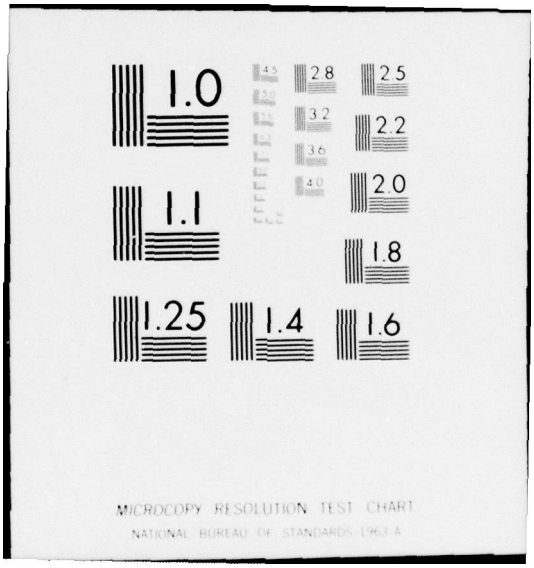
AD-A037 822 NAVAL RESEARCH LAB WASHINGTON D C F/G 12/1
THE BENEFITS OF CONTEXT IN PROCESSING SETS.(U)
MAR 77 G A WILSON
UNCLASSIFIED NRL-MR-3470 NL

1 OF 1
ADA037822

END

DATE FILMED
4-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NRL Memorandum Report 3470

The Benefits of Context in Processing Sets

GERALD A. WILSON

Computer Science Laboratory
Communications Sciences Division

March 1977

NAVAL RESEARCH LABORATORY
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: NRL Memorandum Report 3470	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): THE BENEFITS OF CONTEXT IN PROCESSING SETS	5. TYPE OF REPORT & PERIOD COVERED: Interim report on a continuing NRL problem.
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Gerald A. Wilson	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Research Laboratory, Washington, D.C. 20375	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NRL Problem B02-23
11. CONTROLLING OFFICE NAME AND ADDRESS: NRL-MR-3470	12. REPORT DATE: March 1977
	13. NUMBER OF PAGES: 18
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 19p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Sets
Set union
Set intersection
Algorithm analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The problem of finding unions and intersections of sets in an efficient manner pervades many areas of computer science. A profusion of examples can be found in both business and scientific computer applications. This paper discusses the importance of tailoring the algorithm to be employed to the intended application. A simple union and intersection algorithm is analyzed in detail, and the results of this analysis are compared to several similar analyses published by other researchers. It is shown that some relatively small changes in the nature of the problem conditions and/or the

→ next page

(Continues)

20. Abstract (Continued)

cont →

algorithm can lead to significant changes in the expected amount of work required to accomplish the desired task.

CONTENTS

THE BENEFITS OF CONTEXT IN PROCESSING SETS

1. Introduction

The problem of finding unions and intersections of sets in an efficient manner pervades many areas of computer science. A profusion of examples can be found in both business and scientific computer applications and research. In the processes of sorting and merging of sets too large to fit in memory, the most efficient of the algorithms all depend on dividing the items to be sorted into two or more groups to form a single sorted list of items. Indexing schemes employing inverted files require unioning and intersecting of lists of data record pointers obtained from the indexed keywords. Sussman and McDermot [8] brought to light the importance of set unions and intersections in AI research problems by the careful treatment of sets and bags (unordered lists ala Feldman [2]) in CONNIVER. In a working paper from MIT, Scott Fahlman [1] has suggested that the problem of set intersection is so important that special hardware and representation schemes should be developed to make set intersections a basic machine operation on a par with floating point multiplications or division. In the course of research aimed at the development of efficient and effective techniques for representing and searching very large knowledge bases, the author [9] also found set intersection and union operations to be of great importance.

This paper presents an analysis of unioning and intersecting of of ordered sets (hereafter referred to as "sets") formed from a restricted domain, and discusses the implications of the results of the analysis. Starting with the hypothesis that the use of context would enable improvements in the efficiency of an algorithm for forming set unions or intersections, a simple algorithm with the desired characteristics was proposed. When a set P is being intersected with a set Q, the term "context" refers to the fact that valuable information is available when an element of P is found to match an element of Q. The algorithm analysed in this study employs the contextual information to reduce the number of setps required to complete the intersection or union process. The results of the analysis in this paper are compared to earlier analyses of sorting and merging problems by Hwang and Lin [5], Hwang and Deutsch [4], Woodrum [10], and Knuth [6]. It is shown that minor alterations in the intersection or union algorithm, together with consideration of the nature of the set composition, can make significant differences in the expected amount of work required to complete the intersection or union task. The conclusion is that tailoring of the algorithms to given applications may enable significant improvements to be realized.

Note: Manuscript submitted February 23, 1977.

2. Statement of the Problem:

Although the algorithm given in Section three is applicable to both sets and lists, several differences are required between the analysis of sets and that of lists. Therefore, the approach taken below is to state and analyze the problem for the case of sets, and then to discuss the expected alterations if lists are treated.

Given a set P of size m and a set Q of size n, we wish to find the minimum, maximum, and expected cost of intersecting or unioning P with Q. The "cost" is measured as the number of pairwise symbol comparisons (one element of P against one element of Q) required to complete the task. Several assumptions are made for this analysis:

(1) The elements of P and Q are positive integers designated by $p_1, p_2, \ldots, p_m$ and $q_1, q_2, \ldots, q_n$ respectively.

(2) P and Q are ordered sets such that

$$p_1 < p_2 < \ldots < p_m$$

and

$$q_1 < q_2 < \ldots < q_n$$

(3) P and Q are composed of elements from a finite domain D of size d. The set P is formed (and similarly the set Q) by selecting m elements of D without replacement.

(4) D is an ordered set of positive integers designated as $a_1, a_2, \ldots, a_d$ such that

$$a_1 < a_2 < \ldots < a_d.$$

(5) It is equally likely for any element of D to appear in P (or in Q).

(6) $1 \leq m \leq d$; $1 \leq n \leq d$.

Note that replacing the reference in assumption (2) to "ordered sets" with "ordered lists", and changing assumption (3) to read "...selecting ... with replacement..." would enable us to also consider the more general intersection and merging problems with lists. Note also that the trivial case where P or Q is empty has been omitted from the analysis.

These assumptions enable us to consider even the most general type of sets with only a small amount of preprocessing. Finite sets composed of arbitrary length strings of characters can be mapped into a finite collection of positive integers by means of a dictionary. The ordering restriction requires only that each set be sorted prior to other processing, but all sets produced as a result of the intersection or

union tasks will be generated as ordered sets as a side effect of the processing algorithm.

Restricting the domain to a finite size should pose no real problem. At any given instant of consideration of most problems, only a finite number of distinct symbols are explicitly represented, even though the general domain may contain an infinite number of distinct symbols. Thus we need only require that no new symbols be added to the domain during the course of the process under consideration.

Assumption (5) is for the convenience of the analysis process. If it was not the case that each element of D was equally likely to be chosen as an element of P, or of Q, it would be necessary to include two probability density functions, one for P and one for Q, in the analysis. While this would make the analysis somewhat more general, the additional complexity would only tend to obscure the desired results. It is clear from the analysis shown in Section four that the results would be essentially unchanged by the use of selection functions with skewed distributions.

The problem as posed matches the conditions of many techniques for data management, question-answering, and problem-solving. For each of these environments, a large data base of information exists which is queried many times. Thus the sets of the data base need be sorted only once, and the cost of the preprocessing is negligible over the life of the data base. Each new question must also be preprocessed, but the cost of that preprocessing is small in general because the sets associated with each question are generally small.

3. An Intersection or Union Algorithm

The algorithm given below can be employed for intersection, union, or merging tasks simply by adding the appropriate output statements to steps (2), (3), and (4). For example, an intersection task would require no output for steps (2) or (4), and the output in step (3) of the matched element. Because the aspect of the algorithm of concern is the number of comparison steps, the output statements have been omitted below.

ALGORITHM (set intersection, set union, list merging)

[0] v ← 1; w ← 1

[1] [do a three way test]

IF (P(v)-Q(w))
{<0} GO TO [2]
{=0} GO TO [3]
{>0} GO TO [4]

```
[2]  [advance to next item of set (list) P]
     v ← v+1
     IF (v>m)  THEN TERMINATE ELSE GO TO [1]

[3]  [advance to next item of both sets (lists) due to a match]
     v ← v+1; w ← w+1
     IF (v>m OR w>n)  THEN TERMINATE ELSE GO TO [1]

[4]  [advance to next item of set (list) Q]
     w ← w+1
     IF  (w>n)  THEN TERMINATE ELSE GO TO [1]
END
```

The amount of work required to perform the complete intersection, union, or merge task can be measured in terms of the number of times step [1] of the algorithm is executed, as step [1] is the point at which pairwise symbol comparison is performed.

4. Analysis of the Algorithm

Careful examination of the algorithm shown above reveals that it can dispose of only one symbol per execution of step [1], except when the pair of symbols match. Thus the algorithm requires a:

(a) minimum of "min (m,n)" comparisons;

(b) maximum of m + n - 1 comparisons.

To determine the expected number of pairwise comparisons required, the following three part approach is used:

(1) Determine the total number of pairwise symbol comparisons required to process all possible pairs of sets of length m and n formed from the domain D, assuming that (a) only one symbol is processed even when a match is found; (b) only "w" is incremented in step [3].

(2) Determine the total number of comparisons saved, termed the correction term, by processing two symbols in one step when matches between symbol pairs are found.

(3) Determine the expected amount of work for sets of the given lengths by dividing the difference of the total number of comparisons and the correction term by the total number of possible pairs of sets of the given lengths which can be constructed from the domain.

The formula for the number of pairwise symbol comparisons of part (1) is expressed as two terms: (a) the work for those cases where the $p_m < q_n$; (b) and the work for those cases where $q_n \leq p_m$.

Each term is determined by computing the sum, over all possible pairs of sets, of the number of comparison steps required to process each case.

Starting with term (a) of part (1) only those cases where $p_m < q_j$ *for some* j, $1 \leq j \leq n$, are to be considered. These can be further subdivided into n groups, one for each of the n possible values of j such that either

$$p_m < q_1$$

or

$$q_{j-1} \leq p_m < q_j \quad \text{and} \quad 2 \leq j \leq n$$

because of the second assumption of the problem statement. This fact implies that for any fixed j, exactly m+j-1 comparisons will be required to handle each case of that group. Note that even when $p_m = q_{j-1}$ exactly m+j-1 comparison steps are required because of the assumption that only w is incremented in step [3] of the algorithm. Those cases where $p_m = p_n$ have been omitted because they are covered in term (b) of part (1). Due to the combination of assumptions two, three and four $p_m = d_i$ for some i, $m \leq i \leq d$. Thus, for any fixed i and j under these conditions there are exactly

$$\binom{i-1}{m-1}$$

distinct ways to form a set P of size m, and

$$\binom{i}{j-1} \qquad \binom{d-i}{n-j+1}$$

distinct ways to form a set Q of size n from the domain D. The total number of comparison steps required to handle all possible cases of a group defined by a given choice of i and j can be expressed as:

$$\binom{i-1}{m-1} \quad (m+j-1) \quad \binom{i}{j-1} \quad \binom{d-i}{n-j+1}$$

Adding the appropriate summations to cover all possible values of i and j provides an expression for term (a) of part (1) given as:

$$\sum_{i=m}^{d} \binom{i-1}{m-1} \sum_{j=1}^{n} (j + m - 1) \binom{i}{j-1} \binom{d-i}{n-j+1}$$

Several steps are required to reduce this expression to a simpler form, each of which requires some of the combinatorial relations given in Chapter II of Feller [3]. The first step uses the relation

$$\binom{x}{r+1} = \binom{x+1}{r+1} - \binom{x}{r}$$

to obtain the rewritten expression

$$\sum_{i=m}^{d} \binom{i-1}{m-1} \left\{ \left[\sum_{j=1}^{n} j \binom{i+1}{j} \binom{d-i}{n-j+1} \right] - \left[\sum_{j=1}^{n} j \binom{i}{j} \binom{d-i}{n-j+1} \right] \right.$$

$$\left. + (m-1) \left[\sum_{j=1}^{n} \binom{i}{j-1} \binom{d-i}{n-j+1} \right] \right\}$$

$$= \sum_{i=m}^{d} \binom{i-1}{m-1} \left\{ \left[\sum_{j=1}^{n} i \binom{i}{j-1} \binom{d-i}{n-j+1} \right] - \left[\sum_{j=1}^{n} i \binom{i-1}{j-1} \binom{d-i}{n-j+1} \right] \right.$$

$$\left. + m \left[\sum_{j=1}^{n} \binom{i}{j-1} \binom{d-i}{n-j+1} \right] \right\}$$

Next a further simplification can be obtained using the fact that

$$\sum_{v=0}^{r} \binom{a}{v} \binom{b}{r-v} = \binom{a+b}{r}$$

can be obtained. This results in the expression:

$$\sum_{i=m}^{d} \binom{i-1}{m-1} \left\{ i \left[\binom{d}{n} - \binom{i}{n} \right] - i \left[\binom{d-1}{n} - \binom{i-1}{n} \right] + m \left[\binom{d}{n} - \binom{i}{n} \right] \right\}$$

$$= \sum_{i=m}^{d} \binom{i-1}{m-1} \left[i \binom{d-1}{n-1} - (m+n) \binom{i}{n} + m \binom{d}{n} \right]$$

$$= \sum_{i=m}^{d} \left[m \binom{d-1}{n-1} \binom{i}{m} - (m+n) \binom{i}{n} \binom{i-1}{m-1} + m \binom{d}{n} \binom{i-1}{m-1} \right]$$

For the final reduction step we must employ the fact

$$\sum_{v=o}^{r} \binom{v+k-1}{k-1} = \binom{r+k}{k}$$

which enables us to obtain the reduced expression

$$m \binom{d-1}{n-1} \binom{d+1}{m+1} - \left[\sum_{i=m}^{d} (m+n) \binom{i}{n} \binom{i-1}{m-1} \right] + m \binom{d}{n} \binom{d}{m}$$

After rewriting this expression in a form more convenient for later manipulation, the simplified expression for term (a) of part (1) is as follows:

$$[F.1] \qquad \left[\frac{mn(d+1)}{d(m+1)} + m \right] \binom{d}{m} \binom{d}{n} - \sum_{i=m}^{d} (m+n) \binom{i-1}{m-1} \binom{i-1}{n}$$

Using an approach analogous to the derivation of term (a), the expression for term (b) of part (1) can be derived. However, due to the assumption that only pointer w to the set Q in the algorithm is incremented in step 3 , the groups are defined slightly differently. The m groups are defined by:

$$\text{of} \quad \begin{array}{l} q_n \leq p_1 \\ p_{j-1} < q_n \leq p_j \end{array} \qquad 2 \leq j \leq m$$

From this we can obtain that for any fixed i and j where $n \leq j \leq d$, exactly $n + j - 1$ comparisons are required, as pointer w will be incremented even when $q_n = p_j$. Thus the expression for term (b) is:

$$[F.2] \qquad \sum_{i=n}^{d} \binom{i-1}{n-1} \sum_{j=1}^{m} (j+n-1) \binom{i-1}{j-1} \binom{d-i+1}{m-j+1}$$

$$= \sum_{i=n}^{d} \binom{i-1}{n-1} \left[i \binom{d-1}{m-1} - (m+n) \binom{i-1}{m} - \binom{d-1}{m-1} + n \binom{d}{m} \right]$$

$$= n \binom{d+1}{n+1} \binom{d-1}{m-1} - \sum_{i=n}^{d} (m+n) \binom{i-1}{n-1} \binom{i-1}{m} - \binom{d}{n} \binom{d-1}{m-1}$$

$$+ n \binom{d}{n} \binom{d}{m}$$

$$= \left[\frac{nm(d+1}{d(n+1)} - \frac{m}{d} + n \right] \binom{d}{m} \binom{d}{m} - \sum_{i=n}^{d} (m+n) \binom{i-1}{n-1} \binom{i-1}{m}$$

The expression for part (2), the correction term, of the total work formula can be completed in one piece. Examination of the algorithm as given in section three reveals that the correction factor is simply the total number of times that execution of step [3] of the algorithm would have resulted in saving a comparison. This is the total number of times step [3] would be executed less the number of times step (3) would result in termination due to $w > n$ (which is the number of cases where $p_j = q_n$, $1 \le j \le m$, in term (a) part (1)). The expected number of pairwise symbol matches per case is $\frac{mn}{d}$, and the total number of cases is $\binom{d}{m} \binom{d}{n}$. Therefore the expression for part (2) is given as:

$$[F.3] \quad \frac{m\,n}{d} \binom{d}{m} \binom{d}{n} - \sum_{i=n}^{d} \binom{i-1}{n-1} \sum_{j=1}^{m} \binom{i-1}{j-1} \binom{d-i}{m-j}$$

$$= \frac{m\,n}{d} \binom{d}{m} \binom{d}{n} - \sum_{i=n}^{d} \binom{i-1}{n-1} \sum_{j=1}^{m} \left[\binom{i}{j} \binom{d-i}{m-j} - \binom{i-1}{j} \binom{d-i}{m-j} \right]$$

$$= \frac{m\,n}{d} \binom{d}{m} \binom{d}{n} - \sum_{i=n}^{d} \binom{i-1}{n-1} \binom{d-1}{m-1}$$

$$= \frac{m\,n}{d} \binom{d}{m} \binom{d}{n} - \binom{d-1}{m-1} \binom{d}{n}$$

$$= \left[\frac{mn}{d} - \frac{m}{d} \right] \binom{d}{m} \binom{d}{n}$$

The combined expression representing the total number of comparisons required over all possible cases is given as the sum of [F.1] + [F.2] - [F.3], from which the following simplified expression can be obtained:

$$[F.4]\quad \left[\frac{mn(d+1)}{d(m+1)} + m + \frac{mn(d+1)}{d(n+1)} + n - \frac{m}{d}\right]\binom{d}{m}\binom{d}{n}$$

$$- \sum_{i=m}^{d} (m+n)\binom{i-1}{m-1}\binom{i}{n} - \sum_{i=n}^{d} (m+n)\binom{i-1}{n-1}\binom{i-1}{m}$$

$$- \left[\frac{mn}{d} - \frac{m}{d}\right]\binom{d}{m}\binom{d}{n}$$

$$= \left[\frac{mn(d+1)}{d(m+1)} + \frac{mn(d+1)}{d(n+1)} + m + n - \frac{mn}{d}\right]\binom{d}{m}\binom{d}{n}$$

$$- \sum_{i=m}^{d} (m+n)\binom{i}{m}\binom{i}{n} + \sum_{i=n}^{d} (m+n)\binom{i-1}{n}\binom{i-1}{m}$$

$$= \frac{mn}{d}\left[\frac{d+1}{m+1} + \frac{d+1}{n+1} - 1\right]\binom{d}{m}\binom{d}{n}$$

To obtain the expression for the expected amount of work to intersect or union two sets of sizes m and n respectively, each composed of elements from a domain of size d, formula [F.4] needs only to be divided by the total number of set pairs possible for sets of size m and n taken from the domain of size d. This results in the expression:

$$[F.5]\quad \left(\frac{mn}{d}\,\frac{d+1}{m+1} + \frac{d+1}{n+1} - 1\right) = \frac{mn(m+n+2)}{(m+1)(n+1)} - \frac{mn(mn-1)}{d(m+1)(n+1)}$$

Before examining the comparison between this analysis and other related analyses, the effects of relaxing some of the assumptions will be examined.

5. Lists and Skewed Selection Functions

Formula [F.5] depends heavily upon two of the assumptions made in section two. These are:

(3) The set P (and similarly the set Q) is formed by selecting elements of D without replacement.

and (5) It is equally likely for any element of D to appear in P (or in Q).

As noted in section two, changing the selection function of assumption (3) to "selection with replacement" would enable more general lists to be considered, because elements in P or Q could be replicated. Full generalization would be obtained if P and Q were each formed using an independent selection function, each with its own probability distribution over the domain D.

These situations are significantly more complex than that modeled in section four, and they have not been examined in detail in this study. The key to the algorithm analysis is the determination of the expected intersection size. It can be seen from formula [F.5] that for any fixed m and n, the expected amount of work for the algorithm will vary inversely in proportion to the expected intersection size. However, it should be noted that skewed selection functions can also affect the number of comparisons required. For example, a lower expected number of comparisons would result if the selection functions for P and Q were skewed such that P favored elements from the first half of the domain and Q favored elements from the second half. A detailed analysis of these more general cases is required, but it appears that formula [F.7], which is given in the next section, represents a reasonable upper bound for even the most general cases of set and list intersection or merging.

6. Comparison to Previous Analyses

A number of analyses have appeared in the literature concerning the merging problem. In each such analysis the algorithm of concern was quite similar to the algorithm given in section three, but no advantage was made of the case where two set (or list) elements matched. The only assumptions other researchers used were that P and Q are disjoint sets such that

$$p_1 < p_2 <\ldots< p_m$$

and $$q_1 < q_2 <\ldots< q_n$$

Therefore the algorithms employed in those analyses did not need to handle the case of pairwise element matches. A typical algorithm analyzed under these assumptions, using the same notation employed in section three, would be as follows:

```
[0]  v ← 1, w ← 1
[1]  IF P_v < Q_w THEN GO TO [3]
[2]  w ← w + 1
     IF w > n THEN TERMINATE ELSE GO TO [1]
[3]  v ← v + 1
     IF v > m THEN TERMINATE ELSE GO TO [1]
```

Because of the assumption of disjoint sets, the equality case in step [1] never occurred. Woodrum [10], Hwang and Deutsch [4] (min and max only), Hwang and Lin [5], and Knuth [6] all report the analysis results as:

(a) The minimum number of comparisons is

$$\min(m,n)$$

(b) The maximum number of comparisons is

$$m + n - 1$$

(c) The expected number of comparisons is

[F.6] $$\frac{mn(m+n+2)}{(m+1)(n+1)}$$

Two cases which have received special attention are ($m=1$, $n>1$) and ($m=n$). H. Nagler [7] found that for the case $m=n$ the expected value is asymptotically $2n - 2 + 0(n^{-1})$. Hwang and Lin [5] pointed out that the case ($m=1$, $n>1$) is really an insertion problem for which a binary search requires the least possible number of comparisons.

It is interesting to note that all of the analyses cited not only fail to take advantage of pairwise matches in the algorithm, but also explicitly exclude from the analyses any cases in which $P \cap Q$ is non-empty. While the assumption of an empty intersection might be appropriate for some applications for merging, it clearly is not characteristic of the general case. In this paper the analysis has considered the more general case.

The analysis presented in section four includes as part (1) of the model the situation where no advantage is made of non-null intersections but permitting $P \cap Q$ to be non-empty. Thus the effect of relaxing the assumption that P and Q are disjoint can be seen directly from the expression formed as the sum of [F.1] and [F.2]. The resulting expression is:

$$\left[\frac{mn(d+1)}{d(m+1)} + m + \frac{mn(d+1)}{d(n+1)} + n - \frac{m}{d}\right] \binom{d}{m} \binom{d}{n}$$

$$= \sum_{i=m}^{d} (m+n) \binom{i-1}{m-1} \binom{i}{n} - \sum_{i=n}^{d} (m+n) \binom{i-1}{n-1} \binom{i-1}{m}$$

$$= \left[\frac{mn(d+1)}{d(m+1)} + \frac{mn(d+1)}{d(n+1)} - \frac{m}{d}\right] \binom{d}{m} \binom{d}{n}$$

$$= \left[\frac{mn(m+n+2)}{(m+1)(n+1)} + \frac{m(n^2+n-m-1)}{d(m+1)(n+1)}\right] \binom{d}{m}\binom{d}{n}$$

When this is divided by the number of cases to obtain the expected number of comparisons the result is:

[F.7] $$\frac{mn(m+n+2)}{(m+1)(n+1)} + \frac{m(n^2+n-m-1)}{d(m+1)(n+1)}$$

Expression [F.7] is shown in a form which enables a direct visual comparison to expression [F.6] developed by other authors. As might be expected, relaxing the assumption that P and Q are disjoint does result in a change in the expected number of comparisons. Several observations can be made:

(1) Because m, n, and d must all be greater than zero, [F.7] > [F.6] whenever

$$n > \frac{m+1}{n+1}$$

and

[F.7] < [F.6] whenever

$$n < \frac{m+1}{n+1}$$

Thus the expected number of comparisons may be significantly different from that predicted by [F.6] when the sizes of P and Q are disparate.

(2) Unlike [F.6], formula [F.7] is not symetric with respect to m and n. A simple way to determine the effect of being asymetric can be examined by comparing the general case of [F.7] to the case where n = m. The resulting inequality is:

$$\frac{m(n^2+n-m+1)}{d(m+1)(n+1)} < \frac{m(m^2+m-m-1)}{d(m+1)(m+1)}$$

$$n - m < \frac{m+1}{n+1} - 1$$

from which it is clear that the general case of [F.7] is less than the case where n = m only when n < m. Clearly also when n > m the reverse is true. The expected cost given by [F.7] of the intersection algorithm can thus be minimized by adding a test

in step [0] to interchange P and Q
if n>m.

(3) In expression [F.7] the $\frac{1}{d}$ multiplier of the second term is simply the probability of a pairwise match between an element of P and one of Q. Regardless of how this probability is obtained, as the probability of a pairwise match increases the potential difference between [F.6] and [F.7] increases. On the other hand, [F.6] is the limit for [F.7] as the probability of a pairwise match goes to zero.

It is clear from these observations that the results of the prior analyses leading to the expression [F.6] should not be blindly applied without a careful examination of the context of the problem. What may on the surface appear to be minor deviations from the real problem conditions may result in significant discrepancies in the analysis conclusions.

The effect of employing the contextual information about the expectation of non-null intersections and the size of the domain from which the sets are drawn can be clearly seen in the comparison between formula [F.5] and [F.7]. As the size of the domain (D) grows large relative to the size of the sets being processed, which corresponds to the probability of a pairwise match growing small, [F.5] reduces to [F.7]. But, as shown in Table 1, for many interesting cases the effect of the special treatment of the intersection is significant. The examples shown in Table 1 illustrate the anticipated result that the expected savings from the use of the intersection information increase as the expected size of the intersection increases. Thus, for situations where the intersection of the two sets being processed can be expected to be non-null in general, savings of 10% or more over the alternative algorithms might be realized.

7. Implementation Considerations

The algorithm of Section 3 is not intended as the ultimate intersection/union algorithm. For a general algorithm considerations such as those employed in the Hwang and Lin [5] "generalized" binary algorithm must also be employed. In that algorithm both a binary search and a linear search are employed, each being used under the appropriate conditions. One must also consider the nature of the machine being employed. Some machines have three way tests for less-than, equal, and greater-than. Others have only two way tests which must be combined judiciously to enable the desired efficiencies. In microcode based machines it may also be possible to create special test instructions to facilitate the intersection/union processing.

m	n	d	[F.7]	[F.5]	%reduction
5	5	100	8.367	8.167	2.4%
5	10	100	12.958	12.508	3.5%
10	5	100	12.908	12.508	3.1%
5	20	100	21.593	20.643	4.4%
20	5	100	21.443	20.643	3.7%
5	30	100	30.087	28.637	4.8%
30	5	100	29.837	28.637	4.0%
15	30	100	42.918	38.568	10.1%
30	15	100	42.768	38.568	9.8%
5	20	500	21.461	21.271	0.9%
20	5	500	21.431	21.271	0.8%
5	20	1000	21.445	21.350	0.4%
20	5	1000	21.430	21.350	0.4%
25	25	1000	48.100	47.500	1.3%
25	50	1000	72.634	71.410	1.7%
50	25	1000	72.610	71.410	1.7%
25	50	10000	72.591	72.469	0.2%
50	25	10000	72.589	72.469	0.2%

Table 1: Some Examples of the Savings Predicted by the Use of Context.

8. Conclusions

This paper has presented and analysed a simple algorithm for processing set intersections and set unions. It was shown that this algorithm can enable significant savings, in terms of the number of pairwise comparison steps, over algorithms previously analysed in the literature. The point of concern is not the algorithm per se, but the approach to the problem. The use of contextual conditions of the problem to be solved enabled a significant improvement to be realized in a very simple algorithm. By the use of modeling and analysis of the algorithm, realistic predictions about the algorithm behavior, and the significance of the modifications, could be clearly understood. The importance of using realistic assumptions in the analysis process was illustrated by a comparison of the results of this study to earlier analyses appearing in the literature.

REFERENCES

1. Fahlman, S., "The Intersection Problem," Working Paper 115, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, November 1975.

2. Feldman, J. A., Low, J. R., Swinehart, D. C., and Taylor, R. H., "Recent Developments in SAIL - An ALGOL Based Language for Artificial Intelligence," AFIPS, FJCC, Vol. 37 (Fall 1972) pp. 1193-1202.

3. Feller, W., An Introduction to Probability Theory and Its Applications, Vol. 1, John Wiley and Sons, New York, 1968.

4. Hwang, F. K., and Deutsch, D. N., "A Class of Merging Algorithms," Journal of the ACM, Vol. 20, No. 1 (January 1973,) pp. 148-159.

5. Hwang, F. K., Lin, S., "A Simple Algorithm for Merging Two Disjoint Linearly Ordered Sets," SIAM Journal of Computing, Vol. 1, No. 1, (March 1972), pp. 31-39.

6. Knuth, D. E., The Art of Computer Programming: Sorting and Searching, Vol. 3, Addison-Wesley Publishing Company, Reading, Massachusetts, 1973.

7. Nagler, H. "An Estimation of the Relative Efficiency of Two Internal Sorting Methods," Communications of the ACM, Vol. 3, No. 11, (November 1960), pp. 618-620.

8. Sussman, G. J. and McDermott, D. V., "From PLANNAR to CONNIVER - A Genetic Approach," AFIPS, FJCC 1972, pp. 1171-1179.

9. Wilson, G. A., "A Description and Analysis of the PAR Technique - An Approach to Parallel Search in Problem Solving Systems," Ph.D. Thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 1976.

10. Woodrum, L. J., "Internal Sorting with Minimal Comparing," IBM Systems Journal, Vol. 8, No. 3, (June 1969), pp. 189-203.

DEPARTMENT OF THE NAVY

NAVAL RESEARCH LABORATORY
Washington, D.C. 20375

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

POSTAGE AND FEES PAID
DEPARTMENT OF THE NAVY
DoD-316