deletion of S-objects. Since S-objects are just nodes of a semantic representation, their management is carried out by managing their identifiers (*S-object IDs*).

3.2 S-Relation Manager

This module manages the creation and deletion of S-relations and S-links. Three types of S-relations are formed based on how they produce S-links:

- Denotative S-relations
  S-links are stored one by one.
- Connotative S-relations
  The existence of an S-link is determined by evaluating a calculation procedure that corresponds to the semantics of the S-relation. For example, the following `Is-Sum-Of` S-relation representing $z = x+y$ belongs to a connotative S-relation:

  ```
  Is-Sum-Of [ Sum = Z, Value1 = X, Value2 = Y ]
  ```

  In addition, a deductively derived S-relation is considered as a connotative S-relation. For example,

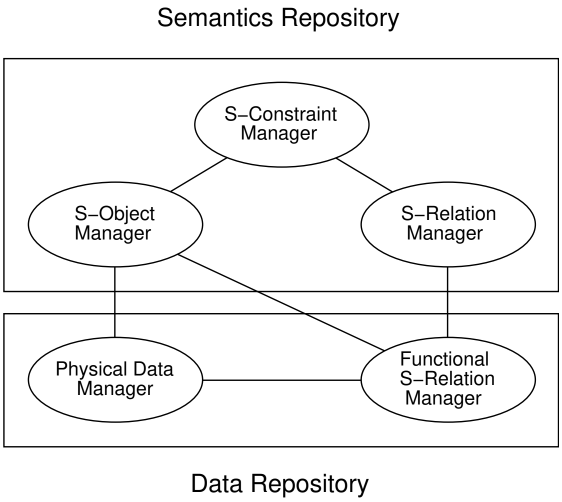


FIGURE 10: System Structure of the S-Object Database

the `Is-GrandFather-Of` S-relation is a connotative S-relation since it can be derived from the `Is-Father-Of` S-relation as follows:

```
Is-GrandFather-Of [ GrandFather = X,
                    GrandChild = Z ]
  ← Is-Father-Of [ Father = X, Child = Y ] ∧
    Is-Father-Of [ Father = Y, Child = Z ].
Is-GrandFather-Of [ GrandFather = X,
                    GrandChild = Z ]
  ← Is-Father-Of [ Father = X, Child = Y ] ∧
    Is-Mother-Of [ Mother = Y, Child = Z ].
```

– Composite S-relations

  Both denotative and connotative S-links are produced by a composite S-relation. For example, the `Is-Member-Of` S-relation stores denotative and connotative S-links. The connotative ones are recursively derived as follows:

```
Is-Member-Of [ Set = U, Member = X ]
  ← Is-Member-Of [ Set = L, Member = X ] ∧
    Is-Kind-Of [ SuperSet = U, SubSet = L ].
```

S-relations are additionally classified into two types based on the number of S-links produced.

– Finite S-relations

  This type stores a finite number of S-links, i.e., all S-links can be obtained from the S-relations. Although all denotative S-relations are finite ones, connotative and composite S-relations are not necessarily finite S-relations.

– Infinite S-relations

  This type produces an infinite number of S-links, i.e., S-links cannot be determined unless values are inserted into the specific terms. For example, the `Is-Sum-Of` S-relation is an infinite S-relation, since an S-link cannot be determined unless values are inserted into any two of three terms.

3.3 S-Constraint Management

This module manages definition, deletion, and evaluation of S-constraints, which are explained in Section 4.

3.4 Physical Data Storage

Abstract data types are employed in the S-object database to manage physical data. Data types are defined according to their physical properties: number, character string, text, image, etc. For each type, two types of declarations are described: one is the declaration of a data structure to be used as an internal representation, and the other is the declaration of operations on the data structure.

3.5 Functional S-Relation Management

Relationships inherent to physical data need to be provided for database users, e.g., the `Is-Substring-Of` relationship:

```
Is-Substring-Of [ Part = "BC", Whole = "ABCD" ]
```

Because these types of relationships can be considered as a semantic relationship among S-objects, they are treated in the S-object database as components of a semantic representation, being termed as *functional S-relations*. And, since such relationships are usually numerically resolved, they are in general connotative S-relations.

4 NOTATION AND MANAGEMENT OF S-CONSTRAINTS

4.1 Classification of S-Constraints

The S-constraints within the S-object model are classified as follows (Figure 11):

Contradiction: The state of S-relations (Γ) causes a contradiction.

$$\leftarrow \Gamma$$

Derivation: The S-links defined in an S-relation (R) are derived from the state of S-relations (Γ).

$$R \leftarrow \Gamma$$

Equivalence: The state of S-relations (Γ_1) is equivalent to another state (Γ_2).

$$\Gamma_1 \leftrightarrow \Gamma_2$$

The state of S-relations means what S-links are defined in them, being described based on predicate logic, i.e., expressed with predicates, variables, constants, logical negations ($\neg$), and logical conjunctives ($\wedge$). Predicates are interpreted as true when the corresponding S-link exists, otherwise interpreted as false.

Contradiction: The S-object in the `Author` term of `Is-Author-Of` must be a person.

$$\begin{array}{l}\leftarrow \texttt{Is-Author-Of [Author = X, Document = Y]} \wedge \\ \quad \neg\ \texttt{Is-Member-Of [Set = }\langle\texttt{Person}\rangle\texttt{, Member = X].}\end{array}$$

Derivation: `Is-GrandFather-Of` is derived from `Is-Father-Of`.

$$\begin{array}{l}\texttt{Is-GrandFather-Of [GrandFather = X, GrandChild = Z]} \\ \qquad \leftarrow \texttt{Is-Father-Of [Father = X, Child = Y]} \wedge \\ \qquad\quad \texttt{Is-Father-Of [Father = Y, Child = Z].}\end{array}$$

Equivalence: `Is-Nth-Chapter-Of` is equivalent to `Is-Chapter-Of` and `Is-ChapterNo-Of`.

$$\begin{array}{l}\texttt{Is-Nth-Chapter-Of [Document = D, Chapter = C, ChapterNo = N]} \\ \qquad \leftrightarrow \texttt{Is-Chapter-Of [Document = D, Chapter = C]} \wedge \\ \qquad\quad \texttt{Is-ChapterNo-Of [Chapter = C, ChapterNo = N].}\end{array}$$

FIGURE 11: Constraint Examples

4.2 Handling of S-Constraints

To preserve consistency of the S-object database, one of the following processes are executed for each S-constraint:

Contradiction: Before updates are committed, verification is carried out to ensure that no contradiction will occur.

Derivation: When information is retrieved from an S-relation, associated S-links are derived by evaluating corresponding constraints.

Equivalence: When an update occurs for Γ_1 or Γ_2, it is respectively reflected back to the other.

The contradiction and equivalence processes can be simply processed, whereas for the derivation process, we apply the stratified program [Llo87, Ull88, Min88] used in deductive databases.

In the stratified program, $\leftarrow$ is interpreted as the derivation of a new predicate and not as the logical implication. Moreover, a condition is imposed on predicates that there are stratified relationships among predicates in terms of derivation. When this condition is satisfied, the inference becomes monotonic even if the body (the right side of $\leftarrow$) includes negations ($\neg$) and it is known that the fixed point exists.

4.3 Evaluating the State of S-Relations

Since the state of S-relations is described with predicates, variables, constants, and logical operators ($\neg$, $\wedge$), then in principle, only the S-links defined in the S-relations need to be checked to evaluate the description of a state. However, the state's description may contain infinite S-relations which cannot be directly evaluated. Therefore, we employed a method to evaluate the description of the state.

Representation of the Evaluation Modes of an S-Relation

Let us consider the following infinite S-relation which represents a summation relationship.

`Is-Sum-Of [ Sum = Z, Value1 = X, Value2 = Y ]`

This S-relation can be evaluated by two modes: (1) Inserting values into the variables X, Y, Z and checking if they satisfy the relationship, and (2) Inserting values into two of three variables, with the S-relation identifying the value of the remaining one. However, if we want to identify two variables by inserting a value into only one variable, then the resultant values of identification will be infinite. Thus, as for an infinite S-relation, the evaluation modes are limited.

Consequently, we define an *evaluation graph*, which represents a mode to evaluate an S-relation (Figure 12), where the arrows indicate how values are inserted into the terms. For a finite S-relation, because all the values of terms can be identified, all arrows are pointing outward (Figure 13). On the other hand, for an S-relation with negation ($\neg$), all arrows are pointing inward (Figure 14) because the existence of an S-link cannot be determined unless the values of all terms are inserted.

Determination of an Evaluation Procedure

The search algorithm for valid evaluation procedures is as follows:

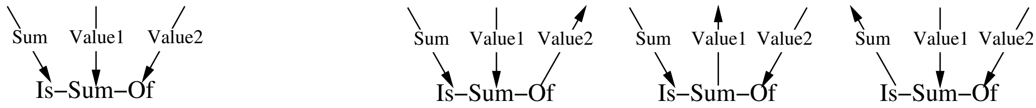
(1) Repeat the processes (2) – (4) for all possible combinations of evaluation graphs of the S-relations.

(2) Connect the evaluation graphs through variables.

(3) Traverse along arrows starting from the constants and finite S-relations to determine which variables can be identified.

(4) The path identifying all variables gives the evaluation procedure.

For example, the evaluation graphs shown in Figure 15 were obtained for the following description:



(a) Inserting Values into Three Terms (b) Inserting Values into Two Terms

FIGURE 12: Representation of the Evaluation Modes for the Is-Sum-Of S-Relation

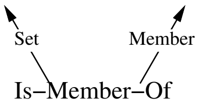


FIGURE 13: A Finite S-Relation

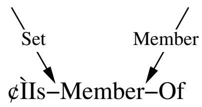
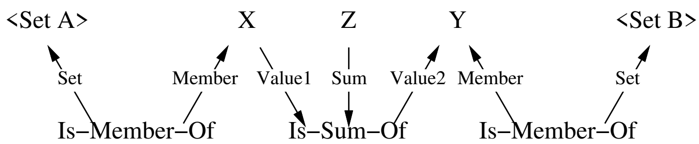
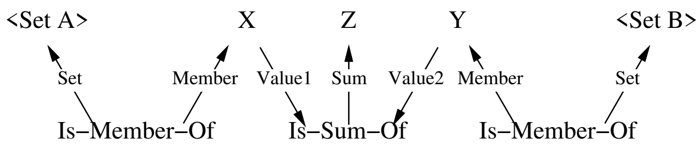


FIGURE 14: An S-Relation with Negation



(a) Not Providing An Evaluation Procedure



(b) Providing An Evaluation Procedure

FIGURE 15: Determination of an Evaluation Procedure

```
define Query [ Document = D, Title = T ]
   ← Is-Member-Of [ Set = ⟨Document⟩, Member = D ] ∧
      Is-Title-Of [ Document = D, Title = T ] ∧
      Is-Substring-Of [ Part = "database", Whole = T ].

foreach Query [ Document = D, Title = T ]
   print D, T
```

FIGURE 16: Example of a Query by Defining an Ad-Hoc S-Relation

```
Is-Member-Of [ Set = ⟨Set A⟩, Member = X ] ∧
Is-Member-Of [ Set = ⟨Set B⟩, Member = Y ] ∧
Is-Sum-Of [ Sum = Z, Value1 = X, Value2 = Y ]
```

In Figure 15(a), the variable Z cannot be identified, while in Figure 15(b), all the variables can be identified; thereby giving the evaluation procedure.

5 QUERY PROCESSING

Navigational Query

Because a network structure is employed for a semantic representation in the S-object database, the navigational query can be implemented like hypertexts, where target data is searched for by moving from one S-object to another by traversing along S-links.

Declarative Query

The declarative query can be executed by defining a new S-relation whose derivation rule is the search condition, e.g., to determine all documents whose title contains a string "`database`", we describe a query as shown in Figure 16

6 FUTURE WORK

Development of a Prototype System

We are now in designing a prototype system to verify the validity of the S-object database, where a full-text database is being adopted as the application. Our efforts are directed at the management of structured documents, such as SGML documents [ISO86], as well as implementing a user interface that provides for both navigational and declarative queries.

Semantic Aspects of Updates

The present study does not include the semantic aspects of updates to a semantic representation, and therefore we will focus on this aspect in parallel with prototype development.

Optimization

Because the S-object database has a drawback from the system efficiency standpoint, it is necessary to investigate corresponding enhancement techniques. One possible optimization method involves caching the result of the derivation of S-links.

7 CONCLUSIONS

An S-object database is presented as a database with an explicit semantic representation. The main characteristics of the S-object database is that a semantic representation and the physical data are managed separately, and that integrity constraints and queries can be described based on a semantic representation with predicate logic. With this approach, a semantic representation of a database can be flexibly handled. The S-object database is most suitable for the data having complex structures and various semantics, which arise in novel database applications such as full-text databases.

REFERENCES

[ABD+89] Atkinson, A., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., and Zdonik, S., "Object-Oriented Database System Manifesto," *Proc. of the 1st Int. Conf. on Deductive and Object-Oriented Databases* (Dec. 1989), 40–57.

[Ari90] Arisawa, H. (Ed.), "Deductive Databases," *J. Information Processing Society of Japan*, **31**, 2 (1990).

[DK79] Deliyanni, A. and Kowalski, R. A., "Logic and Semantic Networks," *Commun. ACM*, **22**, 3 (Mar. 1979), 184–192.

[GMN84] Gallaire, H., Minker, J., and Nicolas, J.-,M., "Logic and Databases: A Deductive Approach, " *ACM Comp. Surv.*, **16**, 2 (Jun. 1984), 153–185.

[ISO86] ISO 8879, *Standard Generalized Markup Language (SGML)*, (1986).

[Kim90] Kim, W., *"Introduction to Object-Oriented Databases,"* MIT Press (1990).

[Leh92] Lehmann, F., "Semantic Networks," *Computers and Mathematics with Applications*, **23**, 2-5 (1992), 1–50.

[Llo87] Lloyd, J. W., *Foundations of Logic Programming*, Springer-Verlag (1987).

[Min88] Minker, J. (Ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Publishers (1988).

[Nag90] Nagao, S. (Ed.), *Iwanami Information Science Dictionary*, Iwanami Shoten (1990).

[PM88] Peckham, J. and Maryanski, F., "Semantic Data Models," *ACM Comp. Surv.*, **20**, 3 (Sep. 1988), 153–188.

[TNH+91] Tanaka, K., Nishikawa, N., Hirayama, S., and Nanda, K., "Query Pairs As Hypertext Links," *Proc. of the 7th Int. Conf. on Data Engineering*, IEEE (Apr. 1991), 456–463.

[Ull88] Ullman, J., *Principles of Database and Knowledge-base Systems*, Computer Science Press, Vol. I/II (1988).

[YY92] Yokota, K. and Yasukawa, H., "Towards an Integrated Knowledge-Base Management System," *Proc. of the Int. Conf. on Fifth Generation Computer Systems 1992*, ICOT (Jun. 1992), 89–112.