Acta Cybernetica 13 (1998) 257-275.

On the Information Content of Semi-Structured Databases

Mark Levene *

Abstract

In a semi-structured database there is no clear separation between the data and the schema, and the degree to which it is structured depends on the application. Semi-structured data is naturally modelled in terms of graphs which contain labels which give semantics to its underlying structure. Such databases subsume the modelling power of recent extensions of flat relational databases, to nested databases which allow the nesting (or encapsulation) of entities, and to object databases which, in addition, allow cyclic references between objects.

Due to the flexibility of data modelling in a semi-structured environment, in any given application there may be different ways in which to enter the data, but it is not always clear when the semantics are the same. In order to compare different approaches to modelling the data we investigate a measure of the *information content* of typical semi-structured databases in order to test whether such databases are *information-wise equivalent.* For the purpose of our investigation we use a graph-based data model, called the hypernode model, as our model for semi-structured data and formalise flat, nested and object databases as subclasses of hypernode databases.

We use formal language theory to define the context-free grammar induced by a hypernode database, and then formalise the *information content* of such a database as the language generated by this context-free grammar. Intuitively, the information content of a database provides us with a measure of how flexible the database is in modelling the information from different points of view. This enables us to prove the following results regarding the expressive power of databases: (1) in general, hypernode databases and thus semi-structured databases express the general class of context-free languages, (2) the class of flat databases expresses the class of finite languages whose words are of restricted length between one and four, (3) the class of nested databases expresses the class of finite languages, and (4) the class of object databases expresses the general class of regular languages.

We then define two hypernode databases to be *information-wise equivalent* if they generate the same context-free language. This allows us to prove

*Department of Computer Science University College London Gower Street London WC1E 6BT, U.K. email: mlevene@cs.ucl.ac.uk

the following results regarding the computational complexity of determining whether two databases are information-wise equivalent or inequivalent: (1) the problem of determining information-wise equivalence of hypernode databases and thus semi-structured databases is, in general, undecidable, (2) the problem of determining information-wise equivalence of flat databases can be solved in time polynomial in the size of the two databases, (3) the problem of determining information-wise inequivalence of nested databases is NP-complete, and (4) the problem of determining information-wise inequivalence of object databases is PSPACE-complete.

Keywords semi-structured databases, flat databases, nested databases, object databases, information content, information-wise equivalence

1 Introduction

In a traditional data model such as the relational model [Cod79] there is a clear separation between the schema and the data itself. Recently it has been recognised that there are applications where the data is *self-describing* in the sense that it does not come with a separate schema, and the structure of the data, when it exists, has to be inferred from the data itself. Such data is called *semi-structured*, the Web providing us with a rich source of semi-structured data to experiment with. Semi-structured data is also useful when integrating several databases, some of which may be structured. In such an integration process the data may come from several different sources and thus it may be difficult to constrain the integrated database to a single unifying schema. (For two recent surveys on semi-structured data, which provide more motivation and examples of semi-structured databases, see [Abi97, Bun97].)

Semi-structured data is naturally modelled in terms of graphs which contain labels that give semantics to the underlying structure [Abi97, Bun97]. Herein we use the *hypernode model* [PL94, LL95] as our data model for semi-structured data. The hypernode model is well-suited for this task as it is a graph-based data model that supports both complex objects of arbitrary structure and cyclic references between such objects. There have been several other previous proposals for graph-based data models [KV85, CM90, GPG90], all having the common thread of modelling objects as graphs (or subgraphs) which can reference each other. (See [BH90] for the graph-theoretic terminology.) Intuitively a *hypernode database* (or simply a database) is a collection of directed graphs, called hypernodes, each such hypernode modelling a unique object in the database which can reference other hypernodes.

Traditionally flat databases, as in the relational model, have been sufficient to model most applications, but recently, it has been proposed to extend the modelling power of flat databases to nested (or complex object) databases [AFS89] which allow the nesting (or encapsulation) of entities, and to object-oriented databases [Kim90] which, in addition, allow cyclic references between objects. (For the purpose of this paper we concentrate on the data modelling aspects of objects and ignore the wider issues of object-orientation in databases.) The hypernode model, as are other semi-structured models [Abi97, Bun97], is more general than the above extensions to the flat relational model, and in the sequel we will define suitable restrictions of hypernode databases that allow us to model flat, nested and object databases.

Due to the flexibility of data modelling in a semi-structured environment, in any given application there may be different ways in which to enter the data, but it is not always clear when the semantics are the same. In order to compare different approaches to modelling the data we would like to measure the *information content* of typical semi-structured databases in order to test whether such databases are *information-wise equivalent*. Moreover, for practical purposes it is essential to know what the computational cost of testing for such equivalence is, given that the database designer may have to choose one of the representations for the actual database. In particular, it would be useful to compare the expressive power of the above mentioned extensions to the basic flat data model in terms of the information content of the databases which are in the subclass of databases induced by each such extension.

We illustrate the modelling power of hypernode databases with a running example showing part of a hypernode database, where the labels of hypernodes represent unique identifiers of hypernodes in the database, providing the means by which hypernodes can reference each other. The hypernode shown in Table 1, which is labelled *EMPS*, models an entity set of employees, where each entity in *EMPS* is represented by an isolated node in the hypernode. Correspondingly, the hypernode shown in Table 2, which is labelled *ED2*, models the subset of employees in *EMPS* working in the Maths department. The hypernode shown in Table 3 models the information pertaining to *EMP1*, where each attribute and value of *EMP1* is represented by an arc in the hypernode. Similarly, the hypernode labelled *DEPTS*, shown in Table 4, models an entity set of departments and the hypernodes labelled *DEPT1* and *DEPT2*, shown in Tables 5 and 6, respectively, model the information pertaining to *DEPT1* and *DEPT2*. Note that in *DEPT2* the actual address of the department is missing and also that it has the additional attribute *faculty* which is missing from *DEPT1*. Finally, the hypernode labelled *WORKS*, shown in Table 7, models the relationship of an employee working in a department, where each employee and their department is represented by an arc in the hypernode. We observe that nesting (or alternatively encapsulation) is achieved by referencing another hypernode from within a hypernode; for example, *EMP1* and *EMP2* are nested in *DEPT1*, and *ED2* and *EMP3* are nested in *DEPT2*. Note the difference in modelling the set of employees working in a department from within *DEPT1* and *DEPT2*. In addition, we observe that cyclic references are achieved by two hypernodes referencing each other; for example a cyclic reference exists between *EMP1* and *DEPT1*, since *EMP1* references *DEPT1* and *DEPT1* references *EMP1*.

We use formal language theory [HU79] in order to reason about the information content of databases by showing that a hypernode database induces a context-free grammar and thus generates a context-free language. We then define two hypernode

$EMPS$
$EMP1$
$EMP2$
$EMP3$
$EMP4$

Table 1: The entity set EMPS

$ED2$
$EMP3$
$EMP4$

Table 2: A subset of EMPS

EMP1		
(*attribute*	$\rightarrow$	*value*)
ename	$\rightarrow$	*john*
dept	$\rightarrow$	$DEPT1$
boss	$\rightarrow$	$EMP2$

Table 3: The entity EMP1

$DEPTS$
$DEPT1$
$DEPT2$

Table 4: The entity set DEPTS

databases to be *information-wise equivalent* if they generate the same context-free language. In general, the problem of information-wise equivalence of hypernode databases and thus semi-structured databases is undecidable, since we show that the general class of hypernode databases expresses the general class of context-free languages. Therefore, we restrict our attention to three subclasses of hypernode databases: flat databases, nested databases and object databases, all of which are defined as suitable syntactic restrictions of hypernode databases.

(For an interesting example of the use of formal languages in database theory, see [Shm93] wherein it was shown that the problem of determining equivalence of Datalog queries is undecidable, by reducing the equivalence problem for context-free grammars to this problem; see also [Ull92] which looks into additional relationships between logic rules and formal languages.)

We prove the following results regarding the expressive power of different classes of databases:

1. The class of flat databases expresses the class of finite languages whose words are of length at most four.

2. The class of nested databases expresses the class of finite languages.

3. The class of object databases expresses the general class of regular languages.

We establish the following results regarding the computational complexity of determining whether two databases are information-wise equivalent or inequivalent:

1. The problem of determining information-wise equivalence of flat databases can be solved in time polynomial in the size of the two databases.

DEPT1		
(*attribute*	$\rightarrow$	*value*)
dname	$\rightarrow$	*computing*
emp	$\rightarrow$	$EMP1$
emp	$\rightarrow$	$EMP2$
head	$\rightarrow$	$EMP2$
address	$\rightarrow$	*london*

Table 5: The entity DEPT1

DEPT2		
(*attribute*	$\rightarrow$	*value*)
dname	$\rightarrow$	*maths*
emp	$\rightarrow$	$ED2$
head	$\rightarrow$	$EMP3$
address		
faculty	$\rightarrow$	*science*

Table 6: The entity DEPT2

$WORKS$		
$EMP1$	$\rightarrow$	$DEPT1$
$EMP2$	$\rightarrow$	$DEPT1$
$EMP3$	$\rightarrow$	$DEPT2$

Table 7: The relationship WORKS

2. The problem of determining information-wise inequivalence of nested databases is NP-complete.

3. The problem of determining information-wise inequivalence of object databases is PSPACE-complete.

It follows that in terms of information-content, object databases are strictly more expressive than nested databases and nested databases are strictly more expressive than flat databases. The interpretation that we place on the notion of being more expressive is that it affords us with more flexible means of modelling information. With respect to our running example, we have modelled the fact that $EMP1$ works in $DEPT1$ in three different ways, through the relationship $WORKS$, through the attribute *dept* in $EMP1$ and through the attribute *emp* in $DEPT1$. (We note that if we view primary keys in the relational model [Cod79] as object-identifiers, then relational databases can easily represent our notion of object databases.)

The problem of measuring the information capacity of database schemas was investigated in [Hul86] in the context of the relational data model, in [HY84, AH88] in the context of a complex objects data model and in [KV85] in the context of a graph-based data model, which supports cyclic references between objects. In [Hul86] several notions of equivalence are considered. The most restrictive measure is *query equivalence*, which informally holds between two database schemas when for any query on the first schema there is an equivalent query on the second schema and vice versa, and the least restrictive measure is *absolute equivalence*, which informally holds between two database schemas when there is a one-to-one correspondence between the number of objects that can be constructed by using sets of domain values over the attributes of both schemas. In the context of complex objects, a complete set of restructuring operations on database schemas that preserve absolute equivalence was exhibited in [HY84, AH88]. Moreover, it was shown in [KV85] that every database schema that has cyclic references is query equivalent (with respect to a well-defined query language which is given in [KV93]) to a database schema *without any* cyclic references; the concluding remark in [KV85] is: "But it is not clear that this measure is the ultimate one. We believe that the issue of cycles deserves further study".

The first difference in our work in comparison to the work mentioned above is that we concentrate on the information content of individual databases at the instance level rather than on the information content of database schemas. Thus we measure the information content of each database without regards to its schema. This difference is not so fundamental when the data has an underlying structure. As can be seen from the running example, a typical hypernode database may induce a database schema over which it is defined. On the other hand, since a hypernode database is only semi-structured it may not be possible to compare the information content of hypernode databases with reference to a fixed schema. The second difference is that we concentrate on the data modelling issues without reference to query equivalence, and as a result our definition of information-wise equivalence seems to be incomparable to the various definitions of equivalence referred to in the above work. This difference is fundamental since, for example, a flat database may be query equivalent to a nested database, in the sense that for every query defined on the flat database there is an equivalent query on the nested database and vice versa, but not information-wise equivalent to it according to our definition of information-wise equivalence. Intuitively, as demonstrated in the running example, nesting and/or cyclic referencing affords the database user with more flexible means of data modelling and therefore with several alternative ways to query the same information. Moreover, we also investigate the computational complexity of determining information-wise equivalence which was not dealt with in the work mentioned above.

The layout of the rest of the paper is as follows. In Section 2 we formalise the concept of a hypernode database, which comprises our model for semi-structured data. In Section 3 we introduce the necessary background material from formal language theory and formalise the notion of the information content of a database. In Section 4 we define flat, nested and object databases and prove our results concerning their expressive power. In Section 5 we prove our results concerning the computational complexity of determining whether two databases are information-wise equivalent. Finally, in Section 6 we give our concluding remarks.

2 Hypernode Databases

We first introduce the basic concepts pertaining to hypernode databases. We refer the reader to [PL94, LL95] for more detail on the hypernode model including a computationally complete query and update language operating on hypernode databases.

Definition 2.1 (Hypernodes) We assume two finite and disjoint sets of constants are available. Firstly we have the set of *labels* $\mathbf{L}$ whose elements are denoted by strings beginning with uppercase letters. Secondly we have the set of *atomic values* (or simply values) $\mathbf{A}$ whose elements are denoted by strings beginning with lowercase letters.

A *hypernode* is defined to be an equation of the form

$$H = (N, E),$$

where $H \in \mathbf{L}$ is termed the *defining label* of the hypernode and (N, E) is a directed graph, termed *the graph of the hypernode*, such that $N \subseteq \mathbf{A} \cup \mathbf{L}$ is a set of nodes and $E \subseteq (N \times N)$ is a set of arcs. We denote the set of *isolated* nodes in N, i.e. the set of nodes which do not appear in any arc in E, by *isolated*(H).

Definition 2.2 (Hypernode databases) A *hypernode database* (or simply a database), say HD, is a finite set of hypernodes satisfying the following two conditions:

(H1) No two (distinct) hypernodes in HD have the same defining label.

(H2) For any label, say H, in the node set of a graph of a hypernode in HD there exists a hypernode in HD whose defining label is H.

We denote the set of all labels that appear in the hypernodes in HD by LABELS(HD) and the set of all atomic values appearing in the hypernodes in HD by ATOMIC(HD). Moreover, we assume that a (possibly empty) set of distinguished labels in LABELS(HD) is associated with HD, which we denote by *root*(HD).

We note that condition H1 above corresponds to the *entity integrity* requirement of the relational data model [Cod79], since each hypernode can be viewed as representing a real-world entity. In object-oriented terminology labels are unique and serve as system-wide object-identifiers [Kim90], assuming that all of the hypernodes known to the system are stored in a single database. Similarly, condition H2 corresponds to the *referential integrity* requirement of the relational data model [Cod79], since it requires that only existing entities be referenced. The intuition behind the set of labels in *root*(HD) is that they represent the set of objects in the database through which all other objects in the database can be accessed.

The *Hypernode Accessibility Graph* (HAG) of a hypernode $H = (N, E)$ in a hypernode database HD (or simply the HAG of H, whenever HD is understood from context) is the directed graph telling us which hypernodes in HD are nested in the hypernode whose defining label is H, when considering nesting as a transitive relationship.

Definition 2.3 (The accessibility graph of a hypernode) The HAG of H, denoted by (N_H, E_H), is the minimal directed graph which is constructed from hypernodes in HD as follows: $H \in N_H$, and if $H' \in N_H$ and $H' = (N', E') \in$ HD (such a hypernode must exist by condition H2), then $(\mathbf{L} \cap N') \subseteq N_H$ and $\forall n' \in (\mathbf{L} \cap N'), (H', n') \in E_H$.

A hypernode database HD is *acyclic* if for all $H \in root$(HD), the HAG of H is acyclic, otherwise HD is *cyclic*.

We close this section with the definition of two operations on hypernode databases which, in the next section, are shown to preserve the information content of the database.

Definition 2.4 (Renaming and duplication) A hypernode database HD$'$ is the result of *renaming* some of the hypernodes in a hypernode database HD, if HD$'$ can be obtained from HD by renaming some of the labels in LABELS(HD) to distinct labels in $\mathbf{L}$ − LABELS(HD).

A hypernode database HD$'$ is the result of *duplicating* some of the hypernodes in HD, if HD$'$ is the union of HD and a hypernode database that is obtained by renaming some of the hypernodes in HD.

3 Information Content of Databases

We next present our notion of the information content of a database and briefly introduce the relevant definitions and results from the theory of context-free grammars [HU79].

Definition 3.1 (Context-Free Grammar) A *context-free grammar* (CFG) is a quadruple (V, T, P, S), where V and T are finite and disjoint sets of *variables* and *terminals*, P is a finite set of *productions* of the form $X \to \alpha$ such that X is a variable and α is a finite and nonempty string of variables and terminals, and $S \in$ V is a distinguished variable called the *start symbol*.

A CFG (V, T, P, S) is said to be a *regular grammar* (RG), if each of the productions in P is either of the form $X \to \alpha Y$ or $X \to \alpha$, where α is a string of terminals (in the production $X \to \alpha Y$ α may be empty).

From now we will assume that G = (T, V, P, S) is a CFG and will refer to a finite string of variables and terminals as a *string* and to a finite string of terminals as a *word*. We define the *length* of a string α to be the number of symbols in α.

Definition 3.2 (Derivations) If β and γ are strings and $X \to \alpha$ is a production in P, then $\beta X \gamma$ *directly derives* $\beta \alpha \gamma$ in G, written $\beta X \gamma \Rightarrow \beta \alpha \gamma$.

We say that a string α *derives* a string β in G, written $\alpha \Rightarrow^* \beta$, if for some natural number $n \geq 0$

$$\alpha \Rightarrow \beta_1, \beta_1 \Rightarrow \beta_2, \ldots, \beta_n \Rightarrow \beta.$$

Thus $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$.

Definition 3.3 (The language generated by a CFG) The *context-free language* (or simply language) generated by G, denoted by $\mathcal{L}(\mathrm{G})$, is the set of all words that can be derived from the start symbol S in G; a context-free language $\mathcal{L}(\mathrm{G})$ is said to be a *regular* language if G is an RG.

Two CFGs, G_1 and G_2, are *equivalent* written $G_1 \equiv G_2$ if $\mathcal{L}(G_1) = \mathcal{L}(G_2)$, otherwise they are *inequivalent.*

We note that according to Definition 3.1 the right-hand side of productions is always nonempty and thus *we only consider languages where the empty word is not a member of* $\mathcal{L}(\mathrm{G})$.

The next lemma allows us to simplify the productions in an RG.

Lemma 3.1 Every RG, G, has an equivalent RG, G$'$, such that every production of G$'$ is either of the form $X \rightarrow Y$, $X \rightarrow aY$ or $X \rightarrow a$, where X and Y are variables and a is a terminal.

Proof. The result easily follows by an induction on the length of α in productions of the form $X \rightarrow \alpha Y$. Suppose that the length of α is greater than one and $\alpha = \beta a$ for some terminal symbol a. Then, replace the production $X \rightarrow \beta a Y$ with the two productions $X \rightarrow \beta Z$ and $Z \rightarrow aY$, where Z is a nonterminal not appearing in any other production in the resulting grammar. It is evident that the newly formed grammar is an RG and is equivalent to G. □

Definition 3.4 (Chomsky Normal Form) A CFG, G, is in *Chomsky Normal Form* (CNF) if all its productions are of the form $X \rightarrow YZ$ or $X \rightarrow a$, where X, Y and Z are variables and a is a terminal.

The next theorem is a well-known result [HU79].

Theorem 3.2 Every CFG, G, (such that $\mathcal{L}(\mathrm{G})$ does not contain the empty word) has an equivalent CFG, G$'$, which is in CNF and is equivalent to G.

Intuitively, the information content of a hypernode database HD is the context-free language that is generated by the CFG induced by G, and two hypernode databases are information-wise equivalent if they generate the same context-free language.

Definition 3.5 (Information content of databases) The CFG induced by a hypernode database HD, denoted by CFG(HD), is a quadruple (V, T, P, S), where $S \notin$ LABELS(HD), V = LABELS(HD) $\cup\{S\}$, T = ATOMIC(HD) and P is the smallest set of productions such that for every label $R \in root$(HD), $S \rightarrow R \in$ P, and for every hypernode $H = (N, E) \in$ HD, $H \rightarrow n \in$ P, if $n \in isolated(N)$ and $H \rightarrow n_1 n_2 \in$ P, if $(n_1, n_2) \in E$.

The language generated by HD, denoted by $\mathcal{L}$(HD), is the language generated by CFG(HD), i.e. $\mathcal{L}$(CFG(HD)).

The *information context* of a hypernode database HD is defined to be the language generated by HD. Two hypernode databases HD1 and HD2 are *information-wise equivalent* (or simply equivalent), denoted by HD1 $\equiv$ HD2, if CFG(HD1) $\equiv$ CFG(HD2), i.e. $\mathcal{L}$(HD1) = $\mathcal{L}$(HD2). Otherwise HD1 and HD2 are *information-wise inequivalent* (or simply inequivalent).

We note that the information content of a hypernode database may be the empty language, for example, if HD contains the single hypernode, $H = (\emptyset, \emptyset)$, or if $root$(HD) is empty.

The next proposition can be verified from Definitions 2.4 and 3.5.

Proposition 3.3 Equivalence of hypernode databases is closed under renaming and duplication.

We next show that every CFG can be represented by a hypernode database.

Definition 3.6 (The hypernode database representing a cfg) The hypernode database *representing* a CFG, G = (V, T, P, S), denoted by DB(G), is constructed as follows. Firstly, by Theorem 3.2, G is converted into an equivalent CFG in CNF, which we also refer to as G. Secondly, we assume that V $\subseteq$ **L** and that T $\subseteq$ **A**, and say that the production X $\rightarrow$ YZ $\in$ P *induces* the hypernode X = ({Y, Z}, {(Y, Z)}) and the production X $\rightarrow a \in$ P *induces* the hypernode X = ($\{a\}$, $\emptyset$). Finally, DB(G) is the smallest set of hypernodes induced by the productions in P and where $root$(DB(G)) = $\{S\}$.

The next proposition is now immediate.

Proposition 3.4 Two context-free grammars G_1 and G_2 are equivalent if and only if DB(G_1) and DB(G_2) are equivalent.

4 The Expressive Power of Database Classes

We are now ready to investigate the expressive power of various classes of hypernode databases in terms of the set of CFGs that they induce.

Definition 4.1 (Expressive power of classes of hypernode databases) A class of hypernode databases, $\mathbf{D}$, is said to *express* a class of context-free languages, $\mathbf{C}$, if for every hypernode database, HD $\in \mathbf{D}$, there is a CFG, G $\in \mathbf{C}$, such that $\mathcal{L}(\mathrm{HD}) = \mathcal{L}(\mathrm{G})$, and for every CFG, G $\in \mathbf{C}$, there is a hypernode database, HD $\in \mathbf{D}$, such that $\mathcal{L}(\mathrm{HD}) = \mathcal{L}(\mathrm{G})$.

A class, $\mathbf{D1}$, of hypernode databases is *more expressive* than a class, $\mathbf{D2}$, of hypernode databases, if the class of context-free languages that is expressed by $\mathbf{D1}$ is a proper superset of the class of context-free languages that is expressed by $\mathbf{D2}$. Two classes of hypernode databases, $\mathbf{D1}$ and $\mathbf{D2}$, are *equally expressive*, if both $\mathbf{D1}$ and $\mathbf{D2}$ express the same class of context-free languages.

The next lemma, which is an immediate consequence of Proposition 3.4 and Definition 3.6, establishes the expressive power of the general class of hypernode databases.

Lemma 4.1 The general class of hypernode databases, and thus the general class of semi-structured databases, expresses the general class of context-free languages. □

For the rest of this section we investigate the expressiveness of various classes of hypernode databases, which correspond to flat, nested and object databases.

Our view of flat databases corresponds closely to Chen's binary entity-relationship model presented in [Che84], which in its essence captures the fundamental notions of the more general entity-relationship model [Che76, MM90, Teo94]. (For the purpose of this paper we do not address the concepts of specialisation and generalisation which are important notions in the entity-relationship model.)

Definition 4.2 (Flat databases) A *flat* database is a hypernode database HD such that the hypernodes $H = (N, E)$ in HD are restricted to be one of the following types:

1. An *entity set*, where $N \subseteq \mathbf{L}$ and $E = \emptyset$.

2. A *value set*, where $N \subseteq \mathbf{A}$ and $E = \emptyset$.

3. An *entity*, where (N, E) is a bipartite graph [BH90], such that N is partitioned into two nodes sets N_1 and N_2, with $N \subseteq \mathbf{A}$ and none of the nodes in N_2 are isolated, and there exists an entity set $H' = (M', \emptyset)$ in HD with $H \in M'$; the nodes in N_1 are called the ***attributes*** of the entity represented by H and the nodes in N_2 are called the ***values*** of the entity represented by H.

4. A *relationship*, where $N = N_1 \cup N_2$, with (N, E) having no isolated nodes, and such that there exist two entity sets $H_1 = (M_1, \emptyset)$ and $H_2 = (M_2, \emptyset)$ in HD such that $N_1 \subseteq M_1$ and $N_2 \subseteq M_2$.

Moreover, *root*(HD) contains a (possibly empty) subset of the set of defining labels of the entity sets, value sets and relationships in HD.

For example, the hypernodes shown in Tables 1, 2 and 4 represent entity sets, the hypernodes shown in Tables 8 and 9 represent entities, the hypernode shown in Table 7 represents a relationship, and the hypernode shown in Table 10 represents a value set.

FLAT-EMP1		
(*attribute*	$\rightarrow$	*value*)
ename	$\rightarrow$	*john*
dept	$\rightarrow$	*computing*
boss	$\rightarrow$	*jack*

Table 8: The entity FLAT-EMP1

FLAT-DEPT1		
(*attribute*	$\rightarrow$	*value*)
dname	$\rightarrow$	*computing*
emp	$\rightarrow$	*john*
emp	$\rightarrow$	*jack*
emp	$\rightarrow$	*jill*
head	$\rightarrow$	*jack*
address	$\rightarrow$	*london*

Table 9: The entity FLAT-DEPT1

$EMP-VALUE$
jack
jill
john

Table 10: The value set EMP-VALUE

It can easily be verified from Definition 4.2 that flat databases are acyclic and that such databases have no nesting of entities or relationships. Moreover, we observe that we represent attributes of entities by atomic values; see the hypernodes of the running example, shown in Tables 3, 5 and 6.

The next lemma thus follows from Definition 3.5 and 4.2.

Lemma 4.2 The class of flat databases expresses the class of finite languages having nonempty words of length less than or equal to four.

We next define grouped databases which modify flat databases such that attribute values of entities are modelled by grouping them into value sets.

Definition 4.3 (Grouped databases) A *grouped* database HD is a variation of a flat database, where the definition of an entity is modified, as follows:

3. A hypernode $H = (N, E)$ is an *entity*, where (N, E) is a bipartite graph, such that N is partitioned into two nodes sets N_1 and N_2, with $N_1 \subseteq \mathbf{A}$, $N_2 \subseteq \mathbf{L}$ and none of the nodes in N_2 are isolated, there exists an entity set $H_1 = (M_1, \emptyset)$ in HD with $H \in M_1$, and for all $n \in N_2$, there exists a value set $H_2 = (M_2, \emptyset)$ such that $n = H_2$.

For example, the hypernode shown in Table 11 represents and entity in a grouped database.

GROUPED-DEPT1		
$(attribute$	$\rightarrow$	$value)$
$dname$	$\rightarrow$	$computing$
emp	$\rightarrow$	$EMP-VALUE$
$head$	$\rightarrow$	$jack$
$address$	$\rightarrow$	$london$

Table 11: The entity GROUPED-DEPT1

The next result follows from Definitions 4.2 and 4.3 on using Definition 4.1.

Lemma 4.3 The classes of flat databases and grouped databases are equally expressive.

Our view of nested databases is to extend flat databases by allowing nesting of entities. In particular, we disallow the nesting of relationships, since otherwise by part (2) of Theorem 5.2, which is given in Section 5, equivalence of such databases would be intractable.

Definition 4.4 (Nested databases) A *nested* database HD is an extension of a flat database such that the definition of an entity is modified as follows, with the restriction that HD is acyclic:

3. A hypernode $H = (N, E)$ is an *entity*, where (N, E) is a bipartite graph, such that N is partitioned into two nodes sets N_1 and N_2, with $N_1 \subseteq \mathbf{A}$, $N_2 \subseteq \mathbf{A} \cup \mathbf{L}$ and none of the nodes in N_2 are isolated, there exists an entity set $H_1 = (M_1, \emptyset)$ in HD with $H \in M_1$, and for all $n \in N_2$, with $n \in \mathbf{L}$, there exists an entity set $H_2 = (M_2, \emptyset)$ such that either $n = H_2$ or $n \in M_2$.

For example, the hypdenodes shown in Tables 5 and 6 may represent entities in a nested databases. In this case the employee entities in the nested database may *not* reference either of the departments in order that HD be acyclic.

We observe that in nested databases we allow only the nesting of entity sets and entities. The next result follows from Definitions 3.5 and 4.4 on using Lemma 3.1, noting that any finite language can be generated by an RG.

Lemma 4.4 The class of nested databases expresses the class of finite languages.

The next corollary follows from Lemmas 4.2 and 4.4.

Corollary 4.5 The class of nested databases is more expressive than the class of flat databases.

Our view of object-oriented databases is to extend nested databases by allowing cycles as long as these do not involve relationships. This restriction is essential, since otherwise by part (1) of Theorem 5.2, which is given in Section 5, equivalence of such databases would be undecidable.

Definition 4.5 (Object databases) An *object* database is an extension of a nested database such that the database may be cyclic.

For example, the database shown in the running example in Section 1 is an object database.

We observe that as is the case of nested databases we disallow nesting of relationships in object databases.

Lemma 4.6 The class of object databases expresses the general class of regular languages.

Proof. By Definitions 3.5 and 4.5 on using Lemma 3.1, it is easy to see that the class of object databases is at least as expressive as the general class of regular languages. It remains to show that relationships do not add expressive power to the class of object databases. By Definition 4.5 relationships are not nested and thus any derivation of a word which uses a production such as $R \rightarrow X_1X_2$ must be of the form

$$S \Rightarrow R \Rightarrow X_1X_2 \Rightarrow^* w,$$

where no other production of the form $R' \rightarrow X_1'X_2'$ is used in the derivation. Therefore, $w = w_1w_2$, where for $i = 1$ and 2, $X_i \Rightarrow^* w_i$, implying that w_i is a member of the language induced by the RG with start symbol X_i. The result now follows, since RGs are closed under concatenation [HU79]. □

The next corollary follows from Lemmas 4.4 and 4.6.

Corollary 4.7 The class of object databases is more expressive than the class of nested databases.

5 The Complexity of Determining Equivalence of Databases

Herein we investigate the complexity of determining equivalence of hypernode databases for the classes of databases defined in Section 4. We assume that the reader is familiar with the notion of undecidability [HU79] and fundamental computational complexity classes NP (nondeterministic polynomial time), PSPACE (polynomial space) and NEXPTIME (nondeterministic exponential time) [GJ79]. (We define the *size* of a set S to be the cardinality of a standard encoding of S.)

Theorem 5.1 The following statements regarding the computational complexity of decision problems for CFGs are true:

(1) Equivalence of CFGs is undecidable [HU79, Theorem 8.12] (see also [HRS79]).

(2) Equivalence of CFGs which generate finite languages is NEXPTIME-hard [HRS79, Theorem 4.5].

(3) Inequivalence of RGs which generate finite languages is NP-complete [Hun73, Theorem 2.3].

(4) Inequivalence of RGs is PSPACE-complete [Hun73, Theorem 3.8].

The next theorem presents the results of this section.

Theorem 5.2 The following statements regarding the computational complexity of decision problems for hypernode databases are true:

(1) Equivalence of hypernode databases is undecidable.

(2) Equivalence of acyclic hypernode databases is NEXPTIME-hard.

(3) Equivalence of flat databases can be tested in polynomial time in the size of the two databases.

(4) Inequivalence of nested databases is NP-complete.

(5) Inequivalence of object databases is PSPACE-complete.

Proof. (1) and (2) are immediate consequences of Proposition 3.4 and parts (1) and (2) of Theorem 5.1, noting that acyclic hypernode databases are finite.

(3) Let HD be a flat database. We show that the size of the language generated by HD is polynomial in the size of HD, implying the result. Let m_1 be the number of entities and value sets in HD, m_2 be the maximal number of arcs and isolated nodes in any entity or value set in HD, m_3 be the number of relationships in HD and m_4 be the maximal number of arcs in any relationship in HD. Now, let m be the maximum of m_i, for $i = 1, 2, 3$ and 4. Thus the number of words in $\mathcal{L}$(HD) is bounded above by $3m^4$, since we need to count the number of words induced by

entity sets, value sets and relationships. The result now follows, since by Lemma 4.2, the length each word in $\mathcal{L}$(HD) is at most four.

(4) NP-hardness follows by Proposition 3.4 and part (3) of Theorem 5.1, on using Lemma 4.4. It remains to show that the equivalence problem for nested databases is in NP.

Given a nested database HD, the maximal length of words in $\mathcal{L}$(HD) is bounded above by twice the size of HD, since we disallow nesting of relationships. Now, let HD1 and HD2 be nested databases and nondeterministically guess a word, say w, whose length is less than or equal to the maximal length of words in either CFG(HD1) or CFG(HD2) and such that its atomic values are in ATOMIC(HD1) $\cup$ ATOMIC(HD2). The result now follows, since membership of a word w in a CFG can be decided in polynomial time in the length of w [HU79].

(5) PSPACE-hardness follows by Proposition 3.4 and part (4) of Theorem 5.1, on using Lemma 4.6. It remains to show that the inequivalence problem for object databases is in PSPACE.

Let HD be an object database. If there are no relationships in HD, the result follows from part (4) of Theorem 5.1, since it can easily be verified that CFG(HD) is an RG. Otherwise, suppose that due to a relationship whose defining label is R we have the production $R \rightarrow X_1 X_2$ in CFG(HD). Due to the fact that relationships cannot be nested, it follows that for $i = 1$ and 2, any derivation, $X_i \Rightarrow^* w$ of a word w, is induced by an RG whose start symbol is X_i. Thus a derivation $R \Rightarrow^* w$ of a word $w \in \mathcal{L}$(HD) can be viewed as the derivation of two words w_1 and w_2 such that $w_1 w_2 = w$ and for $i = 1$ and 2, w_i is a word in the RG induced by X_i. Moreover, R can be chosen nondeterministically from the set of defining labels of relationships in HD. Thus the inequivalence of two object databases HD1 and HD2 reduces to the problem of finding a word $w = w_1 w_2$, as above, which is a member of one of the languages $\mathcal{L}$(HD1) or $\mathcal{L}$(HD2), but is not a member of the other language. The result now follows by part (4) of Theorem 5.1, since both w_1 and w_2 can be derived by RGs in PSPACE. □

6 Concluding Remarks

We have investigated the information content of semi-structured databases and shown that the general class of databases expresses the general class of context-free languages, the class of object databases expresses the general class of regular languages, the class of nested databases expresses the class of finite languages, and the class of flat databases expresses the class of finite languages whose words are of length less than or equal to four. Moreover, we have shown that testing the equivalence of hypernode databases and thus semi-structured databases is, in general, undecidable, but for object databases it is PSPACE-complete, for nested databases it is NP-complete and for flat databases it is polynomial time in the size of the input. Our results support the view that relationships are *not* entities, since otherwise, if we allow relationships to be nested within entities, by parts (1) and (2) of Theorem 5.2 determining equivalence of nested databases would be intractable and determining equivalence of object databases would be undecidable.

The interpretation we place on the notion of being more expressive is that it affords us with more flexible means of modelling information. (We refer the reader back to the running example given in the introduction to verify this statement.) From the user's point of view this flexibility provides several alternative ways of viewing and querying the same information; for example, the fact that an employee works in a department can be modelled in three different ways. Moreover, this flexibility may be an advantage for the query optimiser, when there are several alternative routes to obtain an answer to a query. Although testing for equivalence of object and nested databases is, in general, intractable we can provide restructuring operations as in [HY84, AH88] in order to transform a database into an equivalent one having a different structure. The formulation of a complete set of restructuring operations that preserve information-wise equivalence for object and nested databases is an open problem.

We now briefly outline, through an example, an extension to measure the navigation capacity of hypernode databases, and thus semi-structured databases. Let HD1 be a hypernode database comprising the hypernodes with defining labels A and B shown in Tables 12 and 13, respectively, and let HD2 be a hypernode database comprising the hypernodes with defining labels C and D shown in Tables 14 and 15, respectively. It can easily be verified that both $\mathcal{L}(\text{HD1}) = \mathcal{L}(\text{HD2}) = \{a, b\}$, and thus HD1 and HD2 are information-wise equivalent. In this case the nesting of hypernodes does not increase the information-content of the database. Despite this equivalence, from a navigation point of view HD1 is less expressive than HD2, since in HD1 we cannot directly navigate from A to B or from B to A, while in HD2 it is possible to navigate either directly from C to D or directly from D to C. Thus information content on its own is insufficient to measure expressiveness of a database from the point of view of navigation. We suggest to utilise the hypernode accessibility graph (HAG) for this purpose (see Definition 2.3). In our example, it is evident that with respect to navigation HD2 is more expressive than HD1, since HD1 $\equiv$ HD2 and the HAGs of A and B are subgraphs of the HAGs of C and D, respectively, up to an appropriate renaming of labels.

A
a

Table 12: The hypernode labelled A

B
b

Table 13: The hypernode labelled B

C
a
D

Table 14: The hypernode labelled C

D
b
C

Table 15: The hypernode labelled D

Another open problem is to extend our formalism to deal with integrity constraints such as keys and cardinality constraints. Finally, we mention that an important application of our formalism is in software engineering process modelling [CKO92], as it was shown in [LSO97] that the graph-based approach of the hypernode model provides a suitable platform for such process modelling.

References

[Abi97] S. Abiteboul. Querying semi-structured data. In *International Conference on Database Theory*, pages 1-18, Delphi, 1997. Invited talk.

[AFS89] S. Abiteboul, P.C. Fischer and H.-J. Schek, editors. *Nested Relations and Complex Objects in Databases*, volume 361 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1989.

[AH88] S. Abiteboul and R. Hull. Restructuring hierarchical database objects. *Theoretical Computer Science*, 62:3-38, 1988.

[BH90] F. Buckley and F. Harary. *Distance in Graphs*. Addison-Wesley, Redwood City, Ca., 1990.

[Bun97] P. Buneman. Semistructured data. *Proceedings of ACM Symposium on Principles of Database Systems*, Tucson, Az., 1997. Invited talk.

[Che76] P.P-S. Chen. The Entity-Relationship model - towards a unified view of data, *ACM Transactions on Database Systems*, 1:9-36, 1976.

[Che84] P.P-S. Chen. An algebra for a directional binary entity-relationship model. In *Proceedings of IEEE International Conference on Data Engineering*, pages 37-40, Los Angeles, 1984.

[CKO92] W. Curtis and M.I. Kellner and J. Over. Process Modelling. *Communications of the ACM*, 35:79-90, 1992.

[CM90] M.P. Consens and A.O. Mendelzon. Graphlog : A visual formalism for real life recursion. In *Proceedings of ACM Symposium on Principles of Database Systems*, pages 406-416, Nashville, Tn., 1990.

[Cod79] E.F. Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, 4:397-434, 1979.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, New York, 1979.

[GPG90] M. Gyssens and J. Paredaens and D. Van Gucht.A graph-oriented object database model. In *Proceedings of ACM Symposium on Principles of Database Systems*, pages 417-424, Nashville, Tn., 1990.

[HRS79] H.B. Hunt III, D.J. Rosenkrantz and T.G. Szymanski.On the equivalence, containment, and covering problems for regular and context-free languages. *Journal of Computer and System Sciences*, 12:222-268, 1976.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, Ma., 1979.

[Hul86] R. Hull. Relative information capacity of simple relational database schemata. *SIAM Journal on Computing*, 15:856-886, 1986.

[Hun73] H.B. Hunt III. On the time and tape complexity of languages I. *Proceedings of ACM Symposium on Theory of Computing*, pages 10-19, Austin, Tx., 1973.

[HY84] R. Hull and C.K. Yap. The format model: A theory of database organization. *Journal of the ACM*, 31:518-537, 1984.

[Kim90] W. Kim. *Introduction to Object-Oriented Databases*, MIT Press, Cambridge, Ma., 1990.

[KV85] G.M. Kuper and M.Y. Vardi. On the expressive power of the logical data model. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 180-187, Austin, Tx., 1985.

[KV93] G.M. Kuper and M.Y. Vardi. The logical data model. *ACM Transactions on Database Systems*, 18:379-413, 1993.

[LL95] M. Levene and G. Loizou. A graph-based data model and its ramifications. *IEEE Transactions on Knowledge and Data Engineering*, 7:809-823, 1995.

[LSO97] M. Levene, L. Scott and R. Offen. A framework for seamless conceptual data and process modelling. Research Report, Research Report No. 97/5, Joint Research Centre for Advanced Systems Engineering, CSIRO - Macquarie University, 1997.

[MM90] V.M. Markowitz and J.A. Makowsky. Identifying extended entity-relationship object structures in relational schemas. *IEEE Transactions on Software Engineering*, 16:777-790, 1990.

[PL94] A. Poulovassilis and M. Levene. A nested-graph model for the representation and manipulation of complex objects. *ACM Transactions on Information Systems*, 12:35-68, 1994.

[Shm93] O. Shmueli. Equivalence of Datalog queries is undecidable. *Journal of Logic Programming*, 15:231-241, 1993.

[Teo94] T.J. Teorey. *Data Modelling & Design: The Fundamental Principles* Morgan-Kaufmann, San Francisco, Ca., 2nd edition, 1994.

[Ull92] J.D. Ullman. The interface between language theory and database theory. In J.D. Ullman, editor, *Theoretical Studies in Computer Science*, pages 133-151, Boston, Ma., 1992. Academic Press

Received October, 1997