THE LOGIC OF A RELATIONAL DATA MANIPULATION LANGUAGE

Marco A. Casanova* and Philip A. Bernstein**
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138

ABSTRACT

A logic for a relational data manipulation language is defined by augmenting a known logic of programs with rules for two new statements: the relational assignment, which assign a relational expression to a relation, and the random tuple selection, which extracts an arbitrary tuple from a relation. The usual operations on relations-retrieve, insert, delete, update-are then defined as special cases of the relational assignment, and the for-each construct scanning a relation tuple by tuple is introduced with the help of the random tuple selection.

1. Introduction

A database contains data that models some aspect of the world. The description of a database consists of a set of data structure descriptions and a set of consistency criteria for data values. To say that the data values in a database satisfy the consistency criteria is to say that the data adequately models the world. As a consequence, users expect to observe consistent data and are required to submit updates that will preserve consistency. Such updates are called *transactions* [ES2].

Most of the work concerned with the correctness of multi-user database systems [BE1,ES2,LA2, PA1,ST1,TH1] assumes that only transactions access the database and curb the interaction among transactions by a general correctness criterion. However, we know of no attempt to provide a data manipulation language (DML) with a logic permitting us to prove that a transaction indeed preserves consistency, or that a set of transactions executed concurrently is correct. A DML logic becomes especially important when the set of transactions is known in advance, as the general correctness criterion can be tailored to the application in question and verified beforehand [LA2]. Examples of such database applications are credit card verification, airline and hotel reservation, point-of-sale inventory control and electronic funds transfer [BE2].

We will investigate in this paper the logic of a (sequential) data manipulation language; in the future, we plan to embed it in a general parallel language in the manner of [OW1]. We view our work as the first step towards studying the correctness of database applications such as those quoted above.

To support our DML logic, we start by selecting a framework that is adequate to describe databases and that has a reasonable underlying logic. In fact, we will argue that the Relational Model [CO2] is so close to Predicate Calculus that the former can be subsumed by the latter [CA1,MI1,NI1,VA1]. Hence, the question of the underlying logic is solved gratuituously.

As for the DML logic, we will augment a known logic of programs, Dynamic Logic [PR1], with rules for two new statements: the *relational assignment*, assigning a relational expression to a relation, and the *random tuple selection*, extracting an arbitrary tuple from a relation. The familiar operations [CH1,CO1,HE1,LA1]--retrieve, insert, delete, update--will be defined as special cases of the relational assignment. The for-each construct of [SC1] will also be given a translational definition.

We finally note that our commitment to Dynamic Logic is not essential, and our results should be translatable to any logic of (non-deterministic) programs.

2. A Logic for Data of Type Relation

As mentioned in the Introduction, we choose the Relational Model [CO2] as our framework for describing databases mainly because it can be assimilated into Predicate Calculus. By "assimilated" we mean that the key notions of the former can be rephrased as some of the very basic concepts of the latter (see Fig. 1). The similarities between the Relational Model and Predicate Calculus have profitably been used before [CA1,MI1,VA1], but the identification of relational schema and first-order theories, although clear, appeared only recently (and independently) [NI1].

To substantiate the claims of Fig. 1, let us first review some concepts. In the Relational Model, stored data is viewed as relations (unordered tables) and is described by *relation schemes* giving names to the tables and their columns. An *integrity assertion* then becomes a relational expression written in some appropriate language. A *relational schema* R is a set of relation schemes and a set of

*Research supported by the "Conselho Nacional de Pesquisas, CNPq-Brasil", Grant 1112.1248/76.
**Research supported by the National Science Foundation under Grant MCS-77-05314.

integrity assertions. If a set S of relations, one for each relation scheme of R, satisfies all integrity assertions, then S is called a *consistent database state* of R.

FIGURE 1

1. Relational Model
2. schema
   - 2.1 attribute name
   - 2.2 relation name
   - 2.3 integrity assertion
3. consistent state
   - 3.1 union of underlying domains
   - 3.2 relation extension

1. Predicate Calculus
2. theory
   - 2.1 standard unary predicate symbol
   - 2.2 non-standard predicate symbol
   - 2.3 non-logical axiom
3. model
   - 3.1 domain
   - 3.2 relation

Hence, we have paved the way to the following definition:

DEFINITION 2.1. A first-order theory $T = (L_T, D_T)$ is called a *relational schema* iff

(i) L_T, the set of non-logical symbols of T, includes a finite set of predicate symbols $R_i \in \mathbb{P}_{k_i}$ $(1 \leq i \leq m)$ and $A_j \in \mathbb{P}_1$ $(1 \leq j \leq n)$;

(ii) D_T, the non-logical axioms of T, includes for each R_i exactly one axiom of the form (where $1 \leq i_p \leq n$)

$$\forall x_1 \ldots \forall x_{k_i} \left(R_i(x_1, \ldots, x_{k_i}) \Rightarrow \bigwedge_{p=1}^{k_i} A_{i_p}(x_p) \right)$$

Moreover, we say that:

- R_i is a *database relation name* and A_j is an *attribute name* $(1 \leq i \leq m;\ 1 \leq j \leq n)$;
- the special axiom for R_i is the *relation scheme of* R_i (which we abbreviate $R_i[A_{i_1}, \ldots, A_{i_{k_i}}]$);
- $I \in D_T$ is an *integrity assertion*;
- a model S of T is a *consistent database state*. □

However, not all integrity assertions can be translated as axioms. The so-called *dynamic integrity assertions* [ES1,HA1] impose restrictions on the possible state transitions of the database, rather than on the set of acceptable states. A well-known example is:

(1) the salary of each employee listed in table EMP must be non-decreasing.

In [ES1], the prefixes 'NEW-' and 'OLD-', as in NEW-EMP and OLD-EMP, were proposed to indicate which state must provide the value of EMP. We do not judge this approach elegant because it alters the definition of interpretation from Predicate Calculus in a fundamental way. Therefore, we propose a novel rewrite for dynamic integrity assertions, using pairs of formulae, that avoids this problem. (1) is then rewritten as follows:

(2) $(\forall e \forall s (EMP_O(e,s) \equiv EMP(e,s)),$ $\forall e \forall s \forall s' (EMP(e,s) \wedge EMP_O(e,s') \Rightarrow s > s'))$ where EMP_O is a defined predicate symbol whose sole purpose is to freeze the value of EMP in the initial state (that is, EMP_O plays the role of OLD-EMP).

We then define the notion of schema with dynamic integrity assertions as follows:

DEFINITION 2.2. A triple $T = (L_T, D_T, D_T')$ is called a *relational schema with dynamic integrity assertions* iff

(i) (L_T, D_T) is a relational schema;

(ii) D_T' is a finite set of pairs of formulae of L_T, the *dynamic integrity assertions*. □

The role of relational schemas will become clear later on when we discuss the notion of transaction. For the time being, it suffices to remember that stored data is viewed as relations.

Finally, we introduce the notion of a *key* of a relation:

DEFINITION 2.3. Let $T = (L_T, D_T)$ be a relational schema and $R_i[A_{i_1}, \ldots, A_{i_{k_i}}]$ be the relation scheme of $R_i \in L_T$. A *key* of R_i is a set $k \subset [1, k_i]$ such that

$$\vdash_T \forall \bar{x} \forall \bar{y} \left(R(\bar{x}) \wedge R(\bar{y}) \wedge \bigwedge_{j \in k} x_j = y_j \Rightarrow \bar{x} = \bar{y} \right)$$

where $\bar{x} = \{x_1, \ldots, x_{k_i}\}$, $\bar{y} = \{y_1, \ldots, y_{k_i}\}$. □

In words, the value of R_i at any consistent database state of T must be a relation such that if two tuples agree on the coordinates in k, they are equal. Note that we allow subsets of k to be themselves keys of R_i. Hence, our definition, although sufficient for our purposes, differs from that in [CO2].

3. A Logic for a Data Manipulation Language

Our efforts in this section are directed towards defining a DML logic by augmenting Dynamic Logic [PR1] with rules for two new statements--the relational assignment and the random tuple selection. The logic thus obtained will be called Relational Dynamic Logic (RDL).

In §3.1 we discuss the overall architecture of RDL and then proceed to introduce the two new statements in §§3.2 and 3.3.

3.1 The Architecture of Relational Dynamic Logic

Let $DL[L_E, L_A, D_A, U]$ be the Dynamic Logic over L_E, L_A, D_A and U (see Appendix I). We assume that there is a set of distinguished predicate symbols of L_E, L_R, the *stored relation names*. We construct RDL by adding two new statements to DL, the relational assignment and the random tuple selection, together with their corresponding axioms, as discussed in §§3.2 and 3.3. To stress the parameters of Relational Dynamic Logic, we write $RDL[L_R, L_E, L_A, D_A, U]$.

We say that $RDL[L_R, L_E, L_A, D_A, U]$ has an *adequate* universe iff

(i) for any variable x of L_E and any individual $a \in V_u$, there is a state $I \in U$ such that $x_I = a$;

(ii) for any binary stored relation name $R \in \mathbb{P}_n$ and any binary relation $R^* \subset V_u^n$, there is a state $I \in U$ such that $R_I = R^*$.

The motivation for this concept follows closely the note in Appendix I and will become clear when we discuss relational assignments.

Finally, we say that a schema $T = \langle L_T, D_T \rangle$ is a *schema of RDL* iff L_T coincides with L'_A, the database relation names and attribute names of T are stored relation names, and the underlying deductive system of T is D_A. Similarly, we define a *dynamic schema of RDL*.

3.2 The Relational Assignment

Examples of relational assignments, using the relation schemes BOOKS[ISBN,EDITOR,AUTHOR,TITLE] and WRITERS[NAME,CITY], are:

(1) BOOKS1(a,t) := ∃i∃e(BOOKS(i,e,a,t)
∧ WRITERS(a,'PISA'))

that retrieves into BOOKS1 authors and titles of all books written by Pisa citizens;

(2) WRITERS(n,c) := ∃c'(WRITERS(n,c')
∧ (n ≠ 'GALILEI' ⇒ c = c')
∧ (n = 'GALILEI' ⇒ c = 'PADUA'))

that updates the city listed with Galilei to Padua. Likewise, we could give examples of insertions and deletions written as relational assignments. Thus, we achieve a certain economy by adding the relational assignment to DL, as just one statement suffices to describe the operations commonly provided by existing relational DMLs [CH1,CO1,HE1, LA1] (i.e., retrieve, insert, delete and update).

Our approach to the relational operations is further justified by the simplicity of the relational assignment:

DEFINITION 3.1. Formation Rules for the Relational Assignment.

$$"R(\bar{x}) := A[\bar{x}]" = \{(I,J) \in U^2 / I \underset{R}{\equiv} J \wedge (\forall \bar{a} \in V_u^n)(R_J(\bar{a}) \text{ iff } I \models A[\bar{a}/\bar{x}])\}$$

is called a *relational assignment* of RDL[L_R,L_E,L_A, D_A,U], where R is an n-ary predicate symbol of L_R, $\bar{x} = \{x_1,\ldots,x_n\}$ is a set of distinct variables of L_E, and $A[\bar{x}]$ is an open wff of L_E with free variables $x_i \in \bar{x}$. □

Hence, (I,J) is in the binary relation associated with $"R(\bar{x}) := A[\bar{x}]"$ iff I and J differ only on R and R_J consists of those tuples $\bar{a} \in V_u^n$ such that, when $\bar{x}$ is given $\bar{a}$ as value, A becomes valid in I. The central result about relational assignments goes as follows:

THEOREM 3.1. Suppose that RDL[L_R,L_E,L_A,D_A,U] has an adequate universe. Then, for any wff Q of L_A, $\models_U [R(x) := A[x]]Q \equiv Q_R^A$ where Q_R^A denotes the wff of L'_A obtained by replacing each atomic formula in Q of the form $R(t_1,\ldots,t_n)$ by $A'[t_1/x_1,\ldots,t_n/x_n]$, where A' is a variant of A created by renaming bound variables of A that also occur in Q. □

An example may clarify the construction of Q_R^A:

(3) s = "BOOKS1(a,t) := ∃i∃e(BOOKS(i,e,a,t)
∧ WRITERS(a,'PADUA'))"

(4) Q ≡ ∃i(BOOKS1('GALILEI',i) ∧ INDEX-PROHIBITORUM(i))

(5) [s]Q ≡ ∃i(∃i'∃e(BOOKS(i',e,'GALILEI',i)
∧ WRITERS('GALILEI','PADUA'))
∧ INDEX-PROHIBITORUM(i))

Calling A the right-hand side of s, the equivalence in (5) is obtained by replacing BOOKS1('GALILEI',i) in Q by A'['GALILEI'/a,i/t], where A' is created by renaming i in A by i' (avoiding conflict with the use of i in Q).

The usual relational operations are defined as follows:

DEFINITION 3.2. Relational Operations.

(a) *retrieve* $R(\bar{x})$ *where* $A[\bar{x}] = "R(\bar{x}) := A[\bar{x}]"$, where R does not occur in A

(b) *insert* $R(\bar{x})$ *where* $A[\bar{x}] = "R(\bar{x}) := R(\bar{x}) \vee A[\bar{x}]"$

(c) *delete* $R(\bar{x})$ *where* $A[\bar{x}] = "R(\bar{x}) := R(\bar{x}) \wedge \sim A[\bar{x}]"$

(d) *update* $R(\bar{f}(\bar{x}))$ *where* $A[\bar{x}] = "R(\bar{x}) := \exists \bar{y}(R(\bar{y}) \wedge (\sim A[\bar{y}] \Rightarrow \bar{x} = \bar{y}) \wedge (A[\bar{y}] \Rightarrow \bar{x} = \bar{f}(\bar{y})))"$. □

The translation of the update, the only non-trivial one, should be read as follows (where R is the relation name affected in a transition from I to J):

(6) *for each* tuple $\bar{y}_I$ *in* R_I *do*
if $\bar{y}_I$ does not satisfy A
then add $\bar{y}_I$ to R_j;
else add $\bar{f}_I(\bar{y}_I)$ to R_j;

3.3 The Random Tuple Selection and the For-each Construct

The purpose of introducing the random tuple selection is confined almost entirely to the definition of the for-each construct, the main theme of this section. The for-each construct has the same flavor as the synonymous construct of [SC1]:

(1) *for each* WRITERS(a,c) *where* c = 'USA' *key* NAME
do n := 0;
for each BOOKS(i',e',a',t') *where* a' = a *key* ISBN
do n := n + 1 *end*;
if n > 10 *then display* a;
end

that only displays American authors with more than 10 books published.

In terms of a loose ALGOL with the relational assignment, the for-each construct is defined as follows:

(2) *for each* $R(\bar{x})$ *where* $A[\bar{x}]$ *key* K *do* s *end* =

A. $R_O(\bar{y}) := false \wedge R_O(\bar{y})$;
B. $R_T(\bar{y}) := R(\bar{y}) \wedge A[\bar{y}]$;
C. *while* $\exists \bar{y}(R_T(\bar{y}))$ *do*
D. *begin* RANDOMTUPLE (R_T;$\bar{x}$);
E. $R_O(\bar{y}) := R_O(\bar{y}) \vee \bar{y} = x$;
F. s;
G. $R_T(\bar{y}) := R(\bar{y}) \wedge A[\bar{y}] \wedge \sim\exists \bar{z}(R_O(\bar{z}) \wedge \bar{z}[K] = \bar{y}[K])$;
end

where $\bar{x} = \{x_1,\ldots,x_n\}$ is a set of distinct variables of L_E; K encodes a key of R; and R_O and R_T are

auxiliary predicate symbols of L_R.

The definition of the for-each sketched above can be explained as follows:

A. the set of tuples already scanned is the value of R_O (equal to $\emptyset$ initially);

B. the set of tuples eligible for scanning comprises the value of R_T (equal to the set of tuples in the initial value of R and satisfying A, initially);

C. the loop will terminate when all tuples have been scanned.

D. a random tuple is selected from those not yet scanned, hence giving the system freedom for implementing the scanning operation;

E. the tuple selected above is added to the value of R_O;

F. s is executed;

G. as s can modify the value of R, we have to recompute the set of tuples not yet scanned. This is done with the help of the key K as follows: a tuple in the current value of R is eligible for scanning if it satisfies A and, moreover, its key value does not match that of any tuple already scanned (hence the third conjunct in G).

In order to express (2) in DL, we need the relational assignment, which we already have, tests over quantified wff's, which are readily obtained, and a new statement, the *random tuple selection*, defined as:

DEFINITION 3.3. Formation Rule for the Random Tuple Selection.

$$\text{"RANDOMTUPLE}(R;\bar{x})\text{"} = \{(I,J) \in U^2 / I \underset{\bar{x}}{=} J \wedge R_I(\bar{x}_J)\}$$

is called a *random tuple selection* of $RDL[L_R,L_E,L_A,D_A,U]$, where R is an n-ary predicate symbol of L_R and $\bar{x} = \{x_1,\ldots,x_n\}$ are distinct variables of L_E. □

Hence, (I,J) is in the binary relation associated with "RANDOMTUPLE(R;$\bar{x}$)" iff I and J differ only on $\bar{x}$, and $\bar{x}_J$ comes from R_I. The central result about the random tuple selection goes as follows:

THEOREM 3.2. Suppose that $RDL(L_R,L_E,L_A,D_A,U]$ has an adequate universe. Then, for any wff Q of L_A, $\models_U [\text{RANDOMTUPLE}(R;\bar{x})]Q \equiv \forall\bar{x}(R(\bar{x}) \Rightarrow Q)$. □

We emphasize that the random tuple selection is in itself uninteresting, but it gives precision to the definition of the for-each construct sketched in (2).

DEFINITION 3.4. The for-each Construct

for each $R(\bar{x})$ *where* $A[\bar{x}]$ *key* K *do* s *end* =

$R_O(\bar{y}) := false \wedge R_O(\bar{y})$;
$R_T(\bar{y}) := R(\bar{y}) \wedge A[\bar{y}]$;
$(\exists\bar{y}(R_T(\bar{y}))?;$ RANDOMTUPLE$(R_T;\bar{x})$;
$R_O(\bar{y}) := R_O[\bar{y}] \vee \bar{y} = \bar{x}$;
s;
$R_T(\bar{y}) := R(\bar{y}) \wedge A[\bar{y}] \wedge \neg\exists\bar{z}(R_O(\bar{z}) \wedge \bigwedge_{i\in k} z_i = y_i))^*$;
$\forall\bar{y}(\neg R_T(\bar{y}))?$

is called a *for-each construct* of $RDL[L_R,L_E,L_A,D_A,U]$, where R, R_O and R_T are n-ary predicate symbols in L_R (R_O and R_T are chosen anew for the translation of each for-each construct occurring in a program), $\bar{x} = \{x_1,\ldots,x_n\}$ are distinct variables of L_E, $A[\bar{x}]$ is a wff of L_E with free variables $x_i \in \bar{x}$, $K \subset \{1,\ldots,n\}$ is a non-minimal key of R, s is a statement of RDL. □

Although no special rule for reasoning about the *for-each* construct is needed, an induction rule on the number of tuples already scanned has been found useful [CA2]. The basic idea lies in introducing an invariant with certain properties in order to hide the translation contained in Definition 3.4. With such a rule one can prove, for instance, that $P \Rightarrow [w]Q$, where:

(3) P = *true*
Q = $\forall i'\forall a'\forall t'$(BOOKS$_O$(i','SPRINGER',a',t')
$\equiv$ BOOKS(10i','SPRINGER',a',t'))
w = *for each* BOOK(i,e,a,t) *key* EDITOR,AUTHOR, TITLE
do if e = 'SPRINGER'
then update BOOKS(10i',e',a',t')
where i' = i $\wedge$ e' = e $\wedge$ a' = a $\wedge$ t' = t;
end

using as inductive assertion:

(4) I = $\forall i'\forall a'\forall t'$(BOOKS$_O$(i','SPRINGER',a',t')
$\Rightarrow$ BOOKS(10i','SPRINGER',a',t'))

The rule we have in mind is stated below.

THEOREM 3.3. Suppose that $RDL[L_R,L_E,L_A,D_A,U]$ has an adequate universe. Then Rule FE stated below is valid in U

(FE)

1. $\neg\exists\bar{y}(R_O(\bar{y})) \wedge P \Rightarrow I$,

2. $R(\bar{x}) \wedge A[\bar{x}] \wedge I' \Rightarrow [s]I$,

3. $$\frac{\forall\bar{x}(R(\bar{x}) \wedge A[\bar{x}] \Rightarrow \exists\bar{y}(R_O(\bar{y}) \wedge \bigwedge_{i\in K} y_i = x_i)) \wedge I \Rightarrow Q}{P \Rightarrow [w]Q}$$

where
R, R_O are predicate symbols in L_R, as in Definition 3.4,
P is a wff of L_A, possibly containing R_O,
I is a wff of L_A, possibly containing R_O but with no occurrences of $x_i \in \bar{x}$,
I' is obtained from I by replacing each atomic formula of I of the form $R_O(\bar{t})$ by $R_O(\bar{t}) \wedge \bar{x} \neq \bar{t}$,
and
w = "*for each* $R(\bar{x})$ *where* $A[\bar{x}]$ *key* K *do* s *end*". □

Rule FE should be read as follows (we remind that R_O holds the tuples in R already processed):

— Premisse 1 asserts that, before entering the *for-each*, P and the fact that no tuple was processed must imply I.
— Premisse 2 asserts that if s is started in a state J where $\bar{x}_J$ holds a tuple in R_J satisfying A, $\bar{x}_J$ is already in R_O (see Definition 3.4) and I holds for all tuples in R_{O_J}, except $\bar{x}_J$, then s must lead to state K where I holds for all tuples in R_{O_K} (including $\bar{x}_J$).
— Premisse 3 asserts that, when the *for-each* ends, I and the fact that all tuples have been

processed must imply Q.

3.4 Summary

The relational data manipulation language we consider in this paper is created by adding two new basic statements to those considered in DL, the relational assignment and the random tuple selection. The corresponding logic of programs, called Relational Dynamic Logic (RDL), is obtained by adding two new axioms to the system of DL:

(RA) $[R(\bar{x}) := A[\bar{x}]]Q \equiv Q^A_R$

(RS) $[\text{RANDOMTUPLE}(R;\bar{x})]Q \equiv \forall\bar{x}(R(\bar{x}) \Rightarrow Q)$

Theorems 3.1 and 3.2 assert that Axioms RA and RS are sound and, moreover, that the assertion language is "expressive" (in the sense of [HA2]) for the new statements. Hence, following [HA2], we can state:

THEOREM 3.4. RDL Arithmetical Completeness. Suppose that $RDL[L_R,L_E,L_A,D_A,U]$ has an adequate universe, L_A contains arithmetic, each state in the universe U assigns the standard interpretation to arithmetical symbols and the deductive system for L_A is complete for U. Then, for any wff Q of RDL, $\models_U Q$ iff $\vdash_{RDL} Q$. □

The relational operations--retrieve, insert, delete, update--and the for-each construct, as defined symbols of RDL, need no special treatment. However, an induction rule (omitted for reasons of brevity) on the number of tuples already scanned can be stated in order to hide some details of the definition of the for-each construct.

4. Transactions

Given a relational dynamic schema $T = (L_T,D_T,D'_T)$, we say that a program s is a *transaction* w.r.t. T if s preserves the consistency of the database described by T, that is, iff s satisfies two conditions:

(i) s maps consistent states into consistent states;

(ii) for each dynamic integrity assertion $(A,B) \in D'_T$, if s starts in a state satisfying A, s must terminate in a state satisfying B.

It would certainly be helpful if we could find a formal characterization of (i) and (ii). But as we noted in §3.4, Relational Dynamic Logic is arithmetically complete (in the sense of [HA2], with the necessary provisos stated in Theorem 3.4). Therefore, we have:

THEOREM 4.1. Let RDL be constructed as in Theorem 3.4 and $T = (L_A,D_T,D'_T)$ be a relational dynamic schema of RDL. Then, a program s of RDL is a transaction iff

(i) for any axiom $A \in D_T$, $D_T \vdash_{RDL} [s]A$

(ii) for any dynamic integrity assertion $(A,B) \in D'_T$, $D_T \vdash_{RDL} A \Rightarrow [s]B$. □

Thus, at least in theory, we will not fail to prove that a program is a transaction due to weaknesses of RDL. Evidently, the much more difficult question of finding a proof procedure for our logic remains unanswered.

5. Conclusion

In retrospect, Dynamic Logic was augmented to include a relational data manipulation language, creating a logic of programs that we called Relational Dynamic Logic (RDL). RDL contains two axioms, in addition to those of DL, for the relational assignment and the random tuple selection. These new axioms are sound and the whole formal system can be made complete, with certain provisos as in [HA2].

The virtue we see in our work lies in its simplicity: the four operations on relations--retrieve, insert, delete, update--were defined in terms of the relational assignment and the for-each construct was accounted for with just one more statement, the random tuple selection.

Finally, from the point of view of program verification, the Relational Model was shown to provide a reasonable programming language interface to databases. No special purpose logic is needed, as the Relational Model can be understood as an application of the very basic concepts of Predicate Calculus.

REFERENCES

[BE1] Bernstein, P.A. *et al.* "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases", Computer Corporation of America Tech. Rep. 77-09, Dec. 1977.

[BE2] Bernstein, P.A. *et al.* "An Approach to the Design and Implementation of Interactive Information Systems", (Draft, Nov. 1977).

[CA1] Cadiou, J.M. "On Semantic Issues in the Relational Model of Data", Proc. Int. Symp. on Math. Foundations of Computer Science, Gdansk, Poland, "Lecture Notes in Computer Science", Springer-Verlag, Sept. 1976.

[CA2] Casanova, M.A. "The Logic of a Data Manipulation Language" (technical report in progress).

[CH1] Chamberlin, D.D. *et al.* "SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control", IBM Tech. Rep. RJ1798, June 1976.

[CO1] Codd, E.F. "A Database Sublanguage Founded on the Relational Calculus", Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control.

[CO2] Codd, E.F. "A Relational Model of Data for Large Shared Data Banks", CACM 13, 6, June 1970.

[CO3] Cook, S.A. "Axiomatic and Interpretative Semantics for an ALGOL Fragment", TR-79, Univ. of Toronto, Feb. 1975.

[ES1] Eswaran, K.P. and D.D. Chamberlin. "Functional Specifications of a Subsystem for Database Integrity", Proc. of the Int. Conf. on Very Large Data Bases, 1975.

[ES2] Eswaran, K.P. *et al.* "The Notion of Consistency and Predicate Locks in a Data Base System", CACM 19, 11, Nov. 1976 (pp. 624-633).

[HA1] Hammer, M.M. and D.J. McLeod. "Semantic Integrity in a Relational Data Base Management System", Proc. of the Int. Conf. on Very Large Data Bases, 1975.

[HA2] Harel, D. "Arithmetical Completeness in Logics of Programs", MIT Laboratory for Computer Science TM-103, April 1978.

[HE1] Held, G.D., M.R. Stonebraker and E. Wong. "INGRESS--A Relational Database System", Proc. AFIPS National Computer Conf., 1975 (pp. 409-416).

[LA1] Lacroix, M. and A. Pirotte. "Domain Oriented Relational Languages", Proc. of the Int. Conf. on Very Large Data Bases, 1977 (pp. 370-378).

[LA2] Lamport, L. "Towards a Theory of Correctness for Multi-User Data Base Systems", Massachusetts Computer Associates Tech. Rep. CA-7610-0712, Oct. 1976.

[MI1] Minker, J. "Search Strategy and Selection Function for an Inferential Relational System", ACM TODS 3, 1, March 1978 (pp, 1-31).

[NI1] Nicolas, J. "First Order Logic Formalization for Functional, Multivalued and Mutual Dependencies", ACM SIGMOD Int. Conf. on Management of Data, May 1978 (pp. 40-46).

[OW1] Owicki, S.S. "Axiomatic Proof Techniques for Parallel Programs", Dept. of Comp. Science, Cornell Univ., Tech. Rep. 75-251, July 1975.

[PA1] Papadimitriou, C.H., P.A. Bernstein and J.B. Rothnie. "Some Computational Problems Related to Database Concurrency Control", Proc. of Conf. on Theoretical Comp. Science, Waterloo, Aug. 1977 (pp. 275-282).

[PR1] Pratt, V.R. "Semantical Considerations on Floyd-Hoare Logic", Proc. of the 17th IEEE Found. of Comp. Science Conf., Oct. 1976 (pp. 109-120).

[SC1] Schmidt, J.W. "Some High-Level Language Constructs for Data of Type Relation", ACM TODS 2, 3, Sept. 1977 (pp. 247-261).

[ST1] Stearns, R.E., P.M. Lewis and D.J. Rosenkrantz. "Concurrency Control for Data Base Systems", Proc. of the 17th IEEE Found. of Comp. Science Conf., Oct. 1976 (pp. 19-32).

[TH1] Thomas, R.H. "A Solution to the Update Problem for Multiple Copy Databases which Uses Distributed Control", BBN Tech. Rep. 3340, July 1975.

[VA1] van Emden, M.H. "Computational and Deductive Information Retrieval", Proc. of the IFIP Conf. of the Working Group on Formal Specifications of Prog. Lang., Aug. 1977 (pp. 17.1-17.19).

[YA1] Yasuhara, A. *Recursive Function Theory and Logic*, Academic Press, 1971.

APPENDIX I

Concepts from Dynamic Logic

We review here just what is essential to understand Sections 3 and 4, referring the reader to [PR1] for a full description of Dynamic Logic (DL for short).

Dynamic Logic is based on a few parameters: a first-order language L_E, the *expression language*, specifying the expressions allowed in programs; an extension of L_E, L_A, the *assertion language*, whose wffs are used to express correctness assertions; a deductive system D_A for formulae of L_A; and a *universe* U of states with common domain V_U, each *state* being a structure for L_A, with domain V_U, together with an assignment of values [YA1] for the variables of L_A. To stress these parameters of Dynamic Logic, we often write $DL[L_E,L_A,D_A,U]$.

A program in DL is defined as a binary relation on U. Two basic programs (or *statements*) are provided:

(1) "$x := t$" $= \{(I,J) \in U^2 / I \underset{x}{=} J \wedge x_J = t_I\}$ is called an *assignment*, where x is a variable and t is a term of L_E ($I \underset{x}{=} J$ indicates that I and J differ only on the value of x).

(2) "$P?$" $= \{(I,J) \in U^2 / I \models P\}$ is called a *test*, where P is a wff of L_E ($I \models P$ indicates that P is valid in I).

Programs are created by the following operations:

(3) $s_1;s_2$, $s_1 \cup s_2$ and s^*, the *composition*, *union* and *reflexive transitive closure* of their components.

In order to express facts about programs, a new formation rule is added to those of first-order languages:

(4) if P is a wff of DL or L_A and s is a program, then, $\langle s\rangle P$ is a wff of DL.

The notion of interpretation is extended to $\langle s\rangle P$ as follows:

(5) $I \models \langle s\rangle P$ if $\exists J((I,J) \in s \wedge J \models P)$
$[s]P$ is then introduced as $\neg\langle s\rangle\neg P$ with interpretation;

(6) $I \models [s]P$ iff $\forall J((I,J) \in s \Rightarrow J \models P)$

DL contains the following axioms and rules of inference, in addition to those of D_A (Q is a wff of L_A):

(7) $[x:=t]Q \equiv Q^t_x$

(8) $[P?]Q \equiv P \Rightarrow Q$

(9) $[s_1 \cup s_2]Q \equiv [s_1]Q \wedge [s_2]Q$

(10) $[s_i;s_2]Q \equiv [s_1][s_2]Q$

(11) Necessitation : $\dfrac{P \Rightarrow Q}{[s]P \Rightarrow [s]Q}$

(12) Invariance : $\dfrac{P \Rightarrow [s]P}{P \Rightarrow [s^*]P}$

$DL[L_E,L_A,D_A,U]$ has an *adequate* universe iff for any variable x of L_E and any individual $a \in V_U$, there is a state $I \in U$ such that $x_I = a$. For adequate universes, Axioms 7 to 10 are sound and Necessitation and Invariance are valid. We refer the reader to [HA2] for a discussion of the "relative completeness" [CO3] of DL.

Note: Universes are in principle arbitrary. However, when discussing a logic of programs, attention must be paid to the choice made. For instance, if no state I in U valuates x as 5, the binary relation associated with "x:=5" is empty, permitting us to derive $\models_U [x:=5]\mathit{false} \equiv \mathit{true}$. Hence, Axiom 7 is not sound in U. The notion of adequacy of universes cures this anomaly.

APPENDIX II

Proofs

Before proving Theorem 3.1, we state a lemma about the construction of M^A_R from M (as described in the statement of the theorem). We use $M \sim N$ to indicate that N is a variant of M (that is, N is obtained from M by renaming variables).

LEMMA 3.1. $\neg M^A_R \sim (\neg M)^A_R$

$M^A_R \vee N^A_R \sim (M \vee N)^A_R$

$\exists x M^A_R \sim (\exists x M)^A_R$

$(M^i_x)^A_x \sim (M^A_R)^i_x$

x is free in M and i is a constant.

(The proof follows trivially from the construction of M^A_R). □

THEOREM 3.1. Suppose that $RDL[L_R,L_E,L_A,D_A,U]$ has an adequate universe. Then, for any wff Q of L_A, $\models_U [R(\bar{x}):=A[\bar{x}]]Q \equiv Q^A_R$. □

Proof: Suppose that $RDL[L_R,L_E,L_A,D_A,U]$ has an adequate universe. Let Q be a wff of L_A and $s = "R(\bar{x}):=A[\bar{x}]"$ be a relational assignment of RDL. (In the sequel, V_U denotes the domain of U and n denotes the arity of R).

We observe initially that

(1) $\models_U Q^A_R \equiv [s]Q$

$= (\forall I \in U)(I \models Q^A_R \equiv [s]Q)$

$= (\forall I \in U)(I \models Q^A_R \equiv (\forall J \in U)((I,J) \in s \Rightarrow J \in Q))$

$= (\forall I \in U)(I \models Q^A_R \equiv s(I) \models Q)$

. by Def. 3.1 and RDL has an adequate universe, s is a total function on U.

Therefore, all we have to prove is that for $(I,J) \in s$:

(2) $I \models Q^A_R = J \models Q$

Moreover, by Def. 3.1, we know that $(I,J) \in s$ iff

(3) $I \underset{R}{\equiv} J \wedge (\forall \bar{a} \in V_U^n)(R_J(\bar{a})$ iff $I \models A[\bar{a}/\bar{x}])$

We then fix $(I,J) \in U^2$ satisfying (3) and prove (2) by induction on the height of Q.

Basis: Assume $Q = S(\bar{t})$, S a predicate symbol of L_A.

Case 1: $S = R$

(4) $J \models Q$

$= J \models R(\bar{t})$

$= R_J(\bar{t}_J)$

$= I \models A[\bar{t}_J/\bar{x}]$. by (3)

$= I \models A[\bar{t}_I/\bar{x}]$. by (3), $I \underset{R}{\equiv} J$ implies $\bar{t}_I = \bar{t}_J$

$= I \models A'[\bar{t}/\bar{x}]$. A' is a variant of A with no variable of Q bound

$= I \models Q^A_R$. construction of Q^A_R

Case 2: $S \neq R$ (follows similarly)

Induction step: Assume (2) for formulae of height less than k and let Q be of height k. We consider only the case where $Q = \exists x M$:

(5) $J \models Q$

$= J \models \exists x M$

$= J \models M^i_x$. i is the name of some individual of J, by def. of $J \models \exists x M$

$= I \models (M^i_x)^A_R$. induction hypothesis, as M^i_x and M have the same height

$= I \models (M^A_R)^i_x$. Lemma 3.1 and Variant Theorem

$= I \models \exists x M^A_R$. i is also the name of some individual of I, as I and J have the same domain, and by def. of $J \models \exists x N$

$= I \models (\exists x M)^A_R$. Lemma 3.1

$= I \models Q^A_R$. construction of Q^A_R

This concludes the proof. □

THEOREM 3.2. Suppose that $RDL[L_R,L_E,L_A,D_A,U]$ has an adequate universe. Then, for any wff Q of L_A, $\models_U [RANDOMTUPLE(R;\bar{x})]Q \equiv \forall\bar{x}(R(\bar{x}) \Rightarrow Q)$. □

Proof: Suppose that $RDL[L_R,L_E,L_A,D_A,U]$ has an adequate universe. Let Q be a wff of L_A and s = "$RANDOMTUPLE(R;\bar{x})$" be a random tuple selection of RDL. (In the sequel, V_U denotes the domain of U and n denotes the arity of R).

Let $I \in U$. Then we have:

(1) $I \mid= [s]Q$

$= (\forall J \in U)((I,J) \in s \Rightarrow J \mid= Q)$. def. of $I \mid= [s]Q$

$= (\forall J \in U)(R_I(\bar{x}_J) \wedge I \underset{\bar{x}}{=} J \Rightarrow J \mid= Q)$. def. 3.3

$= (\forall J \in U)(R_I(\bar{x}_J) \wedge I \underset{\bar{x}}{=} J \Rightarrow I \mid= Q[\bar{x}_J/\bar{x}])$. I and J differ only on $\bar{x}$

$= (\forall \bar{a} \in V_U^n)(R_I(\bar{a}) \Rightarrow I \mid= Q[\bar{a}/\bar{x}])$. RDL has an adequate universe

$= (\forall \bar{a} \in V_U^n)(I \mid= R_I(\bar{a}) \Rightarrow Q[\bar{a}/\bar{x}])$. def. of $I \mid= M \Rightarrow N$

$= I \mid= \forall\bar{y}(R(\bar{y}) \Rightarrow Q[\bar{y}/\bar{x}])$. def. of $I \mid= \forall\bar{y}M$

$= I \mid= \forall\bar{x}(R(\bar{x}) \Rightarrow Q)$

From (1) we can conclude our result:

(2) $(\forall I \in U)(I \models [s]Q = I \models \forall\bar{x}(R(\bar{x}) \Rightarrow Q))$. 1

iff $(\forall I \in U)(I \models [s]Q \equiv \forall\bar{x}(R(\bar{x}) \Rightarrow Q))$. def. of $I \mid= M \equiv N$

iff $\models_U [s]Q \equiv \forall\bar{x}(R(\bar{x}) \Rightarrow Q)$

This concludes the proof. □

THEOREM 3.3. Suppose that $RDL[L_R,L_E,L_A,D_A,U]$ has an adequate universe. Then Rule FE is valid in U. □

Proof: The proof is a straightforward application of the system of RDL. For brevity, we present just a summary in the style of [OW1].

P_1: $\{P\}$

w_1: $R_o(\bar{x}) := \mathit{false} \wedge R_o(\bar{x})$;

P_2: $\{\neg\exists\bar{x}(R_o(\bar{x})) \wedge P\}$

P_3: $\{\neg\exists\bar{x}(R_o(\bar{x})) \wedge I\}$

w_2: $R_T(\bar{x}) := R(\bar{x}) \wedge A[\bar{x}]$;

P_4: $\{\forall\bar{x}(R_T(\bar{x}) \equiv R(\bar{x}) \wedge A[\bar{x}] \wedge \neg\exists\bar{x}(R_o(\bar{x})) \wedge I\}$

P_5: $\{\forall\bar{x}(R_T(\bar{x}) \equiv R(\bar{x}) \wedge A[\bar{x}] \wedge \neg\exists\bar{y}(R_o(\bar{y}) \wedge \bar{y}_n = \bar{x}_n)) \wedge I\}$
$(\{\exists\bar{x}(R_T(\bar{x})) \Rightarrow P_5\}$

w_3: $\exists\bar{x}(R_T(\bar{x}))?;\{P_5\}$

P_6: $\{\forall\bar{x}(R_T(\bar{x}) \equiv R(\bar{x}) \wedge A[\bar{x}] \wedge \neg\exists\bar{y}(R_o(\bar{y}) \wedge \bar{y}_k = \bar{x}_k) \wedge I\}$

w_4: $RANDOMTUPLE(R_R;\bar{x})$;

P_7: $\{R(\bar{x}) \wedge A[\bar{x}] \wedge \neg\exists\bar{y}(R_o(\bar{y}) \wedge \bar{y}_k = \bar{x}_k) \wedge I\}$

P_8: $\{R(\bar{x}) \wedge A[\bar{x}] \wedge I\}$

w_5: $R(\bar{y}) := R(\bar{y}) \vee \bar{y} = \bar{x}$;

P_9: $\{R(\bar{x}) \wedge A[\bar{x}] \wedge I'\}$

w_6: s;

P_{10}: $\{I\}$

w_7: $R_T(\bar{y}) := R(\bar{y}) \wedge A[\bar{y}] i \neg\exists\bar{z}(R_o(\bar{z}) \wedge \bar{z}_k = \bar{y}_k)$
$\{P_5\}$

$U\ \{\neg\exists\bar{x}(R_T(\bar{x})) \Rightarrow P_5\} w_3'$: $\neg\exists\bar{x}(R_T(\bar{x}))?;\{P_5\})^*$

P_{11}: $\{\forall\bar{x}(\neg R_T(\bar{x})) \Rightarrow P_5\}$

w_8: $\forall\bar{x}(\neg R_T(\bar{x}))?;$

P_{12}: $\{\forall\bar{x}(\neg R_T(\bar{x})) \wedge P_5\}$

P_{13}: $\{\forall\bar{x}(\neg(R(\bar{x}) \wedge \neg\exists\bar{y}(R_o(\bar{y}) \wedge \bar{y}_k = \bar{x}_k)) \wedge I\}$

P_{14}: $\{Q\}$ □

THEOREM 3.4. RDL Arithmetical Completeness Suppose that $RDL[L_R,L_E,L_A,D_A,U]$ has an adequate universe, L_A contains arithmetic, each state in the universe U assigns the standard interpretation to arithmetical symbols and the deductive system for L_A is complete for U. Then, for any wff Q of RDL, $\models_U Q$ iff $\models_{RDL} Q$. □

Proof: We argue here that the completeness of DL [HA2] extends to the completeness of RDL. It follows from Theorems 3.1 and 3.2 that, for any relational assignment or random tuple selection s and any wff P and Q of L_A:

(1) there is a wff R of L_A such that $\models_U [s]Q \equiv R$

(2) $\models_U P \Rightarrow [s]Q$ iff $\vdash_{RDL} P \Rightarrow [s]Q$

(3) $\models_U P \Rightarrow \langle s\rangle Q$ iff $\vdash_{RDL} P \Rightarrow \langle s\rangle Q$

Hence, using the completeness of DL and the above results, by induction on the structure of a formula S of RDL and on the structure of a program s of RDL, we can prove:

(4) for any wff S of RDL, there is a wff S' of L_A such that $\models_U S \equiv S'$ ("expressibility" of L_A for RDL)

(5) for any program s of RDL and any wffs P and Q of L_A, $\models_U P \Rightarrow [s]Q$ iff $\vdash_{RDL} P \Rightarrow [s]Q$

(6) for any program s of RDL and any wffs P and Q of L_A, $\models_U P \Rightarrow \langle s\rangle Q$ iff $\vdash_{RDL} P \Rightarrow \langle s\rangle Q$

From (4), (5), (6) we can then conclude the result:

(7) for any wff P of RDL, $\models_U P$ iff $\vdash_{RDL} P$ □

THEOREM 4.1. Let RDL be constructed as in Theorem 3.4 and $T = \langle L_T, D_T, D_T'\rangle$ be a dynamic schema of RDL. Then, a program s of RDL is a transaction of T iff

(i) for any axiom $A \in D_T$, $D_T \vdash_{RDL} [s]A$; and

(ii) for any dynamic integrity assertion $(A,B) \in D'_T$, $D_T \vdash_{RDL} A \Rightarrow [s]B$ □

Proof: Let s be a program of RDL. Let $\bar{A} = \bigwedge_{A \in D_T} A'$, where A' is the universal closure of A. Then s is a transaction of T iff

(1) $(\forall I \in U)(\forall J \in U)(I \models \bar{A} \Rightarrow ((I,J) \in s \Rightarrow J \models \bar{A}))$ and for any $(A,B) \in D'_T$, $(\forall I \in U)(\forall J \in U)(I \models \bar{A} \Rightarrow (I \models A \Rightarrow ((I,J) \in s \Rightarrow J \models B)))$

By definition of $[s]Q$, (1) is equivalent to:

(2) $\models_U \bar{A} \Rightarrow [s]\bar{A}$ and, for any $(A,B) \in D'_T$, $\models_U \bar{A} \Rightarrow (A \Rightarrow [s]B)$

By Theorem 3.4 and assumptions on RDL, (2) is equivalent to:

(3) $\vdash_{RDL} \bar{A} \Rightarrow [s]\bar{A}$ and, for any $(A,B) \in D'_T$, $\vdash_{RDL} \bar{A} \Rightarrow (A \Rightarrow [s]B)$

Finally, by the Deduction Theorem and $\bar{A}$ is closed, s is a transaction iff

(4) $\bar{A} \vdash_{RDL} [s]\bar{A}$ and, for any $(A,B) \in D'_T$, $\bar{A} \vdash_{RDL} A \Rightarrow [s]B$. □