Steps Toward a Formalism for the Formulary Model
David Wartman
Lance J. Hoffman
Stanford University

Introduction

In a previous memo Hoffman [1969] described a system for controlling the access to a large data base in an interactive environment. Simply stated, the system consists of a set of procedures (called a formulary) which interact with the user and control his access to the data base. In this memo we present some mathematical notation which we hope will be of use in discussing the problems of data base access in general and formularies in particular.

Definitions and Notation

Let $p_i$ represent a formulary and $q_j$ represent a data item. Assume for convenience that the $q_j$'s represent atomic (indivisible) quanta of data. That is, the $q$'s stand for the smallest pieces of data which may be accessed individually. Any set of data items not possessing this atomic property may be simply transformed into a (larger) set with this property. Assume for tractability that there is a finite set of $m$ formularies (1 ≤ $i$ ≤ $m$) and a finite set of $n$ data items (1 ≤ $j$ ≤ $n$).

Define a pseudo length function on formularies and data items.

\[
\begin{align*}
|p_i| &= \text{the number of bits required to represent formulary } p_i. \\
|q_j| &= \text{the number of bits required to represent data item } q_j.
\end{align*}
\]
Let $M_p$ be the total number of bits needed to represent formularies, $M_q$ be the total number of bits needed to represent data items, and $M$ be the total number of bits of memory (core or rotating) available.

\[ M_p = \sum_{k \geq 1} |P_k| \quad \quad M_q = \sum_{j \geq 1} |q_j| \]

Clearly we are constrained to have:

\[ M \geq M_p + M_q \]

The storage efficiency $\eta$ of a formulary system can be defined as:

\[ \eta = \frac{M_q}{M_p + M_q} \]

$\eta$ is a measure of how much of the storage used by the system is actually used for storing useful data.

We note that $M_p/m$ is the average number of bits per formulary and $M_q/n$ is the average number of bits per data item. For each data item $q_j$, there are some number of formularies which allow some level of access to it. Note that the association between data items and formularies can be:

a) one-many

\[ q_{13} \rightarrow P_{19} \]
\[ q_{13} \rightarrow P_{23} \]
\[ q_{19} \rightarrow P_{23} \]

\[ q_{19} \rightarrow P_{37} \]

b) many-one

\[ q_{17} \leftarrow P_{17} \]
\[ q_{17} \leftarrow P_{11} \]
\[ q_{18} \leftarrow P_{11} \]
c) more complex

\[ q_1 \leftarrow p_1 \]
\[ q_2 \leftarrow p_2 \]
\[ q_3 \leftarrow p_3 \]
\[ q_4 \leftarrow p_4 \]
\[ q_5 \leftarrow p_5 \]
\[ q_6 \leftarrow p_6 \]

Define a predicate on data items and formularies:

\[ \text{PERMIT}(q, p) \text{ is true iff formulary } p \text{ allows access to data item } q. \]

The value of the predicate is effectively computable since we are dealing with finite sets of formularies and data items and it is possible to try to access each data item with each formulary.

Using this predicate we can define some set relations on formularies and data:

Let \( S_j \) be the index set of formularies which allow access to \( q_j \):

\[ S_j = \{ x \mid \text{PERMIT}(q_j, p) \} \]

Let \( T_1 \) be the index set of data items accessed by \( p_1 \):

\[ T_1 = \{ y \mid \text{PERMIT}(q_j, p_1) \} \]

The set operations of intersection and union have the following interpretations:

\[ S_a \cup S_b \] is the index set of formularies which access \( q_a \) or \( q_b \) (or both).

\[ S_a \cap S_b \] is the index set of formularies which access both \( q_a \) and \( q_b \).

\[ T_a \cup T_b \] is the index set of data items accessed by \( p_a \) or \( p_b \) (or both).

\[ T_a \cap T_b \] is the index set of data items accessed by both \( p_a \) and \( p_b \).
In example c) above:

\[ S_1 = [1] \quad T_1 = (1, 2, 3, 6) \]
\[ S_2 = [1] \quad T_2 = (3, 4, 5) \]
\[ S_3 = [1, 2, 3] \quad T_3 = (3, 5, 6) \]
\[ S_4 = [2] \]
\[ S_5 = [2, 3] \]
\[ S_6 = [1, 3] \]

The protective overhead \( V_j \) for a data item \( q_j \) is defined to be the number of bits of formulary needed to control access to \( q_j \):

\[
V_j = \sum_{i \in S_j} |p_i|
\]

Now

\[
M_p \leq \sum_{j=1}^{n} \sum_{i \in S_j} |p_i| = \sum_{j=1}^{n} V_j
\]

with equality attained iff

\[
(\forall x, y \quad (T_x \cap T_y = \emptyset)) \land (\forall s \exists w \quad (PERMIT(q_s, p_w)))
\]

(i.e., no two formularies control a common data item and there are no formularies which do not control at least one data item).

Similarly we may define the data space \( U_i \) attributed to a formulary \( p_i \) as the number of bits of data accessed by \( p_i \):

\[
U_i = \sum_{j \in T_i} |q_j|
\]

The analogous relation for \( M_q \) is:

\[
M_q \leq \sum_{i=1}^{m} \sum_{j \in T_i} |q_j| = \sum_{i=1}^{m} U_i
\]
with equality iff

\[( \forall x \forall y \ (s_x \cap s_y = \emptyset)) \land (\forall w \exists z \ (\text{PERMIT}(z, \text{P}_x)))\]

(each data item is controlled by only one formulary and there are no inaccessible data items).

**Application to a Common Problem**

One problem which is often studied in conjunction with systems which allow multiple concurrent access to a data base is the problem of simultaneous updating. This situation arises when two users of the data base attempt to change the value of a particular data item at the same time. In a previous memo [Hoffman 1969], techniques were outlined which would solve the problem by allowing a user to LOCK a data item and make it inaccessible to other users. Here we will present a formal condition as to when simultaneous updating can occur.

Consider an interactive environment with some number of formularies actively being invoked by various users to access a common data base. Define

\[R_t = \{x \mid p_x \text{ is active at time } t\}\]

A sufficient (overly restrictive) condition to preclude simultaneous updating of any data item at a particular time \(t\) is:

\[(\forall x \forall y \ ((\{x \in R_y\} \cap \{y \in R_x\}) \supset ((x \neq y) \supset (p_x \land p_y = \emptyset)))\]

**Data on a Practical System**

The quantities defined above can be used to describe memory usage in systems which use the formulary model. In a system which uses memory efficiently we would expect to have \(N << M\). Inefficient memory usage would occur when \(M \gg M_x\).
In the Stanford Health Service system the following data has been observed:

\[
\begin{align*}
M_p & \sim 8.0 \times 10^6 \text{ bits} \\
M_q & \sim 6.4 \times 10^6 \text{ bits} \\
M_p/m & \sim 2.66 \times 10^4 \text{ bits} \\
M_q/n & \sim 640 \text{ bits} \\
\mathcal{T} & = \frac{6.4 \times 10^6}{6.48 \times 10^5} = 0.988
\end{align*}
\]

Reference

Hoffman, Lance J. (1969). "Formularies -- Program Controlled Privacy in Large Data Bases", CUTFM No. 60, Stanford Linear Accelerator Center, Computation Group, Stanford, California, February 1969