UMECRL-TR-2-86

THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY

THE GENERAL CONSISTENT LABELING (OR CONSTRAINT SATISFACTION) PROBLEM

BERNARD NADEL

CRL-TR-2-86

PLEASE RETURN TO
COMPUTER SCIENCE DEPARTMENT ARCHIVES
3440 BOELTER HALL

1079 East Engineering Bldg.
Ann Arbor, MI 48109

THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY*

THE GENERAL CONSISTENT LABELING (OR CONSTRAINT SATISFACTION) PROBLEM

BERNARD NADEL

CRL-TR-2-86

January 1986

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

*Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agency.

THE GENERAL CONSISTENT LABELING (or CONSTRAINT SATISFACTION) PROBLEM[1]

Prof. Bernard A. Nadel[2]

Dept. Electrical Engineering and Computer Science

University of Michigan

Ann Arbor, MI 48109

January 1986

[1] This work was in part supported by the Computer Science Dept. of Rutgers University, New Brunswick, N.J., and in part by the Dept. Electrical Engineering and Computer Science of the University of Michigan, Ann Arbor, MI. This technical report therefore appears as part of the Technical Report series of both departments — DCS-TR-170 at Rutgers U. and CRL-TR-2-86 at U. Michigan.

[2] Previously: Bernard A. Nudel.

THE GENERAL CONSISTENT LABELING (or CONSTRAINT SATISFACTION) PROBLEM

ABSTRACT

A central Problem in Artificial Intelligence and Operations Research is the Consistent Labeling (or Constraint Satisfaction) Problem[3] CLP. This paper formulates CLP at a level of generality far beyond that usually treated, partitions it in various ways and introduces several probability models over the equivalence classes of these partitions. One partition of CLP is into **small-classes**, which have the very desirable ***homogeneity*** property that most instances within a given small-class have similar complexity of solution. This is the partition sought (but not found) by Gaschnig in his thesis [26]. Small-class homogeneity means that a small-class ***expected-case*** value can be used as a good approximation to the ***exact-case*** complexity of solving ***individual*** subsumed instances of the class. Our small-class expected complexity expressions thus provide for the first time a formal means of making problem-solving decisions (regarding which search-ordering, which algorithm and even which problem representation to use) that are good on an instance-by-instance basis. Various enumerative, combinatorial and probabilistic results are derived for the partitions and the probability models introduced over CLP. Our subsequent paper [52] uses these results to derive expected-case complexities for CLP problem-solving by three important algorithms, Backtracking, Forward-Checking and word-wise Forward Checking.

1. INTRODUCTION

This section informally introduces the general **Consistent Labeling Problem (CLP)**, perhaps better known as the Constraint Satisfaction Problem. Previous related work is discussed, in particular the contributions of Knuth [41], Gaschnig [26] and Haralick [36]. A major new theme of the present work is introduced: the value of carrying out a complexity analysis in terms of what we call *homogeneous* problem classes, so that a class expectation may be used as a good predictor of the complexity of solving *individual* subsumed instances of the class. In other words, an analysis with respect to homogeneous classes, allows *expected-case* values to be used as effectively *exact-case* values. Such an approach should be valuable for algorithmic complexity analysis in general; the present application to CLP should be taken as a case-study.

1.1. The Problem and its Algorithms

Despite the importance of the Consistent Labeling Problem in Artificial Intelligence and Operations Research, an adequate theoretical analysis of it and its algorithms has remained lacking. Such an analysis could lead to greatly improved problem-solving efficiency by providing a much-needed formal basis for decision-making during problem-solving. We have recently obtained expected-complexity expressions for three different CLP algorithms, and these have proven useful (see [51]) in guiding not only which algorithm to use, but also which search-ordering and even which problem-solving representation to use.

The term Consistent Labeling Problem or CLP is due to Haralick [34], [37] although it is perhaps best known as the Constraint Satisfaction Problem [19], [46]. It has been much studied and a good many other names also exist for CLP, among them the Satisficing Assignment Problem [26], Network Consistency Problem [46], [50], Relational Consistency Problem [49] and the Relation Synthesis Problem [23]. CLP is a slight generalization of Lieberherr's ψ_d-Satisfiability [45] which itself generalizes Schaeffer's S-Satisfiability [67], also referred to as Generalized Satisfiability in [24] since it in turn generalizes the well-known Satisfiability Problem of Cook [12]. The following additional terms may also be familiar but they describe more a solution method for CLP than the problem itself: Discrete Relaxation [65], Constraint Propagation [6], [70], Constraint Manipulation [19], Range Constriction [79], Waltz Filtering [78], [79], Backtracking [4], [29], [36] and Forward Checking [36].

[3] The word ***problem*** will be used in two senses in this work. The first sense refers to the usual kind of problem — for which a specific answer is sought to some question about some specific situation (or problem structure). The second sense refers to a *class* of problems of the first kind (usually obtained by allowing the problem structure to vary by use of a few parameters). To reduce ambiguity we will often use ***Problem*** (with an upper case P) when referring to such a class, and ***problem*** (with a lower case p), ***Problem instance*** or just ***instance*** when referring to a class member. (This convention is not always adhered to, but the intended sense will generally be clear from context.) Analogously we use CLP for the class of problems we are interested in here, while *clp* will refer to an individual member of CLP.

The problem as studied here however is much more general than the versions usually treated in the literature under the various aliases above — and this generality is carried through in all the complexity analyses of [51] and [52]. The main reason we have sought such generality is because of the ability of our *expected-case* values to approximate well *exact-case* values for individual instances. Since we can thus predict complexity for individual instances, we wish to take advantage of this ability to the full by having results that cover a problem class general enough to include all the various CLP instances that arise in real-world contexts. As described in [51], the results then become practical problem-solving tools, since they can guide decision-making on an instance-specific basis for arbitrary *realistic* instances arising in practice. A formal definition of CLP will appear in section 2. The following informal definition is convenient at this stage.

An instance of the Consistent Labeling Problem consists of

(1) a set of **variables,**

(2) a finite **domain** of candidate **values** associated with each variable — different variables possibly having different associated domains of values,

(3) a set of **constraints** on the values that various combinations of variables may compatibly take on, and

(4) a goal, which is to find all ways to assign to each variable a value from its associated domain in such a way that all constraint are simultaneously satisfied.

Consider the following example: Assign values to the three variables z_1, z_2 and z_3 from their respective value domains $\{0\ 1\}$, $\{0\ 1\}$ and $\{0\ 1\ 2\}$, in such a way that the following four constraints C_1 to C_4 are satisfied.

$$C_1: \quad z_1 \bigvee z_2 \tag{1}$$

$$C_2: \quad (z_3 - 3)^2 + (z_1 - 4)^2 \leq 25 \tag{2}$$

$$C_3: \quad \binom{z_3+1}{z_1} = z_1 + 2 \tag{3}$$

$$C_4: \quad \exp[\, z_1 + z_2 + z_3 \,] > 1.0 \tag{4}$$

There are two solutions, or **consistent labelings,** for this CLP instance. They are $\{z_1{=}1\ z_2{=}0\ z_3{=}2\ \}$ and $\{z_1{=}1\ z_2{=}1\ z_3{=}2\ \}$; that these are solutions can be readily verified by substituting the corresponding values for the variables into each of the four constraints above. This instance reflects many important aspects of the problem class studied here, and as such, will be used repeatedly below as a running example. For future reference we give it the name clp_0.

As in clp_0, we do not require that the variables of a CLP instance all have the same number of candidate values. Also, as with C_4 of clp_0, we do not restrict constraints to be binary (i.e. having only two argument variables). Neither need the constraints of an instance all have the same number of argument variables (compare C_1 and C_4) and there may be more than one constraint over the same set of arguments (C_2 and C_3). Some subsets of variables (such as $\{z_2\ z_3\}$) may have no constraint over them, and this will be explicitly modeled rather than using the usual unrealistic expedient of modeling such "missing constraints" by universal constraints.

The following problem due to Polya [63] suggests an interesting geometrical interpretation for CLP.

> Design a "multipurpose plug" that fits exactly into three different holes, circular, square and triangular. The diameter of the circle, the side of the square and the base and altitude of the isosceles triangle are of equal length, as shown in figure 1-1(a). The required multipurpose plug is to have the circle, square and triangle as its three orthogonal projections.

The most extensive solid satisfying the above requirements is shown in figure 1-1(b). CLP instances are similar to this multipurpose-plug problem in also being "volume-synthesis" problems subject to restrictions on the volume projections. However, in CLP we seek the maximal volume whose projections fall *inside* given regions rather than necessarily being *equal* to given regions as in the multipurpose plug problem. This view is formalized in Appendix A. Freuder in particular adopts this "volume construction" view, and accordingly has called CLP the **Relation Synthesis Problem** [23].

The generality of CLP as defined here, makes it relevant to such diverse applications as declarative language design, computer vision, VLSI design, theorem proving, relational database retrieval, graph problems including those of finding graph isomorphisms, graph colorings and cliques, as well as in a wide variety of puzzles. Such applications will be discussed further in section 1.3. A better understanding of CLP will therefore have considerable practical implications. Apart from this, CLP is important in providing an excellent context for addressing many of the questions central to problem-solving in general. These issues are discussed in the following section.

Besides the formalization, generalization and application of CLP per se, the development of CLP algorithms has also received intense attention within Artificial Intelligence, as exemplified by [4], [11], [23], [25], [29], [33], [34], [37], [46], [49], [50], [70], [78]. The Ph.D. thesis [26] of Gaschnig is especially important in this regard, as is Haralick's paper [36] which contains a nice empirical comparison of seven different CLP algorithm. Three of these algorithms, Backtracking, Forward Checking and bit-parallel Forward Checking are generalized and analyzed in [51] and in [52]. Simplified versions of this work have appeared in [56]-[58].

Fig. 1-1(a): The three holes for Polya's multipurpose plug.

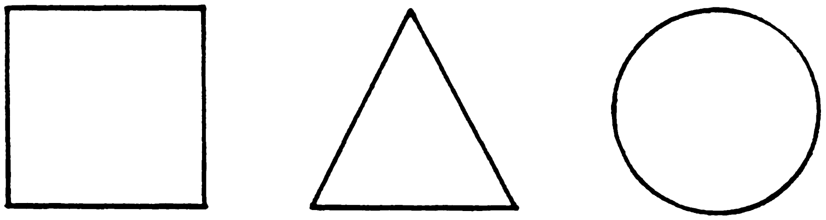
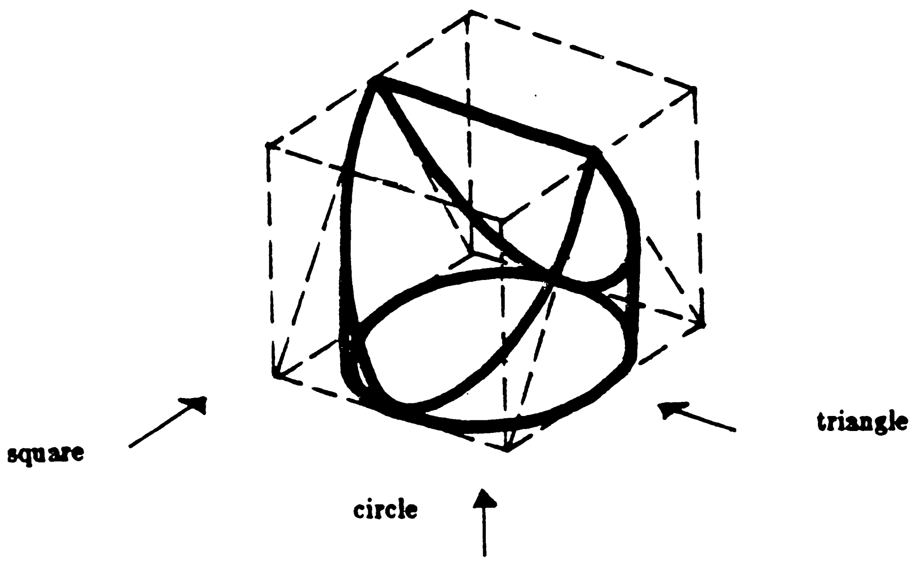


Fig 1-1(b): The maximal multipupose plug.



1.2. Related Work

Considerable work has been done on the Consistent Labeling Problem and its algorithms since the appearance of the seminal papers [77] and [29] in 1960 and 1965 respectively. This section presents some highlights, especially those directly relevant to the work presented here. The first attempt at an analytic understanding of the complexity of solving CLP instances was made by Knuth [41] in 1975 for a version of the standard Backtrack algorithm. He expresses some of our main concerns as follows

> One of the major difficulties associated with the so-called backtrack technique for combinatorial problems has been the inability to predict the efficiency of a given algorithm, or to compare the efficiency of different approaches without actually writing and running the programs.

Knuth's success in treating these issues was only partial however. Despite the above quote, values for the parameters of his expressions are still obtained only by actually coding and running the Backtrack algorithm — albeit for a much simpler Monte Carlo type computation on a given instance.

Gaschnig, in his dissertation [26], set himself the task of discovering empirically a set of problem parameters (i) whose values were more *accessible* than those used in Knuth's analysis and (ii) in terms of which an analysis of CLP would provide largely *instance-specific* predictions of complexity. These are commendable goals. Most complexity analyses — whether worst-case, expected-case or best-case — are relevant only to some artificial equivalence class and, for expected-case analyses, to some artificial probability model over the equivalence class (see section 6.3 of [24]). The results have precious little relevance in practice. Ideally what is needed are *exact-case* complexity results, giving in closed form the complexity of solving an arbitrary individual instance. Unfortunately, such results are in general "too good to be tractable". Gaschnig envisaged however, expected-case results which could be used as *effectively* instance-specific or exact-case — thus providing exact-case results "by the back door". In his own words, [26] page 242, this idea is expressed as follows

> . . . given a sufficiently fine grained partition, analytic or experimental average performance results for an ensemble of problem instances (i.e. an entire equivalence class) can be good predictors of performance of[4] an individual instance (i.e. an individual member of the equivalence class). An objective then is to define problem specification parameters yielding such a fine-grained partition of the problem class.[5]

The present work is also very much in the same spirit as Gaschnig's, and aims at cost-effective analytic expressions that can be used predictively for individual instances. However, Gaschnig emphasizes fineness of partition as the key to instance-specificity, but this is a little restrictive. What really counts is what we call **homogeneity.** A problem partition may be quite coarse and yet the expected-complexity of an algorithm over a given equivalence class may still be close to the complexity of solving each of its subsumed instances — so long as the complexity of individual instances in the class are nearly equal.[6] Such an equivalence class we call *homogeneous* (with respect to the algorithm and complexity measure of interest) — and of course, for such a class, the worst-case, expected-case or best-case value is a good estimate of the exact-case complexity of solving any individual subsumed instance. Fineness of partition per se, is not necessary, although it is true that refining a partition can only improve class homogeneity, and in the limit, where every instance is in its own equivalence class, all classes are in fact perfectly homogeneous. Nevertheless, a fine-grained partition is not in itself the goal, but rather a partition into homogeneous classes.

---

[4] This word "of" would be better written "on" instead.

[5] It is implicit here that in any complexity analysis, the "problem specification parameters" or more accurately, the problem parameters used for the analysis, induce a partition of the problem class into equivalence classes — where instances with the same set of values for the features are grouped together. The expression resulting from the analysis using those parameters refers to the generic such equivalence class; whether one obtains a worst-case, best-case or expected-case complexity expression, the value it gives is with respect to the generic class induced by the analysis parameters.

[6] Note that for a perfectly homogeneous class of problems, the worst-case, expected-case and best-case complexity of solving an instance of the class are all in fact equal — and any one of these three class-dependent complexities will be equally useful (and in fact fully accurate) as an estimate of the exact-case complexity of subsumed individual instances. Moreover, to the extent that the underlying class is homogeneous, the assumed probability model over the class will not effect the result of an expected-case analysis — an expected-case value must always lie between the worst-case and best-case values, and the latter two are equal for a truly homogeneous class. Note also that basing complexity analyses on a single parameter denoting "problem length" is virtually assured to produce results that are hopelessly inadequate in terms of instance-specificity. There is usually far too little information in such a parameter, to induce homogeneous equivalence classes. Analyses in terms of more natural parameters are becoming more common — see section 4.3 of [24] — but the criterion for choice of parameters is left open. The importance of aiming for homogeneous induced sub-classes has not yet been appreciated.

Gaschnig did not quite succeed in finding the set of parameters (a **homogenizing set**) required to induce a partition of CLP into homogeneous classes. The present work however has succeeded in this. The key problem parameter required is a measure of the looseness of a problem constraint that we call **constraint satisfiability.** It is discussed below particularly in sections 2.2 and 2.6. In earlier publications [56]-[58] treating more specialized forms of CLP, we referred to this parameter as inter-variable **compatibility.** Gaschnig got as far as partitioning CLP into what we call **middle-sized classes**, whereas the partition used here, based on constraint-satisfiability, is a refinment of Gaschnig's partition into what we call **small-classes.** The term **big-class** is reserved for the components of Haralick's partition [36] which is still courser than Gaschnig's. Of these three types of equivalence class of CLP instances, only small-classes exhibit the required degree of homogeneity to allow expected-case results to be effectively instance-specific. It should be noted that on page 241, near the end of his dissertation, Gaschnig does in fact propose a CLP partition in terms of small-classes. However, nothing more than this proposal appears concerning them.[7]

Gaschnig's first-mentioned concern above, that his parameters be "accessible" (as opposed to the case with Knuth's parameters, where extensive preprocessing was required to determine their values for a given instance) is also strongly shared in the present work. In general, there appears to be a three-way trade-off between the homogenizing power of a set of parameters, the accessibility of their values and the tractability of the resulting analysis. A reasonable degree of all three is needed, but increasing any two of them seems to require decreasing the third. In the present work on CLP they are all at more or less central values in their range: class homogeneity is good but not perfect, the analysis is of course tractable though certainly non-trivial, and accessibility is no problem for all parameters used in the analysis with the possible exception of the constraint satisfiability parameter as discussed in section 2.6.

In spite of the above-mentioned similarity of spirit between Gaschnig's work and ours, it should be noted that fundamental differences do exist. Gaschnig's work was essentially just an empirical exploration to determine what are the appropriate problem parameters to use. In fact he states (page 145, [26])

> Our objective then is to measure in case studies the values against which a future analysis could test its predictions, and to reveal patterns in the performances that may provide insight and direction by which to guide the development of such analysis.

The results presented here, on the other hand, provide thoroughly analytic, closed-form expressions for predicting the complexity of solving CLP instances. Moreover, our analysis parameters are an extension of those used in Gaschnig's experiments and, as mentioned, they induce the effectively homogeneous classes that he sought, while remaining reasonably accessible (unlike the parameters required by Knuth). It would seem then that the present work attains the goals for which Knuth and Gaschnig were aiming.

An important link between the present work and the earlier work of Knuth, Gaschnig and others, has been the recent contribution of Haralick. In [36] he empirically compares seven different CLP algorithms. Two of these algorithms, Backtracking and Forward Checking, he also compares theoretically. In fact, Haralick's work provides the first analytic closed-form expressions for the complexity of solving Consistent Labeling Problems. However, he treats only a very specialized form of CLP defined here. Similarly, the probability model used by Haralick is quite restrictive. And most importantly, Haralick uses for his analysis a set of parameters capable of inducing only a relatively coarse partition of CLP into the above-mentioned big-classes, which are even less homogeneous than Gaschnig's middle-sized classes. Haralick's results are therefore not useful in providing instance-specific predictions of problem-solving complexity — although by definition, they of course provide perfectly valid[8] expected-case results for big-classes under the probability model he assumes.

Haralick's work of [36] is also important for having introduced the notion of extracting heuristic advice from an analytic complexity expression — which has also been a major concern of our work (as described in [57], in [51] and in anticipated future papers). Since Haralick's results are not effectively instance specific (due to the inhomogeneity of his underlying big-classes) neither are his theory-based heuristics. Since our results are effectively instance-specific (due to homogeneity of the underlying small-classes) so are the theory-based heuristics derived from them. Nevertheless, we will see that big-class expectations are useful, even for obtaining approximations to exact-case complexities. This is so because (as discussed in section 3.5.1) using a probability model slightly more general than Haralick's, big-class

[7] This proposal of Gaschnig's was noticed after we had in fact independently discovered the usefulness of a small-class partition and had already carried out the analysis in terms of the corresponding parameters.

[8] Except for a certain error in Haralick's results, pointed out in [52].

expectations can be used to approximate — with less computational effort — the small-class expectations which in turn provide good approximations to exact-case values.

Note that the above-mentioned two themes of

(i) performing complexity analyses with respect to a partition into homogeneous classes, so that class-expectations can be used as effectively exact-case values, and

(ii) deriving heuristics via minimization of analytic complexity expressions

are not restricted to the context of the Consistent Labeling Problem. They are potentially useful in general for algorithm analysis and algorithm design respectively. Our work on CLP should be considered a case-study in this regard.

1.3. Applications of CLP

The Consistent Labeling Problem provides a natural framework for representing a remarkably wide range of real-world problems. It is relevant in such diverse areas as **Artificial Intelligence**, **Operations Research**, **Combinatorics**, **Theorem Proving**, **Machine Vision**, **VLSI design** and **Relational Database Retrieval**. (Chapter 7 of [51] discusses such applications in more detail.) In fact the CLP formalism is so generally applicable that it provides an attractive basis for the design of **declarative programming languages** as seen in [5], [19], [27], [44], [71]-[74].[9]

One of the earliest and most natural applications of CLP is in solving various puzzles such as n-queens. The latter is the problem of placing n queens on an $n \times n$ chess board in such a way that no two queens attack each other. It was first proposed in 1850 by Frank Nauck and has since become the classic domain for introducing and studying CLP. A complete early history of it and related problems can be found in [1]. More recently, the n-queens problem has appeared in many studies including [19], [21], [26], [36]. Many other puzzles are naturally formulated as CLP instances including **word-sum** or **cryptarithmetic** puzzles [11], [19], [53], **Instant Insanity** [10], and the **Soma Cube** puzzle [20]. Also, the book by Reingold et. al. [64] includes amongst the exercises applications to **Pentominoes**, **Magic Squares**, **Latin Squares** and other puzzles.

Puzzles such as the above posses a certain regularity of structure that makes them relatively easy to describe and understand. This has no doubt been a major reason for their centrality in the study of CLP. However, the appeal of such regularity has inhibited the study of more general forms of the Consistent Labeling Problem. In particular, the **binary** form, where every constraint involves only two variables, has received a disproportionate amount of attention [26], [36], [46], [50]. Note that the common view of CLP as a certain **Network Consistency Problem** arises from this emphasis on the binary-constraints case, since such constraints may be viewed as links between pairs of variables, as described by Montanari [50]. Another restriction usually imposed in systematic studies is that all variables have the same domain of candidate values. These and many other common restrictions are removed in the version of CLP to be treated here — allowing the natural formulation of a far broader class of real-world problems than would otherwise be possible.

However, historically, binary CLP *has* been adequate to model many interesting problems — although sometimes in rather unnatural ways. This is not surprising given the existence of a general transformation (see [51]) for reducing a non-binary CLP instance to an equivalent binary instance. The extension to the general form of CLP treated here is therefore not made because it increases "representational completeness", but rather because it often affords more natural representations and more importantly because it allows representations under which problem-solving may be accomplished more efficiently. Chapter 8 of [51] shows that our analytic results are capable of providing useful guidance in choosing between such alternative representations.

Applications for which the binary form of CLP has been employed quite naturally include many important problems such as the finding of **Subgraph and Graph Isomorphisms** [49], finding **Hamiltonian Circuits** in a graph [54], **Graph Coloring** [54] and the **Bin Packing Problem** of [16] which Haralick points out in [34] is isomorphic to the problem of graph coloring. Also in [34], Haralick uses binary CLP in formulating **Boolean Satisfiability** — the problem of finding a satisfying truth assignment for a conjunctive normal form expression in the propositional calculus. Haralick's formulation of Satisfiability as binary CLP is however not as natural as the non-binary CLP version implicit in the work

[9] In this context, the heuristics derived in the present work become contributions to making such languages more practical by allowing efficient autonomous control (to use Kowalski's expression [43]) by the language interpreter.

of Brown and Purdom [7]-[9].[10] These alternative formulations (further discussed in chapter 7 of [51]) illustrate the above-mentioned need to be able to choose between competing CLP representations of a problem.

Machine vision is another important domain of application for CLP. Waltz employed CLP to formulate the problem of interpretating lines in a drawing of a block's world scene [78]. Again, the binary form of CLP was found to be applicable to this **line labeling** problem, but not altogether naturally. As with Haralick's formulation of Boolean Satisfiability mentioned above, a more natural formulation of the line labeling problem requires a generalization beyond binary CLP. (These alternatives for the line-labeling problem are discussed in chapter 7 of [51] and in [65].) Mackworth also used a CLP formulation for a related line-labeling and region-labeling problem that arises in interpreting maps [47].

At a lower level than the line labeling problem in machine vision is the **line detection** problem, which has also been formulated by Haralick [34], in terms of binary CLP. Roughly speaking, it is the solution of this problem that is the input of the line labeling problem mentioned above. In the same way, the solution of the line labeling problem may be considered the input to the yet higher level problem of **scene labeling** or **object interpretation**, where one seeks to assign a meaning as an object to the various segmented regions of the scene. This also has been formulated in terms of CLP in [2], [35] and [30]. The latter paper is particularly relevant to the present work, as it applies results from our earlier paper [57].

CLP also arises naturally in relational database retrieval where it corresponds to the problem of finding the **join** of several relations, as explained in chapter 7 of [51]. CLP can also be applied to **consistency maintenance**, **query-answering** and **redundancy-checking** in databases as in [31].

Recently many extensions of the basic CLP formalism have been proposed as useful in various areas, especially for machine vision. These include what might be called **hierarchical CLP** [13], [15], **probabilistic CLP** [17], [62], [65], **fuzzy CLP** [65], **inexact** or **weighted CLP** [68] and **optimizing CLP** [17], [76]. Many applications of these extensions of CLP to machine vision are summarized in [14]. Other applications have included **code-breaking** [61], **handwriting interpretation** [39], [60] and **shape segmentation** [66]. An important application of an extended form of CLP is in Kowalski's **connection graph** method for **theorem proving** in the predicate calculus [42]. As mentioned in [42], this method was inspired by the Waltz filtering algorithm developed for solving CLP instances (arising originally from the problem described above of labeling lines in scenes).

1.4. Notation

Table 1-1 introduces some general notation that will be useful. Given a set Z of size n, note that the notation $\binom{Z}{m}$ for the *set* of all size-m subsets of Z, is a natural extension of the notation $\binom{n}{m}$ for the number of such subsets — in the same way that the notation 2^Z for the set of all subsets of Z is obtained from 2^n, the number of subsets of Z.

2. THE CONSISTENT LABELING PROBLEM (CLP)

This section formally defines the Consistent Labeling Problem in a manner far more general than has heretofore been systematiclly studied — whether theoretically or empirically. Problem parameters are introduced that will be important in our later analysis of the complexity of solving CLP instances. In particular, **constraint satisfiability** is introduced as a parameter characterizing the looseness of a constraint. As mentioned above, this is the key problem feature that allows us to partition CLP into classes — called **small-classes** — which are sufficiently homogeneous that class expectations are virtually instance-specific, or exact-case, for most subsumed instances of the class. Section 2.7 discusses a spectrum of approaches for obtaining constraint satisfiability values in practice. Examples of the concepts introduced are given using the 4-queens problem and the instance clp_0 presented in section 1.1.

2.1. CLP Instances

A CLP instance consists of variables, their associated domains of values and a set of constraints. The goal (for the variant we are interested in) is to find *all* ways to assign variables values from their respective domains so as to simultaneously satisfy all constraints. For problem-solving purposes these

[10] Analytic studies of the complexity of solving Satisfiability have recently appeared in [22], [28]. These however treat the problem of finding a single satisfying truth assignment, whereas when our results are specialized to the context of Satisfiability, they give the complexity of finding *all* satisfying assignments.

Table 1-1: Some useful notation.

Symbol	Defined as	
J_m^n	$\{ m\ m{+}1 \ldots n \}$	The set of integers from m to n inclusive
$\biguplus_{j \in I} S_j$		The union of mutually *disjoint* sets S_j
$\bigcap^{\bullet}_{j \in I} E_j$		The intersection of mutually *independent* events E_j
$\binom{n}{m}$		The *number* of combinations of n objects taken m at a time
$\binom{Z}{m}$		The *set* of all size-m subsets of a set Z
2^Z	$\biguplus_{j=0}^{\lvert Z \rvert} \binom{Z}{j}$	The set of all subsets of Z, or the power set of Z

components of a CLP instance *clp* may take on a variety of forms. Variables may have arbitrary names, and their domains may contain arbitrary values of any type: integer, real, boolean, alphanumeric symbols, or compositions of any of these into sets, tuples, or what-have-you. Constraints may be given as tables or as expressions in arbitrary constraint-specification languages — algebraic, logical or whatever. However, for analysis purposes it suffices to model instances of CLP using the simple canonical form described below. This is so because for the algorithms and their complexity measures of interest here, all instances that have the same canonical form also have the same complexity of solution.

A canonical-form CLP instance $clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$ is a five-tuple $(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$[11], where:

- n is the number of problem variables. These variables we denote z_i, and the set of all problem variables is $Z = \{ z_1\ z_2 \ldots z_n \}$.
- $\mathbf{m} = (m_{z_1}\ m_{z_2} \ldots m_{z_n})$ is the vector of domain sizes m_{z_i}, where m_{z_i} is the number of candidate values in the domain d_{z_i} for variable z_i. The domain itself is denoted $d_{z_i} = \{ z_{i1}\ z_{i2} \ldots z_{im_{z_i}} \}$, z_{ij} being the j-th candidate value for z_i. We will also use $\bar{z}_i$ to denote an arbitrary value of z_i from d_{z_i}, and $\mathbf{d} = (d_{z_1}\ d_{z_2} \ldots d_{z_n})$ to denote the vector of all n domains.
- c is the number of constraints.
- A (not necessarily canonical) constraint C_j is some way of specifying which tuples of values for a certain argument set $Z_j \subseteq Z$ of variables are mutually compatible. The algorithms discussed later may have such constraints specified extensively as tables, or intensively as subprograms. But all that counts for the complexity of solving a CLP instance is the canonical form $C_j = (Z_j\ T_j)$ for each constraint C_j. The first component, $Z_j \subseteq Z$, is the set of argument variables which C_j constrains. The second component T_j is the relation induced by constraint C_j; that is, T_j is the set of value tuples that satisfies C_j, where values are of course chosen for each argument variable z_i from its respective domain d_{z_i}. Thus, denoting by $D_j = \times_{z_i \in Z_j} d_{z_i}$ the cartesian product of the domains of the argument variables of constraint C_j, we have that $T_j \subseteq D_j$ consists of all value tuples $\bar{Z}_j \in D_j$ that satisfy C_j.
- The two parameters $\mathbf{Z}$ and $\mathbf{T}$ of the generic CLP instance $clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$ are the vectors of constraint argument sets Z_j and of constraint relations T_j respectively

$$\mathbf{Z} = (Z_1\ Z_2 \ldots Z_c) \text{ and } \mathbf{T} = (T_1\ T_2 \ldots T_c)$$

[11] Note that parameters n and c are actually redundant since $n = \lvert \mathbf{m} \rvert$ and $c = \lvert \mathbf{Z} \rvert = \lvert \mathbf{T} \rvert$. They are however retained for clarity.

A **consistent labeling** is an assignment of values to all n problem variables which satisfies all c constraints. The goal in solving a CLP instance is to find all consistent labelings. That is, the goal is to find the set

$$T = \{ \bar{Z} \mid \bar{Z} \in D \text{ and } \bar{Z}_j(\bar{Z}) \in T_j \quad 1 \le j \le c \} \tag{5}$$

where $D = \bigtimes_{z_i \in Z} d_{z_i}$ is the cartesian product of the domains of all problem variables, and $\bar{Z}_j(\bar{Z})$ denotes the projection of the value-tuple $\bar{Z}$ onto the argument set Z_j.[12] Several alternative characterizations of this solution set T are given in appendix A, including one in terms of the "join" operator $\bowtie$ of relational database theory [48].

2.2. Constraint Satisfiability and Other Parameters

- We use $M_j = \mid D_j \mid$ to denote the size of the cartesian product of (the domains of the argument variables of) the j-th constraint. Of course, from the definition of D_j we have that $M_j = \prod_{z_i \in Z_j} m_{z_i}$.
- We use $S_j = \mid T_j \mid$ to denote the size of the relation T_j induced by the j-th constraint. This is called the **satisfiability** or **looseness** of constraint C_j[13] since, by definition of T_j, it gives the number of value-tuples from D_j that satisfy C_j. The vector of satisfiabilities for the c problem constraints is denoted $\mathbf{S} = (S_1\ S_2\ ..\ S_c)$.

  Constraint satisfiability is a key parameter in the present work, as it allows us to analyze CLP in terms of classes (called **small-classes**) that are largely homogeneous. As a result, expected-case complexities for a small-class are good estimates for the complexities of individual instances in the class, and may thus provide virtually instance-specific information.
- The sum $SS = \sum_{1 \le j \le c} S_j$ of the satisfiabilities for the c constraints of an instance is called the **summed satisfiability**.
- The ratio $R_j = \mid T_j \mid / \mid D_j \mid = S_j / M_j$ is the **satisfiability ratio** of constraint C_j. Note that for any CLP instance we have that

$$\emptyset \subseteq T_j \subseteq D_j \qquad 0 \le S_j \le M_j \qquad 0 \le R_j \le 1 \qquad 0 \le SS \le \sum_{j=1}^{c} M_j \tag{6}$$

- The number of argument variables for constraint C_j is $A_j = \mid Z_j \mid$, called the **arity** of constraint C_j. Constraints with $A_j = 2$ and $A_j = 3$ arguments are called respectively **binary** and **ternary** constraints. Most CLP work in Artificial Intelligence has treated instances all of whose constraints are binary.

2.3. Two Example CLP Instances

In this section the above definitions are made more concrete by means of two example CLP instances: a CLP formulation of the 4-queens problem and the example instance clp_0 introduced in section 1.1. The latter instance is particularly important here, as it will be used as a running example throughout the remainder of this work.

Example 1: The n-queens problem was introduced in section 1.3 as the problem of placing n queens on an $n \times n$ chess board so that no two queens attack each other. We henceforth rename this the q-queens problem, using q queens and a $q \times q$ board, to avoid confusion with the parameter n of our generic CLP instance. The set of problem instances arising as q varies we call QUEENS, so that q-queens is the generic instance of the class of instances QUEENS. This class has been much studied in relation to CLP, but it should be emphasized that as presented above, q-queens is not yet a CLP instance. There are in fact several possible ways to map q-queens into CLP (as discussed in chapter 7 of [51]). The most common of these mappings is used below. Note that this availability of several alternate representations within CLP for a given problem raises the issue of which representation is best. Chapter 8 of [51] shows that our

[12] For example, if $\bar{Z} = (g\ a\ 37\ e\ 12)$ for the variables of $Z = \{z_1\ z_2\ z_3\ z_4\ z_5\}$ and if $Z_j = \{z_2\ z_4\ z_5\}$ then the projection of $\bar{Z}$ onto Z_j is $\bar{Z}_j(\bar{Z}) = (a\ e\ 12)$. We will use the convention that in tuples $\bar{Z}_j$, values $\bar{z}_i$ are arranged left to right so that their associated variables are in order of increasing index.

[13] In [56] and [57] it was called the compatibility of the argument variables of C_j. Volume or simply size of constraint C_j would also be appropriate names for parameter S_j.

analytic complexity results can also be used in making such representational choices.

In formulating q-queens as a CLP instance we first note that there can be no *more* than one queen per row of the board, since any multiple queens in a row would attack each other, thus violating the constraints. Since there must be at most one queen per row, there also cannot be *less* than one queen in any row, else there could not be a total of q queens on the board. Thus if a solution exists for q-queens it must correspond to exactly one queen per row. It is thus appropriate to formulate q-queens as a CLP instance by associating a CLP variable z_i with each board row $1 \le i \le q$, and allowing variable z_i to take as its value the column in which the corresponding queen is to be placed. There are then $n = q$ CLP variables and $Z = \{z_1\ z_2\ ..\ z_q\}$. Identifying columns by integers 1 to q, the domain of a' priori allowed values for each CLP variable is then $d_{z_i} = \{1\ 2\ ..\ q\}$, the same for each variable z_i. Domain sizes are therefore also the same, $m_{z_i} = q$, for each variable.

To avoid two queens attacking each other, the CLP constraints must ensure that no two queens are in the same row, column or diagonal of the board. That they are not in the same row is already ensured by the interpretation given to variables z_i. The remaining requirement that no two queens are in the same column or diagonal is captured mathematically as follows

$$C_{z_i z_j}:\quad (z_i \ne z_j) \wedge (\ |z_i - z_j| \ne j - i\ ) \qquad \forall\ \ i < j,\ \text{where}\ i, j \in \{\ 1\ 2\ ..\ q\ \}. \tag{7}$$

This by the way assumes the usual consecutive numbering of rows from top to bottom of the board, and consecutive numbering of columns from left to right, as in figure 2-1. The first conjunct ensures that the corresponding two queens do not occupy the same column on the chess board, and the second conjunct ensures that they are not in the same right- or left-diagonal.[14] There are $c = \binom{q}{2}$ separate constraints indicated in (7), one for each pair of the q variables. Each such pair of variables constitutes the argument set $Z_{z_i z_j} = \{z_i\ \ z_j\}$ of the corresponding constraint $C_{z_i z_j}$. Note that since there is no more than one constraint having the same argument set, we are in this case able to index constraints by their set of argument variables rather than using less descriptive, though more general, single integer subscripts. Because the domains d_{z_i} are all the same here, the domains $D_{z_i z_j}$ for compound variables $Z_{z_i z_j}$ are also all the same. Specifically, each $D_{z_i z_j}$ is the set $D_{z_i z_j} = d_{z_i} \times d_{z_j}$ of all ordered pairs of integers in $\{1\ 2\ ..\ q\}$. The sizes of $D_{z_i z_j}$ are also all the same, being $M_{z_i z_j} = |\ D_{z_i z_j}\ | = q^2$, while the size of the cartesian product of all $n = q$ domains is $M = |D| = q^q$. Since each constraint is binary, we have that $|\ A_{z_i z_j}\ | = 2$ for all c constraints.

Specializing the above formulation to the case of the 4-queens problem, we have $n = 4$ CLP variables, each with domains $d_{z_i} = \{1\ 2\ 3\ 4\}$ of size $m_{z_i} = 4$. There are $c = \binom{4}{2} = 6$ constraints of the form in (7). The domains of the argument sets Z_j are all

$$D_j = \{\ (11)(12)(13)(14)(21)(22)(23)(24)(31)(32)(33)(34)(41)(42)(43)(44)\ \}$$

with $M_j = |\ D_j\ | = 16$ for all constraints.[15] While the sets D_j and the values M_j are the same for all constraints, the actual constraint relations $T_j \subseteq D_j$ induced by constraint C_j are not all the same, since the particular pairs of column positions that are compatible for two queens depends on their respective rows. (In fact it depends only on the *separation* of these two rows). The $c = 6$ relations T_j for 4-queens are shown in table 2-1 together with the corresponding argument sets Z_j so as to constitute the canonical-form counterparts $C_j = (Z_j\ \ T_j)$ of the analytic constraints in (7). The T_j of table 2-1 are of course such as to include from D_j all (and only) those pairs of values that place the corresponding queens in different columns and in different diagonals.

The sizes of the relations T_j in table 2-1 are the constraint satisfiabilities $S_j = |\ T_j\ |$, so that for 4-queens the satisfiabilities are $S_j = 6, 8, 10, 6, 8$ and 6. We will see in section 2.6.1 that these values can be determined analytically for all constraints of any QUEENS instance. Dividing the six 4-queens satisfiabilities by the corresponding M_j value, which in this case is 16 for all constraints, we obtain the satisfiability ratios $R_j = S_j / M_j$ shown in table 2-1. Summing the six satisfiabilities for 4-queens gives the summed-satisfiability $SS = 44$. The solution set for 4-queens is $T = \{\ (2413)\ (3142)\ \}$, so that the number of solutions or consistent labelings is $S = |\ T\ | = 2$. Figure 2-1 shows both these solutions as a

[14] A right diagonal is one that falls as it is traversed from left to right. A left diagonal rises from left to right.

[15] We now revert to the usual method of constraint indexing using single integers rather than using the constraint's set of ar-

Table 2-1: Some parameter values for the $c = \binom{4}{2} = 6$ constraints C_j of the 4-queens problem.

j	Z_j	T_j	A_j	M_j	S_j	R_j
1	$\{z_1\, z_2\}$	{ (13) (14) (24) (31) (41) (42) }	2	16	6	6/16
2	$\{z_1\, z_3\}$	{ (12) (14) (21) (23) (32) (34) (41) (43) }	2	16	8	8/16
3	$\{z_1\, z_4\}$	{ (12) (13) (21) (23) (24) (31) (32) (34) (42) (43) }	2	16	10	10/16
4	$\{z_2\, z_3\}$	{ (13) (14) (24) (31) (41) (42) }	2	16	6	6/16
5	$\{z_2\, z_4\}$	{ (12) (14) (21) (23) (32) (34) (41) (43) }	2	16	8	8/16
6	$\{z_3\, z_4\}$	{ (13) (14) (24) (31) (41) (42) }	2	16	6	6/16

Fig. 2-1: The two solutions of the 4-queens problem, and their projections onto the various constraint argument sets.

	1	2	3	4
z_1		Q		
z_2				Q
z_3	Q			
z_4			Q	

	1	2	3	4
z_1			Q	
z_2	Q			
z_3				Q
z_4		Q		

$\bar{Z}$:	(2 4 1 3)	(3 1 4 2)
$\bar{Z}_1(\bar{Z})$:	(2 4)	(3 1)
$\bar{Z}_2(\bar{Z})$:	(2 1)	(3 4)
$\bar{Z}_3(\bar{Z})$:	(2 3)	(3 2)
$\bar{Z}_4(\bar{Z})$:	(4 1)	(1 4)
$\bar{Z}_5(\bar{Z})$:	(4 3)	(1 2)
$\bar{Z}_6(\bar{Z})$:	(1 3)	(4 2)

gument variables. Any convenient linear ordering of the constraints is appropriate.

board configuration and as $\overline{Z}$ the corresponding consistent-labeling of Z, and gives for each consistent-labeling its projection $\overline{Z}_j(\overline{Z})$ onto the argument sets Z_j of the six problem constraints.

Example 2: Even though there are an infinite number of QUEENS instances — one for each integer q — the corresponding CLP instances are all of a quite specialized form. Amongst other things, all such instances have

(1) only binary constraints

(2) exactly one constraint (no more, and no less) for each possible pair of variables,

(3) all domain sizes equal to the number of variables.

These simplifications have proven seductive and at least some of them are imposed in virtually all systematic CLP studies to date [26], [36], [46], [49], [50]. In other words, most studies have treated only q-queens-like instances — with the main concession to generality being only to allow the relations T_j to differ from those of q-queens. As a result, precious little systematic knowledge exists about the more general kinds of CLP instances needed to naturally model many real-world problems.

None of the above QUEENS-type restrictions are imposed here — and this relaxation to a much more general class of instances than is usually treated is one of the main contributions of the present work. Instance clp_0, introduced in section 1.1, has in fact been chosen in order to reflect this generality. For convenience it is described again here, before proceeding to a more detailed discussion. Instance clp_0 involves three variables z_1, z_2 and z_3 taking values from domains $d_{z_1} = \{\ 0\ 1\ \}$, $d_{z_2} = \{\ 0\ 1\ \}$ and $d_{z_3} = \{\ 0\ 1\ 2\ \}$ respectively. We seek all assignments of values to the three variables so as to simultaneously satisfy the following four constraints:

$$C_1:\quad z_1 \vee z_2 \tag{8}$$

$$C_2:\quad (\ z_3 - 3\ )^2 + (\ z_1 - 4\ )^2 \le 25 \tag{9}$$

$$C_3:\quad \binom{z_3+1}{z_1} = z_1 + 2 \tag{10}$$

$$C_4:\quad \exp[\ z_1 + z_2 + z_3\ ] > 1.0 \tag{11}$$

We see that clp_0 has $n = 3$ variables z_i, so that the variables set is $Z = \{z_1\ z_2\ z_3\}$. The sizes of the domains for the three variables of clp_0 are $m_{z_1} = 2$, $m_{z_2} = 2$ and $m_{z_3} = 3$. Note that these domain sizes are not all equal, and they are certainly not all equal to the number of variables (in this case 3). In both respects this differs from the situation for QUEENS instances (as formulated above). Each QUEENS instance has all its domains of equal size, and moreover, this size is also equal to the number of variables for that instance.

Furthermore, whereas q-queens instances involve one binary constraint for each pair of variables, this is not the case in clp_0. The argument sets of the clp_0 constraints are $Z_1 = \{\ z_1\ z_2\ \}$, $Z_2 = Z_3 = \{\ z_1\ z_3\ \}$ and $Z_4 = \{\ z_1\ z_2\ z_3\ \}$. The corresponding arities or numbers of arguments for the constraints are therefore $A_1 = A_2 = A_3 = 2$ and $A_4 = 3$ so that the constraints are not all binary, nor are their arities even all the same. Moreover, not all pairs of variables correspond to exactly one constraint — $\{z_2\ z_3\}$ is the argument set of *no* constraint and $\{z_1\ z_3\}$ is the argument set of *two* different constraints, C_2 and C_3.

Also unlike QUEENS instances, the cartesian products D_j for clp_0 constraints are not all the same since their factors d_{z_i} are not all the same. We have for clp_0 that

Table 2-2: CLP Instance clp_0 — our running example.

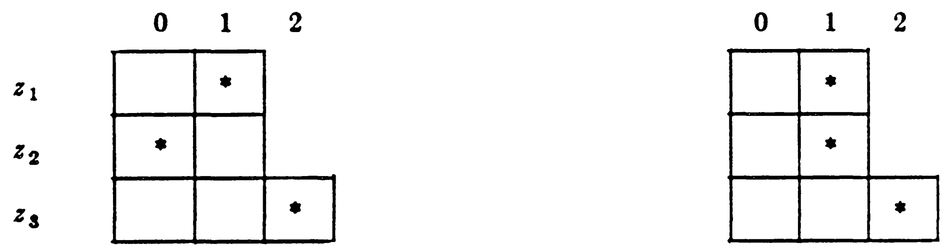
Variables: $n = 3$

i	z_i	d_{z_i}	m_{z_i}
1	z_1	{ 0 1 }	2
2	z_2	{ 0 1 }	2
3	z_3	{ 0 1 2 }	3

Constraints: $c = 4$

j	Z_j	T_j	A_j	M_j	S_j	R_j
1	$\{z_1\ z_2\}$	{ (01) (10) (11) }	2	4	3	3/4
2	$\{z_1\ z_3\}$	{ (00) (01) (02) (10) (11) (12) }	2	6	6	1
3	$\{z_1\ z_3\}$	{ (12) }	2	6	1	1/6
4	$\{z_1\ z_2\ z_3\}$	{ (001) (002) (010) (011) (012) (100) (101) (102) (110) (111) (112) }	3	12	11	11/12

Fig. 2-2: The two solutions of instance clp_0, and their projections onto the various constraint argument sets.



$\bar{Z}$:	(1 0 2)	(1 1 2)
$\bar{Z}_1(\bar{Z})$:	(1 0)	(1 1)
$\bar{Z}_2(\bar{Z})$:	(1 2)	(1 2)
$\bar{Z}_3(\bar{Z})$:	(1 2)	(1 2)
$\bar{Z}_4(\bar{Z})$:	(1 0 2)	(1 1 2)

$$D_1 = d_{z_1} \times d_{z_2} = \{ (00)\,(01)\,(10)\,(11) \}$$

$$D_2 = D_3 = d_{z_1} \times d_{z_3} = \{ (00)\,(01)\,(02)\,(10)\,(11)\,(12) \} \qquad (12)$$

$$D_4 = d_{z_1} \times d_{z_2} \times d_{z_3} = \{ (000)\,(001)\,(002)\,(010)\,(011)\,(012)\,(100)\,(101)\,(102)\,(110)\,(111)\,(112) \}$$

Since $M_j = |\ D_j\ |$, we have $M_1 = 4$, $M_2 = M_3 = 6$ and $M_4 = 12$. The clp_0 constraints in (8) to (11) are expressed in analytic form. Table 2-2 lists the pairs $(Z_j\ T_j)$ constituting the canonical form of each of these analytic constraints. By definition, each T_j shown contains all the labelings of Z_j that satisfy constraint C_j. The satisfiabilities or sizes $S_j = |\ T_j\ |$ of these relations are then $S_j = 3, 6, 1$ and 11. Dividing these values by the corresponding M_j values we have that the satisfiability ratios for the four constraints are $R_1 = 3/4$, $R_2 = 6/6 = 1$, $R_3 = 1/6$ and $R_4 = 11/12$. Summing the four constraint satisfiabilities gives a summed-satisfiability of $SS = 21$.

Like the earlier 4-queens example, instance clp_0 has $S = 2$ solutions or consistent labelings. The solution set for clp_0 is shown in figure 2-2 to be $T = \{ (1\ 0\ 2)\ (1\ 1\ 2) \}$. This will be confirmed in [52] where instance clp_0 is solved in detail in a variety of ways. Note that in the standard formulation of QUEENS instances as CLP instances, the board rows correspond to CLP variables and the columns to candidate values for these variables. Whereas in clp_0 no board occurs in the problem statement, it is nevertheless convenient to display, as in figure 2-2, consistent labelings in an analogous array form with rows corresponding to variables and columns to values. The i-th cell in each row of such arrays corresponds to the i-th value of the corresponding variable's domain under some assumed ordering. (Note that i-th column cells of different rows need not correspond to the same value, since different domains need not even have values of the the same types.) Such boards are thus always left-justified but not necessarily right-justified, since domains may be of different sizes. This representation will be useful in [52] when depicting the searches performed by our CLP algorithms.

2.4. The Class CLP

Having defined the generic CLP instance $clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$, the class CLP is defined as the set of all actual instances obtained from the generic instance as its parameters range over their respective allowed values. Specifically, the number of variables may be any integer $n > 1$, each component domain size of $\mathbf{m}$ may independently be any integer $m_{z_i} > 0$ and the number of constraints may be any integer $c > 0$. Each component constraint argument set of $\mathbf{Z}$ may independently be any subset $Z_j \subseteq Z$ (containing at least two variables) of the set of problem variables Z, and each component constraint relation of $\mathbf{T}$ may independently be any subset $T_j \subseteq D_j$ of the set D_j of possible value-tuples for the constraint argument variables.

This problem class is much more general than that usually treated in the literature, whether theoretically or empirically. This is essential if the results are to be applicable to the wide range of CLP instances that arise in practice either directly or (as discussed in chapter 7 of [51]) via transformation of a CLP instance to an alternative representation within CLP. In particular, it should be noted that the above definition does *not* require that problem variables have domains of the same size, and certainly not that domain sizes are all equal to the number of variables, nor that constraints are all of the same arity, and certainly not that the arities all be two (i.e. binary constraints). Nor does it require that all (and only) pairs of variables have a constraint over them as in [36], [57], nor that all (and only) A-tuples of variables have a constraint over them as in [58]. The above formulation allows that a given set of problem variables may have an arbitrary number — none, one or even more than one — constraints defined over them. Although for some purposes multiple constraints on a given argument set could be considered equivalent to a single constraint, for a model relevant to algorithmic run-time it is necessary to explicitly allow multiple constraints on a given set of arguments, since they will usually be implemented and checked separately. Moreover, subsets of variables that do not have a constraint over them need no longer be treated by the unrealistic expedient of assuming that they are constrained by a universal (maximally loose) constraint. This is certainly not assumed in the actual problem-solving process, and our analyses will make it possible to model the fact that in practice, missing constraints are simply not checked.

2.5. CLP Sub-Problems

Two types of sub-Problem of CLP are particularly important for our analyses. These we call **big-classes** and **small-classes.** CLP is partitioned by either type of class, but small-classes form a finer partition, and have the important property of homogeneity as a result of which small-class expectations provide effectively instance-specific predictions. Results are given for special-form big-classes in [36], [58] and [57]. The latter paper also treats special-form small-classes. We have recently been able to derive results for fully arbitrary big- and small-classes, [51], [52].

In his thesis [26], Gaschnig, besides also using (special-form) big-classes, studied CLP empirically using yet another type of CLP sub-Problem which might be called the (special-form) **middle-sized class.** His results show that middle-sized classes, like big-classes, do not have the desirable homogeneity property, and on page 241 of [26] he proposes small-classes as an improvement. Small-classes were however independently arrived at in this work, as a refinment of Haralick's big-classes. The definitions of these classes are as follows:

The generic big-class $\mathrm{CLP}_{\beta}(n\ \mathbf{m}\ c\ \mathbf{Z})$ is the set of all CLP instances having the generic values n, $\mathbf{m}$, c and $\mathbf{Z}$ respectively for the number of variables, vector of domain sizes, number of constraints and vector of constraint-argument sets. The component constraint-relations T_j of an instance's constraint-relation vector $\mathbf{T}$ are arbitrary subsets of the corresponding cartesian products D_j.

$$\mathrm{CLP}_{\beta}(n\ \mathbf{m}\ c\ \mathbf{Z}) = \{clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})\ \mid\ T_j \subseteq D_j \quad 1 \le j \le c\ \} \tag{13}$$

The generic middle-sized class $\mathrm{CLP}_{\mu}(n\ \mathbf{m}\ c\ \mathbf{Z}\ SS)$ is the set of all CLP instances having the generic values n, $\mathbf{m}$, c and $\mathbf{Z}$ for the corresponding features. In addition, the constraint-relations T_j of an instance are such that the sum of their sizes is SS.

$$\mathrm{CLP}_{\mu}(n\ \mathbf{m}\ c\ \mathbf{Z}\ SS) = \{clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})\ \mid\ \sum_{j=1}^{c} |T_j| = SS\ \} \tag{14}$$

The generic small-class $\mathrm{CLP}_{\sigma}(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$ is the set of all CLP instances having the generic values n, $\mathbf{m}$, c and $\mathbf{Z}$ for the corresponding features. In addition, each constraint-relation T_j of an instance has size S_j, the j-th component of the small-class constraint-satisfiability vector $\mathbf{S}$.

$$\mathrm{CLP}_{\sigma}(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S}) = \{clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})\ \mid\ |T_j| = S_j \quad 1 \le j \le c\ \} \tag{15}$$

In a big-class, each of the relations T_j for a given instance may be an arbitrary one of the 2^{M_j} subsets of D_j. In a middle-sized class, the *pooled relation* $\bigcup_{1\le j\le c} T_j$ is restricted to be one of the $\binom{\sum_{1\le j\le c} M_j}{SS}$ subsets of size SS of the *pooled cartesian product* $\bigcup_{1\le j\le c} D_j$. In a small-class each T_j is restricted to be one of the $\binom{M_j}{S_j}$ subsets of D_j having size S_j. It follows that the number of instances in a big-class, in a middle-sized class and in a small-class, are given respectively by

$$|\,\mathrm{CLP}_{\beta}(n\ \mathbf{m}\ c\ \mathbf{Z})\,| = \prod_{j=1}^{c} 2^{M_j} \tag{16}$$

$$|\,\mathrm{CLP}_{\mu}(n\ \mathbf{m}\ c\ \mathbf{Z}\ SS)\,| = \binom{\sum_{j=1}^{c} M_j}{SS} \tag{17}$$

$$|\,\mathrm{CLP}_{\sigma}(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})\,| = \prod_{j=1}^{c} \binom{M_j}{S_j} \tag{18}$$

Moreover the numbers of middle-sized-classes and of small-classes in a big-class $\mathrm{CLP}_{\beta}(n\ \mathbf{m}\ c\ \mathbf{Z})$ are respectively

$$1 + \sum_{j=1}^{c} M_j \qquad \text{and} \qquad \prod_{j=1}^{c} (1 + M_j) \tag{19}$$

since in the former case the value of SS may be any integer between 0 and $\sum_{1 \le j \le c} M_j$, and in the latter case each component of the satisfiability vector $\mathbf{S} = (S_1\ S_2\ .\ .\ S_c)$ may be any value from 0 to M_j. The number of small-classes in a middle-sized class is given by the result of the Diophantine constraint example of section 2.6.1, as mentioned in a footnote of that section. Summing instances over all small-classes in a big-class we have the following interesting generalization of the binomial expansion of $2^M = (1 + 1)^M$

$$\prod_{j=1}^{c} 2^{M_j} = \sum_{\mathbf{S}=(00\ .\ 0)}^{(M_1 M_2\ .\ M_c)} \prod_{j=1}^{c} \binom{M_j}{S_j}$$

since the number of instances in a big-class equals the sum of the number in each of its constituent small-classes.

The difference between the above three types of CLP sub-Problem is the degree to which the constraint-satisfiabilities S_j are restricted. An orthogonal kind of restriction of historic interest in inducing sub-Problems of CLP is one placed on $\mathbf{Z}$, the vector of argument-sets. It is often assumed that instances are **binary**, that is, have constraints only over *pairs* of variables. Moreover, it is often assumed, as is the case for n-queens instances (under the usual formulation), that *all* possible pairs of variables have exactly one constraint over them. Instances with *exactly* one constraint over each pair of variables, and having no other constraints, we call **pure and simple binary.** If there is *at least* one constraint for each pair of variables, and no other constraints, such instances are called **pure** (but not necessarily simple) **binary.**

Most systematic studies in Artificial Intelligence have considered only pure and simple binary instances. In particular, this is the case in [36] where pure and simple binary big-classes are used, in [26] where pure and simple binary big- and middle-sized classes are used, and in [57] where pure and simple binary big- and small-classes are used.[16] The next level of generality occurs in [58] where we consider **pure and simple A-ary** big-classes — big-classes of instances having exactly one constraint over each combination of A variables, and no other constraints. (If each combination of size 2 through A variables has exactly one constraint, and no other constraints exist, such instances are called **full and simple A-ary.**) No other analysis had ever treated that general a case before. However, our results of [51], and [52] are far more general yet, being in terms of fully arbitrary small-classes and big-classes, with no restriction at all placed on the vector of argument sets Z_j characterizing instances of the class — and this full generality is carried through in the expected-complexity analyses of [51] and [52].

2.6. Finding Constraint Satisfiabilities

As mentioned, the vector $\mathbf{S} = (S_1\ S_2\ .\ .\ S_c)$ of constraint satisfiabilities is an important problem parameter because it allows us to partition CLP into relatively homogeneous small-classes $\mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$. The homogeneity of small-classes is desirable, because it means that a class-average result will approximate well the corresponding instance-specific, or exact-case, values for the subsumed instances in the small-class. Of course, to make this approximation for a given instance requires us to first identify the small-class to which the instance belongs — in order that the appropriate small-class average may be obtained.

Identifying the small-class $\mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$ to which a given CLP instance belongs means, by definition, determining for that instance the values of the small-class parameters n, $\mathbf{m}$, c, $\mathbf{Z}$ and $\mathbf{S}$. Values for the first four of these parameters will always be trivial to determine for a given instance. Such parameters might therefore be called **surface parameters.** The constraint satisfiabilities S_j may also be surface parameters for a given instance if say, the instance constraints C_j are given in canonical form $C_j = (Z_j\ T_j)$, for then we have simply that $S_j = |\ T_j\ |$. In several important domains — such as the relational database and line labeling applications discussed in [51] — constraints do arise in canonical form. However, constraints of real-world problems often are expressed in some non-canonical, symbolic form for which the corresponding value of constraint satisfiability may not be obvious. In such cases S_j is what we call a **deep parameter** as opposed to a surface parameter, and a certain computational cost is

[16] In [57] the (pure and simple binary) big-classes and small-classes are called respectively, v-classes and c-classes.

incurred in extracting the value of S_j.[17] The benefit[18] obtained from knowing the S_j values must then be weighed against any preprocessing cost incurred to extract these values. In this section we consider a spectrum of approaches to obtaining satisfiability values S_j for non-canonical constraints. An earlier version of the following appears in [55].

2.6.1. Exact Analysis

Sometimes satisfiabilities can be obtained exactly by analytic methods, thus incurring no machine cost. The following are three examples where this is possible. When this is not possible one may settle for only an approximation to S_j, or be prepared to incur some machine preprocessing cost or both. These alternatives will be described in the next sections.

Example 1: q-Queens Constraints

For the q-queens problem under the standard formulation described in section 2.3, the constraint satisfiability and satisfiability ratio of each of the resulting $\binom{q}{2}$ constraints in (7) can be expressed as

$$\begin{aligned} S_{z_i z_j} &= q^2 - 3q + 2 \mid i - j \mid \\ R_{z_i z_j} &= \frac{q^2 - 3q + 2 \mid i - j \mid}{q^2} \end{aligned} \qquad \forall\, i < j \in \{1\ 2\ ..\ q\} \tag{20}$$

These are obtained as follows. Both the queen in row i and the queen in row j can a` priori be in any of the $q$ squares of these rows. Therefore the two queens can a` priori be in any of q^2 pairs of squares. However, for these two queens to be placed so as not to attack each other, for each of the q possible squares for the queen in row i, the queen in row j can not be in the square in the same column or the same right or left diagonal. Therefore the q^2 a` priori allowed pairs of positions of the two queens must be reduced by $3q$. However, this over-reduces. If the row-i queen is in a square close enough to the right (left) of the board and rows i and j are far enough apart, then the corresponding rightmost (leftmost) of the three positions excluded for the row j queen was not actually on the board in the first place. That is, the diagonal from the row-i queen's position goes off the board by the time it reaches row j. A little thought shows that an excess of $2 \mid i - j \mid$ positions for the queen in row j were removed. This must be added back to $q^2 - 3q$, giving the required result (20). The satisfiability ratio for constraint $C_{z_i z_j}$ is by definition $R_{z_i z_j} = S_{z_i z_j} / M_{z_i z_j}$. Since $M_{z_i z_j} = q^2$ for all pairs of variables, we thus also obtain $R_{z_i z_j}$ of (20).

Table 2-3 shows the satisfiabilities $S_{z_i z_j}$ for all constraints of the $q = 4$, 6 and 8 queens problems. Note that $S_{z_i z_j}$ is the same for all rows i and j that have the same separation $\mid i - j \mid$. This is as expected. In fact, a little though shows that for q-queens not just the satisfiabilities $S_{z_i z_j}$ but also the relations $T_{z_i z_j}$ themselves must be the same for constraints between equally-spaced variables — see for example table 2-1 for 4-queens. Note also that the larger the row separation, the greater is $S_{z_i z_j}$ and the looser is the corresponding constraint. This also is as expected for the situation in an q-queens problem.

From (20), and to some extent from the examples in table 2-3, we see that as q grows, $R_{z_i z_j}$ goes to 1 for any pair of variables z_i, z_j. In other words, the constraints become looser and looser so that virtually all pairs of column positions are allowed for a given pair of queens. This is why, as q grows, the number of solutions for the q-queens problem tends to infinity.[19] It is also the reason for the meager advantage reported by Haralick (in table 1 of [36]) for the use of "optimal" as opposed to "normal" instantiation order in solving q-queens instances. Whereas in general, optimal ordering can make a considerable difference (as seen in table 2 of [36] and in chapter 8 of [51], where random instances were

[17] This is the price paid for the extra precision afforded by including S as an analysis parameter. Avoiding the problem by simply dropping S from the analysis produces results in terms of big-classes — and these have in fact also been obtained in [52]. But big-class expectations are not (directly) useful for our purpose of approximating exact-case complexities, since big-classes do not exhibit the necessary homogeneity. (It will be seen however that big-class expectations may be used indirectly for this purpose, by approximating small-class expectations.)

[18] Such as the computational savings obtained from use of the theory-based heuristics, derived in chapter 8 of [51] from analytic expressions for small-class expected complexity.

[19] The expected number of solutions for CLP instances is treated in section 3.5.2, after the relevant probability models have been introduced.

Table 2-3: The satisfiabilities $S_{z_i z_j}$ and the satisfiability ratios $R_{z_i z_j}$ for all the constraints of the q = 4, 6 and 8 queens problems.

q	$S_{z_i z_j}$				$R_{z_i z_j}$			
4	$i \backslash j$	2	3	4	$i \backslash j$	2	3	4
	1	6	8	10	1	.375	.500	.625
	2		6	8	2		.375	.500
	3			6	3			.375

q	$S_{z_i z_j}$						$R_{z_i z_j}$					
6	$i \backslash j$	2	3	4	5	6	$i \backslash j$	2	3	4	5	6
	1	20	22	24	26	28	1	.555	.611	.666	.722	.777
	2		20	22	24	26	2		.555	.611	.666	.722
	3			20	22	24	3			.555	.611	.666
	4				20	22	4				.555	.611
	5					20	5					.555

q	$S_{z_i z_j}$								$R_{z_i z_j}$							
8	$i \backslash j$	2	3	4	5	6	7	8	$i \backslash j$	2	3	4	5	6	7	8
	1	42	44	46	48	50	52	54	1	.656	.688	.719	.750	.781	.813	.844
	2		42	44	46	48	50	52	2		.656	.688	.719	.750	.781	.813
	3			42	44	46	48	50	3			.656	.688	.719	.750	.781
	4				42	44	46	48	4				.656	.688	.719	.750
	5					42	44	46	5					.656	.688	.719
	6						42	44	6						.656	.688
	7							42	7							.656

used), this does not show up with q-queens instances simply because as q grows the constraints in a q-queens instance all become nearly identical, virtually universal constraints. Since q-queens variables are distinguished only by the differences in the relations T_j of the constraints for which they are arguments, the variables of q-queens become indistinguishable and all instantiation orders tend to become effectively equivalent as q grows.

Example 2: Logical Constraints

The satisfiability of many logical constraints may be determined analytically.[20] Let C_j denote a constraint over A_j literals l_i formed from A_j different propositional variables z_i (so that therefore, no two literals are the same or are complementary). Letting integers 1 and 0 denote *True* and *False* respectively, allows us to use addition and multiplication to conveniently express a wide range of constraints besides the usual ones AND, OR, NAND and NOR. Some of these are shown in table 2-4. For example, the constraint that exactly m literals of the A_j literals are *True* may be denoted analytically as

[20] The particular case of logical constraints that are disjunctions of literals arises in the natural CLP formulation of the Propositional Satisfiability Problem. This is discussed in chapter 7 of [51] in the context of CLP applications to theorem proving. In an analogous manner, Schaeffer's generalized Satisfiability Problem [67] results in more general logical constraints when modeled as CLP in the natural way. However, even Satisfiability with fully arbitrary conjuncts is still of course very much subsumed by CLP, since CLP allows fully arbitrary constraints and as well allows variables to have an arbitrary finite number of values, rather than only the two values *True* and *False* that a propositional variable may take on.

$$C_j: \sum_{i=1}^{A_j} l_i = m \tag{21}$$

since this requires that exactly m of the l_i have value 1. The satisfiability of this constraint is $\binom{A_j}{m}$ since there are $\binom{A_j}{m}$ ways to chose the m literals that will be *True* from the total of A_j literals. Note that for $m = 0$, the above constraint becomes *None* or *NOR* since it requires that no literal be *True*. The corresponding constraint satisfiability is then $S_j = \binom{A_j}{0} = 1$; there is only one way to make none of the literals *True*. For $m = A_j$ the constraint becomes *All* or *AND* since it requires that all literals be true. The corresponding satisfiability is then $S_j = \binom{A_j}{A_j} = 1$; there is only one way to make all of the literals *True*.

Table 2-4 shows these and other logical constraints, together with their corresponding satisfiabilities S_j. (Some constraints are expressed in analytic form in more than one way.) Results are also shown for the conjunctive normal form (cnf) and the disjunctive normal form (dnf) constraints of the following forms respectively:

$$\bigwedge_{i=1}^{c} \bigvee_{k=1}^{d_i} l_{ik} \quad \text{and} \quad \bigvee_{i=1}^{d} \bigwedge_{k=1}^{c_i} l_{ik}$$

That is, the cnf constraint consists of c conjuncts, the i-th of which is a disjunction of d_i literals, the k-th literal in the i-th conjunct being denoted l_{ik}. Similarly, the dnf constraint consists of d disjuncts, the i-th of which is a conjunction of c_i literals, the k-th literal in the i-th disjunct being denoted l_{ik}. The cnf and dnf constraint satisfiabilities are obtained as follows.

(i) For the cnf constraint, each component conjunct (being a disjunction) can be satisfied by $2^{d_i} - 1$ truth assignments to its d_i literals. The overall cnf expression is satisfied iff each component conjunct is satisfied, and this may be ensured by combining any satisfying assignments for each conjunct — giving a total of $S_j = \prod_{1 \le i \le c} (2^{d_i} - 1)$ satisfying truth assignments to the $A_j = \sum_{1 \le i \le c} d_i$ literals.

(ii) A dnf constraint constraint may be expressed as the negation of a cnf constraint over complemented corresponding literals as follows

$$\bigvee_{i=1}^{d} \bigwedge_{k=1}^{c_i} l_{ik} = \sim\sim \bigvee_{i=1}^{d} \bigwedge_{k=1}^{c_i} l_{ik} = \sim \bigwedge_{i=1}^{d} \bigvee_{k=1}^{c_i} \overline{l_{ik}}$$

The satisfiability of the dnf constraint, as shown in table 2-4, is therefore 2^{A_j} (the number of all possible truth assignments to the new complemented literals $\overline{l_{ik}}$, of which there are still A_j) minus the satisfiability of the unnegated cnf expression, this latter value being as derived above.

Remember, we are assuming above that no two of the A_j literals in the constraints are repeated or are complementary. If this restriction is removed, the satisfiability of cnf and dnf constraints is no longer a function of just the numbers of conjuncts, disjuncts and their sizes. It will also depend on the specific literals involved in the conjuncts and disjuncts. For example, the cnf constraint $(z_1 \vee z_2) \wedge (z_3 \vee z_4)$ has satisfiability 9 as required above, but the cnf constraint $(z_1 \vee z_2) \wedge (\overline{z_1} \vee z_4)$ containing the complementary literals z_1 and $\overline{z_1}$, has a different satisfiability, 4. For the fully general cnf or dnf constraint, the satisfiability may however be derived as an expected satisfiability under a given probability model. For cnf constraints the expected satisfiabilities over big-classes and over small-classes are in fact implicit in the big-class and small-class expected number of solutions $\overline{S}$ for a CLP instance, derived below in section 3.5.2. This is so because satisfying a cnf constraint is just a special case of solving a CLP instance, each conjunct corresponding to a CLP constraint.

Note that in all cases of constraints on A_j literals formed from A_j propositional variables, the total possible number of truth assignments is 2^{A_j}. The satisfiability ratios $R_j = S_j / M_j$ for the constraints in table 2-4 are therefore the corresponding S_j values divided by 2^{A_j}.

Table 2-4: The satisfiabilities S_j for some logical constraints C_j on A_j different and non-complementary literals

	C_j		S_j
	Common Name(s)	Analytic Form(s)	
1	Exactly-m	$\sum_{i=1}^{A_j} l_i = m$	$\binom{A_j}{m}$
2	Exactly-0, None, NOR	$\sum_{i=1}^{A_j} l_i = 0$	$\binom{A_j}{0} = 1$
3	Exactly-A_j, All, AND	$\sum_{i=1}^{A_j} l_i = A_j$ or $\prod_{i=1}^{A_j} l_i = 1$	$\binom{A_j}{A_j} = 1$
4	At-Least-m	$\sum_{i=1}^{A_j} l_i \geq m$	$\sum_{k=m}^{A_j} \binom{A_j}{k}$
5	At-Least-1, Not-None, OR	$\sum_{i=1}^{A_j} l_i \geq 1$	$\sum_{k=1}^{A_j} \binom{A_j}{k} = 2^{A_j} - 1$
6	At-Most-m	$\sum_{i=1}^{A_j} l_i \leq m$	$\sum_{k=0}^{m} \binom{A_j}{k}$
7	At-Most-(A_j - 1), Not-All, NAND	$\sum_{i=1}^{A_j} l_i \leq A_j - 1$ or $\prod_{i=1}^{A_j} l_i = 0$	$\sum_{k=0}^{A_j-1} \binom{A_j}{k} = 2^{A_j} - 1$
8	conjunctive normal form	$\prod_{i=1}^{c} \sum_{k=1}^{d_i} l_{ik} \geq 1$ or $\sum_{i=1}^{c} \prod_{k=1}^{d_i} (1 - l_{ik}) = 0$	$\prod_{i=1}^{c} (2^{d_i} - 1)$
9	disjunctive normal form	$\sum_{i=1}^{d} \prod_{k=1}^{c_i} l_{ik} \geq 1$ or $\prod_{i=1}^{d} \sum_{k=1}^{c_i} (1 - l_{ik}) = 0$	$2^{A_j} - \prod_{i=1}^{d} (2^{c_i} -.1)$

Example 3: Diophantine Constraints

Our third example of extracting constraint satisfiability analytically applies to the important class of linear constraints of the form

$$C_j : \; c_1 z_1 + c_2 z_2 + \,.\,.\, + c_{A_j} z_{A_j} = c_0 \qquad (22)$$

where the c_i are integers, and each variable z_i takes its values from a set of consecutive integers from α_i to β_i inclusive, so that the domain of z_i is $d_{z_i} = \{\alpha_i \; \alpha_i+1 \,.\,.\, \beta_i\}$. The constraint is therefore a Diophantine equation.[21] The satisfiability of a Diophantine constraint (or the number of solutions of a Diophantine equation), can be found by the inclusion-exclusion principle as seen in a special case on page 154 of [75]. One can also use the method of generating functions as on page 58 of [40] where the fully general constraint (22) was used but the special case $\alpha_i = 0$ was assumed for the lower bound of all the domains. The more general case above can be put in a form with lower bounds $\alpha_i = 0$ by transforming to new variables $x_i = z_i - \alpha_i$. The resulting constraint in terms of x_i has the same satisfiability as (22) in terms of variables z_i. Extending the result of [40] accordingly we have then for the satisfiability of constraint (22) that

$$S_j = \text{the coefficient of } t^c \text{ in the expansion of } \prod_{i=1}^{A_j} \Big(\sum_{k=0}^{\gamma_i} t^{kc_i} \Big) \qquad (23)$$

$$\text{where} \quad c = c_0 - \sum_{i=1}^{A_j} c_i \alpha_i \qquad \text{and} \quad \gamma_i = \beta_i - \alpha_i$$

A simple example is the constraint $3z_1 + 5z_2 = 1$ with domains $d_{z_1} = \{-4 \; -3 \,.\,.\, 8\}$ and $d_{z_2} = \{-2 \; -1 \,.\,.\, 4\}$. By (23) the satisfiability, or number of solutions, is the coefficient of t^{23} in the expansion of

$$(1 + t^3 + t^6 + \,.\,.\, + t^{36})\,(1 + t^5 + t^{10} + \,.\,.\, + t^{30})$$

which equals 2. This can be confirmed by drawing the appropriate line and boundaries in the cartesian plane. One sees that there are in fact two solutions: $(z_1 \; z_2) = (-3 \; 2)$ and $(2 \; -1)$. The satisfiability ratio for the constraint is in this case $R_j = S_j / M_j = 2/(13 \times 7) = 2/91$. Of course in the general case $M_j = \prod_{i=1}^{A_j} (\gamma_i + 1)$.

2.6.2. Analytic Approximation

Sometimes, in order to maintain an analytic approach to the obtaining of satisfiability, it may be necessary to settle for only an approximate value. As a simple example, consider the inequality constraint $az_1 + bz_2 < c$ where z_1 and z_2 take values in respective sets d_{z_1} and d_{z_2} of consecutive integers. The exact satisfiability of such a constraint is difficult to determine analytically. Hardy and Littlewood [38] have struggled with similar problems. However, an approximation to the above satisfiability is readily obtained as the area in the plane bounded by the line $az_1 + bz_2 = c$ and the horizontal and vertical lines bounding domains d_{z_1} and d_{z_2}.

2.6.3. Monte Carlo Approximation

When no analytic method is available to determine satisfiability of a constraint exactly or to a sufficiently good approximation, an approximation may always be obtained by the obvious method of randomly sampling from the M_j value-tuples $\underline{Z}_j \in D_j$ for the argument set Z_j of the constraint. Each such tuple is explicitly tested to see if it satisfies constraint C_j. The ratio of the number of satisfying tuples to total tuples tested provides an approximation to the satisfiability ratio R_j. This ratio multipled by M_j approximates the constraint satisfiability S_j. The accuracy of the approximation of course increases with the size of the sample, but the cost of the approximation becomes accordingly more expensive. Techniques from [32] may prove helpful. In the limit this approach becomes exhaustive testing, described next.

[21] Note that the satisfiability-vector **S** of a small-class in a given middle-sized class characterised by SS must satisfy the Diophantine equation $S_1 + S_2 + \,.\,.\, + S_c = SS$, for $0 \le S_j \le M_j$. The satisfiability of this Diophantine equation thus equals the number of small-classes in a middle-sized class.

2.6.4. Exhaustive Testing

If the satisfiability of a constraint is required exactly, and no analytic method is available, then Monte Carlo sampling can be extended to exhaustive sampling in which all possible M_j value tuples $\overline{Z}_j \in D_j$ are explicitly tested for satisfaction of constraint C_j. The number of tuples found to satisfy the constraint is of course by definition the required satisfiability S_j. The computational cost will be M_j constraint checks. If all c constraints of a given CLP instance must have their satisfiabilities determined in this way then this will incur a preprocessing cost of $\sum_{1 \le j \le c} M_j$ checks.

3. PROBABILITY MODELS and RESULTS for CLP

This section introduces several probability models for the distribution of instances in the CLP small-classes and big-classes defined in section 2. These classes are thus turned from sets of instances into sample spaces where the instances are outcomes. Various events are then defined, their probabilities derived and certain expected-values determined over small-classes and big-classes under the different probability models.

The results in this section are all *algorithm-independent* — reflecting only the problem class and the probability model concerned. In [52], these results are used to obtain the expectations of *algorithm-dependent* quantities — in particular the expected complexity of solving a CLP instance via the three CLP algorithms: Backtracking, Forward-Checking and word-wise Forward-Checking. Note that while the results of this section and of [52] are of course dependent on the probability-model assumed in the corresponding derivations, this dependence is relatively minor for the expected complexities over small-classes — this being precisely why small-classes are so important here. The homogeneity of small-classes (see sections 1.2 and 2.5) allows our small-class expected-complexities to be largely probability-model independent[22] and moreover, allows them to provide good approximations to the exact-case values for most subsumed instances in the corresponding small-class.

Thus our small-class expected-complexities are to be seen both as interesting expectations in their own right, as well as being useful approximations for exact-case values. On the other hand, our big-class expected-complexities at first glance would seem to be of interest only in the first sense — since big-classes are far from homogeneous. However, as will be discussed in section 3.5.1, big-class expectations can be used to provide good approximations for small-class expectations, and hence can also be used as "indirect" approximations for exact-case values.

3.1. Some Notation

This section presents some notation and results that will be useful. Section 3 assumes a familiarity with the theory of probability for finite sample spaces, useful references for which are [18], [59] and [40].

Probability Spaces: We will be associating probability models with CLP big-classes and small-classes. To emphasize that these big- and small-classes then become sample-spaces — where CLP instances are outcomes — we write $\Omega_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ and $\Omega_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$, or more simply Ω_β and Ω_σ, respectively for $\mathrm{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ and $\mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$. An instances $clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$ or more simply clp, may accordingly be denoted as $\omega(n\ m\ c\ \mathbf{Z}\ \mathbf{T})$ or ω. When the distinction between big-class Ω_β and small-class Ω_σ is not important we use Ω to stand for either one.

We will often be discussing parameterized events $E = E(a_1\ ..\ a_m)$ in Ω, where the a_i are arguments whose values determine the specific event intended. In this case, we abbreviate probability $P(E(a_1\ ..\ a_m))$ of the event by $P(a_1\ ..\ a_m)$. Of course, for events $E_j,\ j \in J$ that are mutually independent or mutually disjoint we have respectively

$$P(\bigcap_{j \in J} E_j) = \prod_{j \in J} P(E_j) \quad \text{and} \quad P(\bigcup_{j \in J} E_j) = \sum_{j \in J} P(E_j) \tag{24}$$

We will mostly be working with the following definition of the expected value of a random variable $Q(\omega)$

$$\overline{Q} = \sum_{\omega \in \Omega} Q(\omega) P(\omega) \tag{25}$$

which is equivalent to the more usual definition as a sum over all possible values q of $Q(\omega)$, each value weighted by its probability $P(q)$.

Indicator Functions: A given event $E \subseteq \Omega$ can be represented by its **characteristic** or **indicator function**

$$\delta(E\ \omega) = \begin{cases} 1 & \text{if } \omega \in E \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

When the event E is given parametrically as above, then we abbreviate $\delta(E(a_1\ ..\ a_m)\ \omega)$ as

[22] Remember that an expected-case value must lie between the worst-case value and the best-case value for a class, and if all instances of the class have similar complexity values, then the expected-case value cannot vary much irrespective of the assumed probability model.

$\delta(a_1 \,..\, a_m \; \omega)$. A result that will prove useful is that the expected value of the indicator function of an event E is equal to the probability of that event:

$$\bar{\delta}(E) = \sum_{\omega \in \Omega} \delta(E \; \omega)\, P(\omega) = \sum_{\omega \in E} P(\omega) = P(E) \tag{27}$$

The Inclusion-Exclusion Principle: The probability of a union of not necessarily disjoint sets is given by the following **inclusion-exclusion formula**

$$P(\bigcup_{j=1}^{n} E_j) = P_1 - P_2 + P_3 - \ldots (-1)^{n+1} P_n \tag{28}$$

$$\text{where } P_l = \sum_{j_1 < j_2 < \,.\, < j_l} P(E_{j_1} \cap E_{j_2} \,..\, \cap E_{j_l}) \quad 1 \le j_k \le n$$

In words, P_l is the sum of the probabilities of all l-fold intersections of events E_j. When all sets E_j are subsets of a set $E_0 \subseteq \Omega$, then the probability of the complement in E_0 of the union of all E_j can be expressed in terms of (28) as

$$P(\,(\bigcup_{j=1}^{n} E_j)^{c(E_0)}\,) = P(E_0) - P(\bigcup_{j=1}^{n} E_j) = P_0 - P_1 + P_2 - P_3 + \ldots (-1)^n P_n \tag{29}$$

where $P_0 = P(E_0)$ and where $A^{c(B)}$ denotes the compliment of set A in its superset B. In the special case that $E_0 = \Omega$ we have $P_0 = 1$ in (29).

3.2. Probability Models over Big-Classes and Small-Classes

Within a given big-class or small-class, the instances $clp(n \; \mathbf{m} \; c \; \mathbf{Z} \; \mathbf{T})$ differ only in the value of their constraint-relation vector $\mathbf{T} = (T_1 \; T_2 \,..\, T_c)$. Thus a probability model for the instances of a big-class or small-class is equivalent to one for the corresponding set of $\mathbf{T}$ vectors. By the definition of big-class and small-class (in section 2.5) we have that the relation-vectors $\mathbf{T}$ for the instances in a big-class $\mathrm{CLP}_\beta(n \; \mathbf{m} \; c \; \mathbf{Z})$ and in a small-class $\mathrm{CLP}_\sigma(n \; \mathbf{m} \; c \; \mathbf{Z} \; \mathbf{S})$ are respectively those given by

$$\mathbf{T} \in \bigtimes_{j=1}^{c} 2^{D_j} \quad \text{and} \quad \mathbf{T} \in \bigtimes_{j=1}^{c} \binom{D_j}{S_j} \tag{30}$$

The following are three probability models for the distribution of these $\mathbf{T}$ vectors. Models 0 and 1 induce a distribution on the vectors given in the first part of (30), and hence induce a distribution on the CLP instances of a big-class. Model-2 induces a distribution on the vectors given in the second part of (30), and hence induces a distribution on the instances of a small-class.

Table 3-1: CLP Probability Models Used

In all three models, the vector $\mathbf{T} = (T_1 \; T_2 \,..\, T_c)$ of constraint relations arises by c independent experiments, one for each component T_j.

0 **Model-0 on a big-class:** A given T_j arises as the result of M_j independent experiments, each determining for a different one of the value-tuples $\bar{Z}_j \in D_j$ whether or not it is in T_j. Each $\bar{Z}_j$ has probability p of belonging to T_j.

1 **Model-1 on a big-class:** As for model-0, but generalizes so that each $\bar{Z}_j$ in D_j has probability p_j of belonging to T_j.

2 **Model-2 on a small-class:** A given T_j arises by a single experiment of randomly selecting a subset of S_j value-tuples from D_j. All subsets of size S_j are equally likely.

Model-0 is essentially that used by Haralick [36] except that we apply it to a more general type of big-class. Model-1 generalizes model-0 in allowing the probability for value-tuple inclusion in a relation T_j to be p_j instead of p and thus to vary with the relation. Parameter p_j (and p) we call the **satisfiability rate** for the j-th constraint, since it is the probability that any given value-tuple $\overline{Z}_j \in D_j$ satisfies constraint $C_j = (Z_j\ T_j)$. Note that under model-1, selecting an instance is a compound experiment made up of $\sum_{1 \le j \le c} M_j$ independent component experiments: M_1 Bernoulli trials with success rate p_1, M_2 trials with success rate $p_2 \ldots$ and M_c trials with success rate p_c.

Thus probability model-1 is characterized by a vector $\mathbf{p} = (p_1 \ldots p_c)$ of c satisfiability rates, one for each constraint C_j. Model-0 results when $\mathbf{p} = (p\ p\ \ldots\ p)$ and is characterized by a single satisfiability rate p. We use $P_\beta^1(E\ ;\ \mathbf{p})$ to denote the probability of the event that a random instance clp is in set E when selected from big-class $\mathrm{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ under model-1 with satisfiability-rate vector $\mathbf{p} = (p_1 \ldots p_c)$. The corresponding expression for model-0 is $P_\beta^0(E\ ;\ p)$.

The simple assumptions of model-0 and model-1 are not appropriate for model-2, where instances are constrained to belong to a small-class characterized by some vector $\mathbf{S}$ of constraint-relation sizes whereas under models 0 and 1 any size $0 \le S_j \le M_j$ was possible for the j-th constraint.

As with models 0 and 1, the experiment of selecting a CLP instance is a compound experiment made up of c independent component experiments of selecting the respective individual relations T_j. However unlike models 0 and 1, the relation T_j is selected from $\overline{D}_j$ as a single indivisible experiment of simultaneously selecting S_j tuples $\overline{Z}_j$ from the M_j tuples in D_j. (Equivalently, one can view the selection of a relation T_j under model-2 as corresponding to a sequence of S_j independent selections *without* replacement, of value-tuples $\overline{Z}_j$ from D_j.) Corresponding to $P_\beta^1(E\ ;\ \mathbf{p})$ and $P_\beta^0(E\ ;\ p)$ above, we use $P_\sigma^2(E)$ to denote the probability that event E occurs when a CLP instance is randomly selected from a small-class under model-2. Note that model-2 does not depend on any distribution parameter such as parameters p and $\mathbf{p}$ of models 0 and 1 respectively.

For big-classes under model-0 and under model-1, and for small-classes under model-2, the expected value of a random variable $Q(clp)$ is given by the corresponding versions of (25), which are respectively

$$\overline{Q}_\beta^0(n\ \mathbf{m}\ c\ \mathbf{Z}\ ;\ p) = \sum_{clp\ \epsilon\ \Omega_\beta(n\,\mathbf{m}\,c\,\mathbf{Z})} Q(clp)\, P_\beta^0(clp\ ;\ p) \tag{31}$$

$$\overline{Q}_\beta^1(n\ \mathbf{m}\ c\ \mathbf{Z}\ ;\ \mathbf{p}) = \sum_{clp\ \epsilon\ \Omega_\beta(n\,\mathbf{m}\,c\,\mathbf{Z})} Q(clp)\, P_\beta^1(clp\ ;\ \mathbf{p}) \tag{32}$$

$$\overline{Q}_\sigma^2(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S}) = \sum_{clp\ \epsilon\ \Omega_\sigma(n\,\mathbf{m}\,c\,\mathbf{Z}\,\mathbf{S})} Q(clp)\, P_\sigma^2(clp) \tag{33}$$

3.3. Events

Having introduced the probability models above, we now consider some events that are of intrinsic interest and/or that will be needed in [52] when deriving the expected complexity of solving CLP instances. The events are with respect to a big-class or a small-class as the sample-space, and are defined in table 3-2. In the next section, the probabilities of such events will be determined according to the various probability models that apply. Table 3-10 summarizes these probabilities for the events of table 3-2, under models 1 and 2. Note that all results of section 3 are independent of any algorithm. However, they are closely related to algorithm-dependent quantities of interest, and in [52] we employ them in deriving algorithm-dependent events, probabilities and expectations for the Backtracking, Forward Checking and word-wise Forward Checking algorithms.

As indicated in table 3-2, the events we consider can be divided into three types, based on (i) whether constraints of an instance are satisfied by a given value-tuple, (ii) whether constraint relations of an instance are equal to given sets and (iii) whether constraint relations of an instance have a certain size. The following paraphrases and expands on the formal specification of the events in table 3-2.

- $E(j\ \overline{Z}_j)$ is the set of instances in Ω whose j-th constraint relation $T_j(clp)$ is satisfied by a given value-tuple $\overline{Z}_j$ from the corresponding cartesian product.
- $E(j\ \overline{X})$ is the set of instances in Ω whose j-th constraint relation $T_j(clp)$ is satisfied by a given value-tuple $\overline{X}$ labeling $X \subseteq Z$. X need not equal the list Z_j of argument variables for T_j, but may subsume it. In this case, T_j is satisfied by $\overline{X}$ if the projection $\overline{Z}_j(\overline{X})$ of $\overline{X}$ onto Z_j is in T_j.

Table 3-2: Some Basic Subsets of $\Omega = \Omega_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ or $\Omega_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$.
(The corresponding event probabilities are in table 3-10.)

Symbol	Defined as	Defined for
Events based on constraint satisfaction		
$E(j\ \overline{Z}_j)$	$\{clp \mid \overline{Z}_j \in T_j(clp)\}$	$\overline{Z}_j \in D_j,\ j \in J_1^c$
$E(j\ \overline{X})$	$\{clp \mid \overline{Z}_j(\overline{X}) \in T_j(clp)\}$	$X \subseteq Z,\ Z_j \subseteq X,\ j \in J_1^c$
$E(J\ \overline{X})$	$\{clp \mid \overline{Z}_j(\overline{X}) \in T_j(clp)\ \forall\ j \in J\}$	$X \subseteq Z,\ J \subseteq J_1^c,\ Z_j \subseteq X\ \forall\ j \in J$
$E(J\ f\ \ t\ \overline{X})$	$\{clp \mid \exists$ exactly t values $\overline{f} \in d_f$ s.t. $[\overline{Z}_j(\overline{X}\ \overline{f}) \in T_j(clp)\ \forall\ j \in J]\ \}$	$X \subset Z,\ J \subseteq J_1^c,\ f \notin X,$ $\{f\} \subseteq Z_j \subseteq X \cup \{f\}\ \forall\ j \in J,\ t \in J_0^{m_f}$
$E(J\ f\ \ \geq 1\ \overline{X})$	$\{clp \mid \exists$ at least one value $\overline{f} \in d_f$ s.t. $[\overline{Z}_j(\overline{X}\ \overline{f}) \in T_j(clp)\ \forall\ j \in J]\ \}$	$X \subset Z,\ J \subseteq J_1^c,\ f \notin X,$ $\{f\} \subseteq Z_j \subseteq X \cup \{f\}\ \forall\ j \in J$
Events based on constraint equality		
$E(j\ T_j)$	$\{clp \mid T_j(clp) = T_j\}$	$\left\{\begin{array}{ll} T_j \in 2^{D_j} & \text{Model-1} \\ T_j \in \binom{D_j}{S_j} & \text{Model-2} \end{array}\right.\quad j \in J_1^c$
$E(\mathbf{T})$	$\{clp \mid \mathbf{T}(clp) = \mathbf{T}\}$	$\left\{\begin{array}{ll} \mathbf{T} \in \times_{j=1}^{c} 2^{D_j} & \text{Model-1} \\ \mathbf{T} \in \times_{j=1}^{c} \binom{D_j}{S_j} & \text{Model-2} \end{array}\right.$
Events based on constraint-size equality		
$E(j\ S_j)$	$\{clp \mid S_j(clp) = S_j\}$	$\left\{\begin{array}{ll} S_j \in J_0^{M_j} & \text{Model-1} \\ S_j = S_j(\Omega_\sigma) & \text{Model-2} \end{array}\right.\quad j \in J_1^c$
$E(\mathbf{S})$	$\{clp \mid \mathbf{S}(clp) = \mathbf{S}\}$	$\left\{\begin{array}{ll} \mathbf{S} \in \times_{j=1}^{c} J_0^{M_j} & \text{Model-1} \\ \mathbf{S} = \mathbf{S}(\Omega_\sigma) & \text{Model-2} \end{array}\right.$

- $E(J\ \bar{X})$ is the set of instances in Ω whose constraints in the set indexed by J are all satisfied by a given value-tuple $\bar{X}$ labeling $X \subseteq Z$.
- $E(J\ f\ t\ \bar{X})$ is the set of instances in Ω for which exactly t of the m_f possible values in domain d_f for variable f each allow all the constraints that are indexed by J to be satisfied when other argument variables take their values as given by $\bar{X}$. Thus $E(J\ f\ t\ \bar{X})$ can be viewed as the event that exactly t values for f each "survive" all the constraints in $T_J = \{T_j \mid j \in J\}$ given the instantiations in $\bar{X}$. Note that as indicated in table 3-2, this set of instances is well defined only for f not in X and when each constraint in T_J has f as one argument and takes its other arguments only from the variables X. This and the next set of instances will be important in the analysis of algorithm gFC.
- $E(J\ f\ \geq 1\ \bar{X})$ is as for $E(J\ f\ t\ \bar{X})$ except that instead of requiring exactly t surviving values from d_f, we require at least 1.
- $E(j\ T_j)$ is the set of instances in Ω whose j-th constraint relation $T_j(clp)$ is equal to a given relation T_j.
- $E(\mathbf{T})$ is the set of instances in Ω whose vector $\mathbf{T}(clp) = (T_1(clp)\ .\,.\ T_c(clp))$ of constraint relations is equal to a given relation vector $\mathbf{T} = (T_1\ .\,.\ T_c)$. As mentioned, within a given big or small-class there is a one-to-one correspondence between vectors $\mathbf{T}$ and instances $clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$, so that the set $E(\mathbf{T})$ is really just the singleton set $\{clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})\}$. As a consequence, the probability of event $E(\mathbf{T})$ is the probability of outcome $clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$.
- $E(j\ S_j)$ is the set of instances in Ω for which the size $S_j(clp) = \mid T_j(clp) \mid$ of the j-th constraint relation $T_j(clp)$, is equal to a given number S_j.
- $E(\mathbf{S})$ is the set of instances in Ω whose vector $\mathbf{S}(clp) = (S_1(clp)\ .\,.\ S_c(clp))$ of constraint relation sizes, is equal to a given vector $\mathbf{S} = (S_1\ .\,.\ S_c)$ of sizes. In table 3-2, $\mathbf{S}(\Omega_\sigma)$ denotes the satisfiability vector $\mathbf{S}$ characterizing the small-class $\Omega_\sigma = \mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$, and $S_j(\Omega_\sigma)$ denotes its j-th component.

Having now defined the events of interest, table 3-3 presents some relationships between them which will be useful when deriving the event probabilities of the next section. Remember from table 1-1, that the symbol $\dot{\cap}$ indicates an intersection between sets which as events are mutually independent. Analogously, $\dot{\cup}$ denotes a union between mutually disjoint sets.

(34) follows since an instance clp has a relation vector $\mathbf{T}(clp) = (T_1(clp)\ .\,.\ T_c(clp))$ equal to a given vector $\mathbf{T} = (T_1\ .\,.\ T_c)$ iff $T_j(clp) = T_j,\ \forall\, 1 \leq j \leq c$. Most importantly, for different j the events $E(j\ T_j)$ are mutually independent in all models, since the relations $T_j(clp)$ arise via independent experiments under all models.

(35) is a direct consequence of the definitions of $E(j\ \bar{Z}_j)$ and $E(j\ \bar{X})$.

(36) follows analogously to (34).

(37) follows since at least one value for f surviving is the same as the event that one value survives, or two values, . . , or the maximum m_f values survive, these possibilities all being mutually disjoint. It is also equal to the certain event minus the event that zero values survive.

(38) says simply that $\bar{Z}_j \in T_j(clp)$ for instance clp iff $T_j(clp) = T_j$ where $\bar{Z}_j \in T_j$. Sets $E(j\ T_j)$ are of course disjoint for different T_j since $T_j(clp_1) \neq T_j(clp_2)$ implies that $clp_1 \neq clp_2$.

(39) is the only relation in table 3-3 that is not applicable under all probability models. As indicated by the sub and super scripts, it applies for a big-class under model-1 (or model-0 as a special case). It says that a given set T_j will equal the j-th relation $T_j(clp)$ of an instance clp iff all value-tuples $\bar{Z}_j$ of T_j are in $T_j(clp)$ and all $\bar{Z}_j$ not in T_j are not in $T_j(clp)$. As such, this is also true under model-2. However (39) also says that events $E(j\ \bar{Z}_j)$ are mutually independent for different $\bar{Z}_j$, and this is not the case under model-2, but only for models 0 and 1 where the inclusion of $\bar{Z}_j$ in T_j is decided by an independent experiment for each $\bar{Z}_j$.

(40) says simply that $S_j = \mid T_j(clp) \mid$ for instance clp iff $T_j(clp) = T_j$ where $\mid T_j \mid = S_j$. As with (38), sets $E(j\ T_j)$ are disjoint for different T_j.

(41) follows since an instance clp has a relation-size vector $\mathbf{S}(clp) = (S_1(clp)\ .\,.\ S_c(clp))$ equal to a given vector $\mathbf{S} = (S_1\ .\,.\ S_c)$ iff $S_j(clp) = S_j,\ \forall\, 1 \leq j \leq c$. As in (34) and (36), for different j, the events $E(j\ T_j)$ are mutually independent.

Table 3-3: Some Relationships between the Events of Table 3-2.

$$E(\mathbf{T}) = \dot{\bigcap}_{j=1}^{c} E(j\ T_j) \tag{34}$$

$$E(j\ \bar{X}) = E(j\ \bar{Z}_j(\bar{X})) \tag{35}$$

$$E(J\ \bar{X}) = \dot{\bigcap}_{j \in J} E(j\ \bar{X}) \tag{36}$$

$$E(J\ f\ \geq 1\ \bar{X}) = \dot{\bigcup}_{t=1}^{m_f} E(J\ f\ t\ \bar{X}) = \Omega - E(J\ f\ 0\ \bar{X}) \tag{37}$$

$$E(j\ \bar{Z}_j) = \dot{\bigcup}_{T_j \text{ s.t. } \bar{Z}_j \in T_j} E(j\ T_j) \tag{38}$$

$$E^1_\beta(j\ T_j) = \dot{\bigcap}_{\bar{Z}_j \in T_j} E^1_\beta(j\ \bar{Z}_j) \dot{\bigcap}_{\bar{Z}_j \in D_j - T_j} E^1_\beta - E^1_\beta(j\ \bar{Z}_j) \tag{39}$$

$$E(j\ S_j) = \dot{\bigcup}_{T_j \text{ s.t. } |T_j| = S_j} E(j\ T_j) \tag{40}$$

$$E(\mathbf{S}) = \dot{\bigcap}_{j=1}^{c} E(j\ S_j) \tag{41}$$

3.4. Probabilities

In this section we derive the probabilities of the events in table 3-2 under probability models 1 and 2. These probabilities appear in tables 3-5 and 3-6 respectively. Results for model-0 follow as a special case of those for model-1, and are not given explicitly. For convenient comparison, table 3-10 combines the results of tables 3-5 and 3-6 for models 1 and 2.

The relationships of table 3-3 between the events of table 3-2 can readily be converted into corresponding relationships between the event probabilities by taking the probability of both sides of the equations in table 3-3 and using (24) to simplify. The resulting probability equations are shown in table 3-4.

3.4.1. Probabilities Under Models 0 and 1

Table 3-5 presents the probabilities $P^1_\beta(E\ ;\ \mathbf{p})$ for the events $E \subseteq \Omega$ of table 3-2, under probability model-1 over a big-class $\Omega = \Omega_\beta = \mathrm{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$. The model-0 probabilities $P^0_\beta(E\ ;\ p)$ follow as a special case by using p for all components p_j of model-1 parameter vector $\mathbf{p} = (p_1\ .\ .\ p_c)$.

(50) follows by definition from model-1 (since the relations T_j of an instance arise independently and an arbitrary tuple $\bar{Z}_j$ from cartesian product D_j has probability p_j of belonging to T_j). Note that the result is the same for all $\bar{Z}_j$ in D_j.

Note that (46) gives an expression for $P(j\ \bar{Z}_j)$ in terms of $P(j\ T_j)$. This equation will be useful for finding the model-2 version of $P(j\ \bar{Z}_j)$ in the next section, but in the present model-1 (and model-0) case the equation is not useful since $P(j\ \bar{Z}_j)$ follows much more directly in the manner just shown. Nevertheless, (46) remains valid for model-1, and it is interesting to check that it is satisfied by (50)

Table 3-4: Relationships Between Probabilities of Events of Table 3-2
(based on table 3-3)

$$P(\mathbf{T}) = \prod_{j=1}^{c} P(j\ T_j) \tag{42}$$

$$P(j\ \bar{X}) = P(j\ \bar{Z}_j(\bar{X})) \tag{43}$$

$$P(J\ \bar{X}) = \prod_{j \in J} P(j\ \bar{X}) \tag{44}$$

$$P(J\ f\ \geq 1\ \bar{X}) = \sum_{t=1}^{m_f} P(J\ f\ t\ \bar{X}) = 1 - P(J\ f\ 0\ \bar{X}) \tag{45}$$

$$P(j\ \bar{Z}_j) = \sum_{T_j \text{ s.t. } \bar{Z}_j \in T_j} P(j\ T_j) \tag{46}$$

$$P_\beta^1(j\ T_j) = \prod_{\bar{Z}_j \in T_j} P_\beta^1(j\ \bar{Z}_j) \prod_{\bar{Z}_j \in D_j - T_j} 1 - P_\beta^1(j\ \bar{Z}_j) \tag{47}$$

$$P(j\ S_j) = \sum_{T_j \text{ s.t. } |T_j| = S_j} P(j\ T_j) \tag{48}$$

$$P(\mathbf{S}) = \prod_{j=1}^{c} P(j\ S_j) \tag{49}$$

Table 3-5: Model-1 Probabilities for the Events of Table 3-2

$$P_\beta^1(j\ \bar{Z}_j\ ;\mathbf{p}) = p_j \tag{50}$$

$$P_\beta^1(j\ \bar{X}\ ;\mathbf{p}) = p_j \tag{51}$$

$$P_\beta^1(J\ \bar{X}\ ;\mathbf{p}) = \prod_{j \epsilon J} p_j \tag{52}$$

$$P_\beta^1(J\ f\ t\ \bar{X}\ ;\mathbf{p}) = \binom{m_f}{t} \Big[\prod_{j \epsilon J} p_j\Big]^t \Big[1 - \prod_{j \epsilon J} p_j\Big]^{m_f - t} \tag{53}$$

$$P_\beta^1(J\ f\ \geq 1\ \bar{X}\ ;\mathbf{p}) = 1 - \Big[1 - \prod_{j \epsilon J} p_j\Big]^{m_f} \tag{54}$$

$$P_\beta^1(j\ T_j\ ;\mathbf{p}) = p_j^{\ S_j} (1 - p_j)^{M_j - S_j} \quad \text{where } |\ T_j\ | = S_j \tag{55}$$

$$P_\beta^1(clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})\ ;\mathbf{p}) = P_\beta^1(\mathbf{T}\ ;\mathbf{p}) = \prod_{j=1}^{c} p_j^{\ S_j} (1 - p_j)^{M_j - S_j} \quad \text{where } |\ T_j\ | = S_j \tag{56}$$

$$P_\beta^1(j\ S_j\ ;\mathbf{p}) = \binom{M_j}{S_j} p_j^{\ S_j} (1 - p_j)^{M_j - S_j} \tag{57}$$

$$P_\beta^1(\mathbf{S}\ ;\mathbf{p}) = \prod_{j=1}^{c} \binom{M_j}{S_j} p_j^{\ S_j} (1 - p_j)^{M_j - S_j} \tag{58}$$

and (55) of table 3-5. This is done by partitioning the T_j of (46) according to their size S_j and noting that there are $\binom{M_j - 1}{S_j - 1}$ different T_j of size S_j such that $\bar{Z}_j\ \epsilon\ T_j$ for a given $\bar{Z}_j$. Using (55) and the Binomial Expansion in reverse, we then obtain — as required by (50) — for the right side of (46)

$$\sum_{S_j = 1}^{M_j} \binom{M_j - 1}{S_j - 1} p_j^{\ S_j} (1 - p_j)^{M_j - S_j} = p_j\ (p_j + (1 - p_j))^{M_j - 1} = p_j \tag{59}$$

(51) follows from (50) using (43). Note that since in (50) the result is independent of $\bar{Z}_j$, (51) is also independent of $\bar{X}$.

(52) follows from (51) using (44). Like (51), it is also independent of $\bar{X}$.

(53) will be explained in the following section. Again, this probability is independent of $\bar{X}$.

(54) follows from (53) and (45). This probability is also independent of $\bar{X}$.

(55) follows from (47) and (50). Note that as expected, it is just the probability of a specific distribution of $|\ T_j\ | = S_j$ successes in M_j Bernoulli trials of success rate p_j.

(56) follows from (55) and (42). Note that due to the one-to-one correspondence between instances and relation vectors **T**, (56) gives the probability of outcome $clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$ in big-class $\mathrm{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ under model-1.

(57) follows from (55) and (48), using the fact that there are $\binom{M_j}{S_j}$ different T_j of size S_j that can be selected from the M_j tuples of D_j. Note that as expected, (57) is just the Binomial probability of S_j successes in M_j Bernoulli trials of success rate p_j.

(58) follows from (57) and (49). Note that (58) gives the probability that a random instance of big-class $\mathrm{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ falls in the particular subsumed small-class $\mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$.

3.4.2. Probabilities Under Model-2

Table 3-6 is the model-2 analog of table 3-5. It presents the probabilities $P_\sigma^2(E\ ;\ \mathbf{p})$ for the events $E \subseteq \Omega$ of table 3-2, under probability model-2 over a small-class $\Omega = \Omega_\sigma = \mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$.

(65) follows essentially by definition from model-2 (since the relations T_j of an instance arise independently and since there are $\binom{M_j}{S_j}$ equally-likely ways to choose relation T_j). This is in contrast to the model-1 case, where it was (50), the analog of (60), that followed most directly.

(66) follows from (65) and (42). Note that due to the one-to-one correspondence between instances and relation vectors $\mathbf{T}$, (66) gives the probability of outcome $clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$ in small-class $\mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$ under model-2. Moreover, since (65) is independent of T_j, (66) is therefore independent of $\mathbf{T}$ and we see that model-2 does induce a uniform distribution on the CLP instances of a small-class. This is in contrast to the highly non-uniform distribution (56) that model-1 (or model-0) induces over the instances of a big-class.

Table 3-6: Model-2 Probabilities for the Events of Table 3-2

$$P_\sigma^2(j\ \bar{Z}_j) = R_j = S_j/M_j \tag{60}$$

$$P_\sigma^2(j\ \bar{X}) = R_j = S_j/M_j \tag{61}$$

$$P_\sigma^2(J\ \bar{X}) = \prod_{j\in J} R_j = \prod_{j\in J} S_j/M_j \tag{62}$$

$$P_\sigma^2(J\ f\ t\ \bar{X}) = \frac{\binom{m_f}{t}\sum_{l=0}^{m_f-t}(-1)^l\binom{m_f-t}{l}\prod_{j\in J}\binom{M_j-t-l}{S_j-t-l}}{\prod_{j\in J}\binom{M_j}{S_j}} \tag{63}$$

$$P_\sigma^2(J\ f\ \geq 1\ \bar{X}) = 1 - \frac{\sum_{l=0}^{m_f}(-1)^l\binom{m_f}{l}\prod_{j\in J}\binom{M_j-l}{S_j-l}}{\prod_{j\in J}\binom{M_j}{S_j}} \tag{64}$$

$$P_\sigma^2(j\ T_j) = \binom{M_j}{S_j}^{-1} \tag{65}$$

$$P_\sigma^2(clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})) = P_\sigma^2(\mathbf{T}) = \prod_{j=1}^{c}\binom{M_j}{S_j}^{-1} \tag{66}$$

$$P_\sigma^2(j\ S_j) = 1.0 \tag{67}$$

$$P_\sigma^2(\mathbf{S}) = 1.0 \tag{68}$$

(67) follows simply because under model-2 all selections T_j must be of size S_j by definition. More specifically, it can be obtained from (48) by adding $P(j\ T_j)$ of (65) once for each of the $\binom{M_j}{S_j}$ possible selections T_j since all are of size S_j.

(68) follows from (67) and (49). Note that this is just the probability that an outcome in small-class $\Omega_\sigma = \mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$ is in that small-class, and this is of course the certain event Ω_σ. Results (67) and (68) are actually included here only for symmetry with their model-1 analogs (57) and (58).

(60) follows from (65) and (46). There are $\binom{M_j-1}{S_j-1}$ selections T_j that contain a specified tuple $\bar{Z}_j$, since the other S_j-1 value-tuples to make up the required total of S_j must be chosen from the other M_j-1 possible tuples of D_j. There are thus this many terms in the sum (46), each term contributing the same probability from (65). The sum, and hence $P_\sigma^2(j\ \bar{Z}_j)$, is thus equal to $\binom{M_j-1}{S_j-1}/\binom{M_j}{S_j} = S_j/M_j = R_j$ as in (60). Note that as for (50) in the model-1 case, (60) is the same for all $\bar{Z}_j$ in D_j.

(61) follows from (60) and (43). Note that since in (60) the result is independent of $\bar{Z}_j$, (61) is also independent of $\bar{X}$ as was its analog (51) in the model-1 case.

(62) follows from (61) and (44). Like (61), it is also independent of $\bar{X}$.

(63) will be explained in the following section. Again, this probability is independent of $\bar{X}$.

(64) follows from (63) and (45). This probability is also independent of $\bar{X}$.

3.4.3. The Probability of t Surviving Domain Values

We have now derived the probability, under both models 1 and 2, of all the events in table 3-2, except for one. This is event $E(J\ f\ t\ \bar{X})$ that exactly t values from the m_f values in the domain of variable f survive all the constraints indexed by J, given that other argument variables take their values as given by the instantiation list $\bar{X}$. The problem of finding the probabilities $P_\beta^1(J\ f\ t\ \bar{X})$ and $P_\sigma^2(J\ f\ t\ \bar{X})$ of this event under models 1 and 2 is relatively difficult, at least in the model 2 case. However, the problem is of considerable interest. Similar versions have cropped up in seemingly unrelated alternate forms in various other disciplines; see the note at the end of [57].

Tables 3-7 and 3-9 below present four alternate forms of our problem, in the model 1 and model 2 cases respectively. Formulations A_1 and A_2 are our original model-1 and model-2 problems. Formulations C_1 and C_2 correspond to the recursive version of these problems employed in [57]. Formulations D_1 and D_2 correspond to the work of Shenton [69] and of Bernard [3]. Note that in these equivalent problems J is an integer equal to the size of the original index set so that $J = |J|$, and m equals the domain size m_f of the original problem.

3.4.3.1. Under Models 0 and 1

The probabilities sought in the four problems of table 3-7 are all equal — this being the intended sense in which the problems are equivalent. Here, we solve our original problem A_1 of finding $P_\beta^1(J\ f\ t\ \bar{X})$ by finding probability $P_{B_1}(J\ m\ t)$ of problem B_1.

An example of the kind of J-tuple required in problem B_1 is shown in figure 3-1. This depicts $J = 3$ subsets or selections T_j taken from sets D_j, $1 \le j \le 3$. The D_j are of sizes $|D_1| = M_1 = 9$, $|D_2| = M_2 = 7$ and $|D_3| = M_3 = 8$. Selected integers are shown underlined. The selections happen to be of sizes $|T_1| = S_1 = 5$, $|T_2| = S_2 = 6$ and $|T_3| = S_3 = 6$, but note that there is nothing in problem B_1 that requires this. Exactly $t = 3$ of the first $m = 5$ integers are common to all three selected sets. This is also true for $m = 3$, 4, and 6 but not for $m = 7$ unless t were 4, since integer 7 is also common to all three selections. As implied in this $t = 4$, $m = 7$ case, the t integers common to all the selections need not be the first t integers $\{1\ 2\ .\ .\ t\}$, but may be any subset of the first m integers.

In forming the J-tuples $\mathbf{T} = (T_1\ T_2\ .\ .\ T_J)$ of problem B_1 each integer of D_j independently has probability p_j of being selected for T_j and the T_j are selected independently. Therefore each integer has probability $\prod_{j \in J} p_j$ of being selected in all J selections. The probability that exactly t of the first m integers is selected in all J selections is then just the Binomial probability of t successes in a run of m Bernoulli trails with success rate $\prod_{j \in J} p_j$. Therefore the probability required by problem B_1 is just

Table 3-7: Four Equivalent Problems

(A_1) **A Constraint Survival Problem:** Find the probability $P_\beta^1(J\ f\ t\ \bar{X})$ of event $E(J\ f\ t\ \bar{X})$ of table 3-2 under probability model 1. Briefly, this is the probability that in a CLP big-class under model 1, a random instance allows exactly t values from the domain of variable f to each satisfy all the constraints indexed by J, when other argument variables take their values as in instantiation list $\bar{X}$.

(B_1) **A J-Urn Coincidence Problem:** From each set of integers $D_j = \{1\ 2\ 3\ .\ .\ M_j\}$ $1 \le j \le J$, select independently a subset T_j. Each integer of D_j independently has probability p_j of being selected to belong in T_j. Given a number $m \le \underset{1 \le j \le J}{\text{Min}} \{M_j\}$, what is the probability $P_{B_1}(J\ m\ t)$ that of the first m integers $\{1\ 2\ .\ .\ m\}$, exactly t are common to all J sets selected? Explicitly, we require the probability of the event

$$E(J\ m\ t) = \{\ (T_1\ T_2\ .\ .\ T_J)\ :\ \left| \bigcap_{j=1}^{J} T_j \bigcap J_1^m \right| = t\}$$

This is trivially equivalent to a J-urn problem where the sets D_j correspond to urns containing numbered balls. It is also clearly equivalent to having a "chess board" of J rows with the j-th row having M_j squares. The rows are aligned at the left, but not necessarily at the right. In each row j, balls are placed in each square with probability p_j. What is the probability that exactly t of the first m columns are filled (i.e. have all J squares occupied by a ball)?

(C_1) **A Non-stationary Markov Chain Problem:** Let $P_{C_1}(j\ m\ t)$ be the probability of being in state t of the states $\{0\ 1\ 2\ .\ .\ m\}$ after exactly j steps of a non-stationary Markov chain

$$P_{C_1}(0\ m\ t) = \begin{cases} 1 & \text{if } t = m \\ 0 & \text{otherwise} \end{cases}$$

$$P_{C_1}(j\ m\ t) = \sum_{u=0}^{m} P_{C_1}(j{-}1\ m\ u)\, P(j\ u\ t) \qquad j \ge 1$$

having a nonstationary Binomial transition probability $P(j\ u\ t)$ for going from state u to state t at step j

$$P(j\ u\ t) = \binom{u}{t} p_j^t\, (1-p_j)^{u-t}$$

Note that $P(j\ u\ t) = 0$ for $t > u$. Find $P_{C_1}(J\ m\ t)$, the probability of being in state t after exactly J steps.

(D_1) **A Single-Urn J-Cycle Addition-Deletion Problem:** A single urn initially contains only m red balls. A sequence of J cycles is performed on the urn, each cycle consisting of an addition of an addition phase where white balls are added to the urn, followed by a deletion phase where each ball in the urn is independently deleted with probability $1 - p_j$. If d_j balls are deleted at the deletion phase of the j-th cycle, then the number of white balls added during the j-th addition phase is

$$a_j = \begin{cases} 0 & \text{if } j = 1 \\ d_{j-1} & \text{if } 2 \le j \le J \end{cases}$$

Find the probability $P_{D_1}(J\ m\ t)$ that, after J such addition-deletion cycles, there are exactly t red balls left in the urn.

Fig. 3-1: Three selections in which exactly $t = 3$ integers of the first $m = 5$ are common to all selections. They happen also to be the *first* 3 integers.

$$\begin{array}{lcc}
 & M_j & S_j \\
D_1 = \{\ \underline{1}\,\underline{2}\,\underline{3} \mid \underline{4}\,5 \mid 6\,\underline{7}\,8\,9\ \} & 9 & 5 \\
D_2 = \{\ \underline{1}\,\underline{2}\,\underline{3} \mid \underline{4}\,\underline{5} \mid 6\,\underline{7}\ \} & 7 & 6 \\
D_3 = \{\ \underline{1}\,\underline{2}\,\underline{3} \mid 4\,\underline{5} \mid \underline{6}\,\underline{7}\,8\ \} & 8 & 6
\end{array}$$

$$P_{B_1}(J\ m\ t) = \binom{m}{t} \left[\prod_{j=1}^{J} p_j \right]^t \left[1 - \prod_{j=1}^{J} p_j \right]^{m-t} \tag{69}$$

with special cases:

$$P_{B_1}(1\ m\ t) = \binom{m}{t} \left[p_1 \right]^t \left[1 - p_1 \right]^{m-t} \tag{70}$$

$$P_{B_1}(J\ m\ m) = \left[\prod_{j=1}^{J} p_j \right]^m \tag{71}$$

$$P_{B_1}(J\ m\ 0) = \left[1 - p_1 \right]^m \tag{72}$$

Note that (70) is a Binomial distribution and so is (69), independent of the value of J. All that changes is the success rate. In the model 2 case below we will see that the $J = 1$ case is a hypergeometric distribution, but that for larger J the hypergeometric form is *not* maintained. By the correspondence between problem B_1 and our original problem A_1, we have

$$P^1_\beta(J\ f\ t\ \bar{X};\mathbf{p}) = \binom{m_f}{t} \left[\prod_{j \in J} p_j \right]^t \left[1 - \prod_{j \in J} p_j \right]^{m_f - t} \tag{73}$$

$$P^1_\beta(J\ f\ \geq 1\ \bar{X};\mathbf{p}) = 1 - \left[1 - \prod_{j \in J} p_j \right]^{m_f} \tag{74}$$

Result (74) follows from (73) by use of (45). Note that probabilities (73) and (74) are actually independent of the argument $\bar{X}$ as well as being independent of the sizes M_j. The model 1 probabilities of (50) to (52) were similar in this respect.

Solving problem B_1 above to obtain (69) was quite easy. The analogous solution for problem B_2 of the next section will be considerably harder to obtain. It will require the use of the inclusion-exclusion formula (29). Actually, even though the above method of solution suffices, problem B_1 can also be solved via the inclusion-exclusion formula. We now carry out such a solution since it emphasizes the similarity between problems B_1 and B_2 and much of the work can in any case be carried over to the solution of B_2.

Let us first make explicit the sample space for problem B_1. Remember that a J-tuple $\mathbf{T}$ of selections T_j is obtained by independently selecting each T_j from the corresponding set D_j. Selection T_j may be any subset of D_j, hence any element of the power set 2^{D_j} of D_j. Thus we may consider the outcomes $\mathbf{T}$ as elements from a product space Ω made up of component spaces $\Omega^j = 2^{D_j}$.

$$\Omega = \mathop{\times}_{j=1}^{J} \Omega^j \qquad \Omega^j = 2^{D_j} \tag{75}$$

Table 3-8: Some Events in Ω.

Symbol	Defined as	Defined for
$E(J\ m\ t)$	$\{\mathbf{T} : \lvert \bigcap_{j=1}^{J} T_j \bigcap J_1^m \rvert = t\}$	$0 \le t \le m$
$E(J\ m\ J')$	$\{\mathbf{T} : \bigcap_{j=1}^{J} T_j \bigcap J_1^m = J'\}$	$J' \subseteq J_1^m$
E_0	$\{\mathbf{T} : \bigcap_{j=1}^{J} T_j \supseteq J_1^t\}$	
E_0^j	$\{\mathbf{T} : T_j \supseteq J_1^t\}$	
E_i	$\{\mathbf{T} : \bigcap_{j=1}^{J} T_j \supseteq J_1^t \bigcup \{i\}\}$	$t+1 \le i \le m$
E_i^j	$\{\mathbf{T} : T_j \supseteq J_1^t \bigcup \{i\}\}$	$t+1 \le i \le M_j,\ 1 \le j \le J$
$E^j(i)$	$\{\mathbf{T} : T_j \supseteq \{i\}\}$	$1 \le i \le M_j,\ 1 \le j \le J$

Problem B_1 asks for the probability $P_{B_1}(J\ m\ t)$ of event $E(J\ m\ t)$ that a random J-tuple $\mathbf{T}$ has component selections T_j having exactly t integers from the first m integers in common. This event can be written explicitly as $E(J\ m\ t)$ of table 3-8. Now $E(J\ m\ t)$ can be partitioned in terms of disjoint events $E(J\ m\ J')$ as follows

$$E(J\ m\ t) = \biguplus_{\substack{J' \subseteq J_1^m \\ |J'| = t}} E(J\ m\ J') \tag{76}$$

so that by (24)

$$P(J\ m\ t) = \sum_{\substack{J' \subseteq J_1^m \\ |J'| = t}} P(J\ m\ J') \tag{77}$$

There are $\binom{m}{t}$ terms in this sum corresponding to the ways to choose t integers from the first m and by symmetry each term is equal. Thus

$$P(J\ m\ t) = \binom{m}{t} P(J\ m\ J_1^t) \tag{78}$$

where $E(J\ m\ J_1^t)$ is any one of the $\binom{m}{t}$ events partitioning $E(J\ m\ t)$, let's say the one for which exactly the <u>first</u> t integers are those common to all T_j from amongst the first m. The simpler events of table 3-8 can be used to re-express this latter event. But first we present some relationships between events of table 3-8 that will be needed later. Clearly the following equalities hold.

$$E_i = \bigcap_{j=1}^{J} E_i^j \tag{79}$$

$$E_0 = \bigcap_{j=1}^{J} E_0^j \tag{80}$$

$$E_i^j = \bigcap_{k=1}^{t} E^j(k) \bigcap E^j(i) \tag{81}$$

$$E_0^j = \bigcap_{k=1}^{t} E^j(k) \tag{82}$$

$$P(\,E^j(i)\,) = p_j \tag{83}$$

We now express $E(J\ m\ J_1^t)$ as a compliment in E_0 in term of the E_i, as follows

$$E(J\ m\ J_1^t) = (\,\bigcup_{i=t+1}^{m} E_i\,)^{c(E_0)} \tag{84}$$

This is valid since it says simply that **T** has exactly the first t of the first m integers common to all selections T_j iff it has the first t integers common to all T_j but does not have any integer from $t+1$ to m common to all selections. The probability of $E(J\ m\ J_1^t)$ can then be found using the inclusion-exclusion formula (29). It therefore remains only to find P_l of (28), which in the present context is

$$P_l = \sum_{i_1 < i_2 < \,.\, < i_l} P(E_{i_1} \bigcap E_{i_2} \,.\,.\, \bigcap E_{i_l}) \quad t+1 \le i_k \le m \tag{85}$$

There are $\binom{m-t}{l}$ terms in this sum corresponding to the different l-fold intersections formable amongst the $m-t$ sets E_i. As above in deriving (78), all such intersections have the same probability. Using the probability of a particular l-fold intersection as representative of all other such intersections, we have

$$P_l = \binom{m-t}{l} P\left(\bigcap_{i=t+1}^{t+l} E_i \right) \tag{86}$$

$$= \binom{m-t}{l} P\left(\dot{\bigcap_{j=1}^{J}} \bigcap_{i=t+1}^{t+l} E_i^j \right) \quad \text{by (79)} \tag{87}$$

$$= \binom{m-t}{l} \prod_{j=1}^{J} P\left(\bigcap_{i=t+1}^{t+l} E_i^j \right) \quad \text{by (24)} \tag{88}$$

$$= \binom{m-t}{l} \prod_{j=1}^{J} P\left(\bigcap_{i=t+1}^{t+l} \left(\dot{\bigcap_{k=1}^{t}} E^j(k) \,\dot{\bigcap}\, E^j(i) \right) \right) \quad \text{by (81)} \tag{89}$$

$$= \binom{m-t}{l} \prod_{j=1}^{J} P\left(\dot{\bigcap_{i=1}^{t+l}} E^j(i) \right) \tag{90}$$

$$= \binom{m-t}{l} \prod_{j=1}^{J} \prod_{i=1}^{t+l} P(E^j(i)) \quad \text{by (24)} \tag{91}$$

$$= \binom{m-t}{l} \left[\prod_{j=1}^{J} p_j \right]^{t+l} \quad \text{by (83)} \tag{92}$$

Similarly

$$P_0 = P(E_0) \tag{93}$$

$$= P\left(\dot{\bigcap_{j=1}^{J}} E_0^j \right) \quad \text{by (80)} \tag{94}$$

$$= \prod_{j=1}^{J} P(E_0^j) \quad \text{by (24)} \tag{95}$$

$$= \prod_{j=1}^{J} P\left(\dot{\bigcap_{i=1}^{t}} E^j(i) \right) \quad \text{by (82)} \tag{96}$$

$$= \prod_{j=1}^{J} \prod_{i=1}^{t} P(E^j(i)) \quad \text{by (24)} \tag{97}$$

$$= \left[\prod_{j=1}^{J} p_j \right]^{t} \quad \text{by (83)} \tag{98}$$

Substituting these expressions for P_l into the inclusion-exclusion expansion (29) for (84) and using (78) we obtain

$$P_{B_1}(J\ m\ t) = \binom{m}{t} \sum_{l=0}^{m-t} (-1)^l \binom{m-t}{l} \left[\prod_{j=1}^{J} p_j \right]^{t+l} \tag{99}$$

Using the Binomial expansion in reverse to simplify this, we again obtain the Binomial probability (69) as required. As mentioned, this more laborious route to (69) is useful in unifying the present problem B_1 with B_2 of the next section. The two derivations are very similar, and many of the above results will be cited in solving B_2 below.

3.4.3.2. Under Model-2

Analogously to table 3-7 above, table 3-9 shows four alternative versions of our problem of finding the model-2 probability $P_\sigma^2(J\ f\ t\ \overline{X})$. Here, we solve the original problem A_2 of finding $P_\sigma^2(J\ f\ t\ \overline{X})$ by finding probability $P_{B_2}(J\ m\ t)$ of problem B_2. Probability $P_{B_2}(J\ m\ t)$ can be obtained by the analogue of the second method for finding $P_{B_1}(J\ m\ t)$ in the previous section. Of course, the sample space for problem B_2 is different than that for B_1. Paralleling (75) we now have

$$\Omega = \mathop{\times}_{j=1}^{J} \Omega^j \qquad \Omega^j = \binom{D_j}{S_j} \tag{100}$$

where, as defined in table 1-1, $\binom{D_j}{S_j}$ is the set of all subsets of D_j having size S_j. However, the events defined in table 3-8 remain relevant (although the "Defined for" column needs some adjustment, to correspond to precisely S_j integers now being selected in forming T_j.) Essentially all prior results apply except (81) and (82) and results dependent on these two. In place of these latter two equations, we have

$$E_i^j = \bigcap_{k=1}^{t} E^j(k) \bigcap E^j(i) \tag{101}$$

$$E_0^j = \bigcap_{k=1}^{t} E^j(k) \tag{102}$$

where the events being intersected are no longer mutually independent. Thus in place of (90) we have

$$P_l = \binom{m-t}{l} \prod_{j=1}^{J} P\left(\bigcap_{i=1}^{t+l} E^j(i)\right) \tag{103}$$

This requires the probability that the first $t+l$ integers belong to T_j. There are $\binom{M_j}{S_j}$ possible T_j corresponding to the ways to choose the required S_j integers from the M_j integers in set D_j. By assumption in B_2, these are all equally likely. Of these, $\binom{M_j-t-l}{S_j-t-l}$ contain the first $t+l$ integers since, having chosen the first $t+l$ integers to belong to T_j, there remains S_j-t-l to be selected (so that T_j has size S_j) from the remaining M_j-t-l integers of D_j. Thus

$$P_l = \binom{m-t}{l} \prod_{j=1}^{J} \frac{\binom{M_j-t-l}{S_j-t-l}}{\binom{M_j}{S_j}} \tag{104}$$

Similarly

$$P_0 = \prod_{j=1}^{J} \frac{\binom{M_j-t}{S_j-t}}{\binom{M_j}{S_j}} \tag{105}$$

Substituting these expressions for P_l into the inclusion-exclusion expansion (29) for (84) and using (78) we obtain

$$P_{B_2}(J\ m\ t) = \frac{\binom{m}{t} \sum_{l=0}^{m-t} (-1)^l \binom{m-t}{l} \prod_{j=1}^{J} \binom{M_j-t-l}{S_j-t-l}}{\prod_{j=1}^{J} \binom{M_j}{S_j}} \tag{106}$$

with special cases:

$$P_{B_2}(1\ m\ t) = \frac{\binom{m}{t}\binom{M_1-m}{S_1-t}}{\binom{M_1}{S_1}} \tag{107}$$

Table 3-9: Four Equivalent Problems

(A_2) **A Constraint Survival Problem:** Find the probability $P_\sigma^2(J\ f\ t\ \bar{X})$ of event $E(J\ f\ t\ \bar{X})$ of table 3-2 under probability model-2. Briefly, this is the probability that in a CLP small-class under model-2, a random instance allows exactly t values from the domain of variable f to each satisfy all the constraints indexed by J, when other argument variables take their values as in instantiation list $\bar{X}$.

(B_2) **A J-Urn Coincidence Problem:** From each set of integers $D_j = \{1\ 2\ 3\ ..\ M_j\}\ 1 \le j \le J$, select independently a subset T_j containing S_j integers. For a given D_j all subsets of size S_j are equally likely. Given a number $m \le \underset{1 \le j \le J}{\text{Min}} \{M_j\}$, what is the probability $P_{B_1}(J\ m\ t)$ that of the first m integers $\{1\ 2\ ..\ m\}$, exactly t are common to all J sets selected? Explicitly, we require the probability of the event

$$E(J\ m\ t) = \{\ (T_1\ T_2\ ..\ T_J) :\ |\ \bigcap_{j=1}^{J} T_j \bigcap J_1^m\ |\ = t\}$$

This is trivially equivalent to a J-urn problem where the sets D_j correspond to urns containing numbered balls. It is also clearly equivalent to having a "chess board" of J rows with the j-th row having M_j squares. The rows are aligned at the left, but not necessarily at the right. In each row j, exactly S_j balls are placed, each choice of S_j squares in row j being equally likely. What is the probability that exactly t of the first m columns are filled (i.e. have all J squares occupied by a ball)?

(C_2) **A Non-stationary Markov Chain Problem:** Let $P_{C_2}(j\ m\ t)$ be the probability of being in state t of the states $\{0\ 1\ 2\ ..\ m\}$ after exactly j steps of a non-stationary Markov chain

$$P_{C_2}(0\ m\ t) = \begin{cases} 1 & \text{if } t = m \\ 0 & \text{otherwise} \end{cases}$$

$$P_{C_2}(j\ m\ t) = \sum_{u=0}^{m} P_{C_2}(j\text{-}1\ m\ u)\, P(j\ u\ t) \qquad j \ge 1$$

having a nonstationary Hypergeometric transition probability $P(j\ u\ t)$ for going from state u to state t at step j

$$P(j\ u\ t) = \frac{\binom{M_j - u}{S_j - t}\binom{u}{t}}{\binom{M_j}{S_j}}$$

Note that $P(j\ u\ t) = 0$ for $t > u$. Find $P_{C_2}(J\ m\ t)$, the probability of being in state t after exactly J steps.

(D_2) **A Single-Urn J-Cycle Addition-Deletion Problem:** A single urn initially contains only m red balls. A sequence of J cycles is performed on the urn, each cycle consisting of an addition of a_j white balls to the urn, followed by a deletion of d_j balls from the urn. To correspond to our original problem we require a_j and d_j as follows, with all subsets of size d_j being equally likely to be deleted

$$a_j = \begin{cases} M_1 - m & \text{if } j = 1 \\ M_j - S_j - 1 & \text{if } 2 \le j \le J \end{cases}$$

$$d_j = M_j - S_j \qquad 1 \le j \le J$$

Find the probability $P_{D_2}(J\ m\ t)$ that, after J such addition-deletion cycles, there are exactly t red balls left in the urn.

$$P_{B_2}(J\ m\ m) = \frac{\prod_{j=1}^{J} \binom{M_j - t}{S_j - t}}{\prod_{j=1}^{J} \binom{M_j}{S_j}} \tag{108}$$

$$P_{B_2}(J\ m\ 0) = \frac{\sum_{l=0}^{m} (-1)^l \binom{m}{l} \prod_{j=1}^{J} \binom{M_j - l}{S_j - l}}{\prod_{j=1}^{J} \binom{M_j}{S_j}} \tag{109}$$

Unfortunately, unlike (99), (106) does not collapse in general. Only when $J = 1$ can the sum over l be removed using Vandermonde's identity to give (107). Note that (107) is a hypergeometric distribution while (106) for arbitrary J is not hypergeometric. Contrast this with the model-1 analogs (69) and (70) which are Binomial both for $J = 1$ and for arbitrary J. By the correspondence between problem B_2 and our original problem A_2, we have

---

$$P_\sigma^2(J\ f\ t\ \overline{X}) = \frac{\binom{m_f}{t} \sum_{l=0}^{m_f - t} (-1)^l \binom{m_f - t}{l} \prod_{j \epsilon J} \binom{M_j - t - l}{S_j - t - l}}{\prod_{j \epsilon J} \binom{M_j}{S_j}} \tag{110}$$

$$P_\sigma^2(J\ f\ \geq 1\ \overline{X}) = 1 - \frac{\sum_{l=0}^{m_f} (-1)^l \binom{m_f}{l} \prod_{j \epsilon J} \binom{M_j - l}{S_j - l}}{\prod_{j \epsilon J} \binom{M_j}{S_j}} \tag{111}$$

Result (111) follows from (110) by use of (45). Note that probabilities (110) and (111) are actually independent of the argument $\overline{X}$ but, unlike (73) and (74), they do depend on the sizes M_j. The model-2 probabilities of (60) to (62) were similar in this respect. Expression (111) is not at all as simple as the analogous expression (74) for the model-1 case. The latter is therefore simpler to use in numerical calculations. This is the reason that certain model-1 expectations, derived in [52], are useful alternatives to the model-2 versions even though the model-1 result is generally less exact as an instance-specific estimate.

It is interesting to note the close similarity between the two probabilities $P(J\ f\ t)$ of (73) and (110) for the model-2 and model-1 cases respectively. First note that

$$\begin{aligned}
\frac{\binom{a-c}{b-c}}{\binom{a}{b}} &= \frac{b(b-1)(b-2)\ldots(b-c+1)}{a(a-1)(a-2)\ldots(a-c+1)} \\
&= \frac{\frac{b}{a} \cdot (\frac{b}{a} - \frac{1}{a})(\frac{b}{a} - \frac{2}{a})\ldots(\frac{b}{a} - \frac{c}{a} + \frac{1}{a})}{1(1 - \frac{1}{a})(1 - \frac{2}{a})\ldots(1 - \frac{c}{a} + \frac{1}{a})} \\
&\to \left[\frac{b}{a}\right]^c \quad \text{for arbitrary fixed } b \text{ and } c\text{, as } a \to \infty
\end{aligned} \tag{112}$$

Making the identifications $a = M_j$, $b = S_j$ and $c = t+l$, we can then write for the model-2 probability (110) that, for arbitrary fixed S_j and m_f, as $M_j \to \infty$

$$\begin{aligned} P_\sigma^2(J\ f\ \ t\ \bar{X}) &\to \binom{m_f}{t} \sum_{l=0}^{m_f-t} (-1)^l \binom{m_f-t}{l} \prod_{j \in J} \left[\frac{S_j}{M_j} \right]^{t+l} \\ &= \binom{m_f}{t} \left[\prod_{j \in J} \frac{S_j}{M_j} \right]^t \left[1 - \prod_{j \in J} \frac{S_j}{M_j} \right]^{m_f-t} \\ &= P_\beta^1(J\ f\ \ t\ \bar{X}\ ;\ \mathbf{p}) \quad \text{for } \mathbf{p} = (p_1 \,.\, p_2 \,.\,.\, p_c) = (S_1/M_1 \,.\,.\, S_c/M_c) \end{aligned} \tag{113}$$

where we have employed the Binomial expansion in reverse to eliminate the sum over l. Thus $P(J\ f\ \ t\ \bar{X})$ under model-2 for a small-class $\text{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$ is asymptotically equal to a model-1 version for big-class $\text{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ when the parameter $\mathbf{p} = (p_1 \,.\,.\, p_c) = (S_1/M_1 \,.\,.\, S_c/M_c) = (R_1 \,.\,.\, R_c)$. This generalizes the more familiar result that the Binomial distribution can be arrived at as the limit of the Hypergeometric distribution. Note that in (51) and (61) the model-1 and model-2 versions of $P(j\ \bar{X})$ are respectively $P_\beta^1(j\ \bar{X}\ ;\ \mathbf{p}) = p_j$ and $P_\sigma^2(j\ \bar{X}) = R_j$ so that these two probabilities also become equal when $\mathbf{p}$ is as above with $p_j = R_j$; and this equality holds in general, not only in the above asymptotic limit. These equalities and near-equalities between corresponding probabilities will turn out to be the basis of our ability to approximate model-2 expected complexities, by their model-1 counterparts. This is further discussed in the next section.

3.5. Expected Values

3.5.1. Model-2 Expectations via Model-1

Models 1 and 2 are perhaps equally relevant (or irrelevant) as models of the way CLP instances are distributed in given real-word contexts. But model-2 expectations have the added virtue of being able to approximate exact-case complexities, as a consequence of the homogeneity of small-classes. Since big-classes are far from homogeneous, big-class expectations would appear not to be useful in this second sense. However, because of the asymptotic relation in (113), big-class expectations may in fact be used as approximations for small-class expectations and hence may be used to indirectly approximate exact-case values.

Why is this possibility relevant, seeing that the model-2 expectations have proven tractable? The reason is that the computation of numerical results is often far faster using model-0 and model-1 expressions than using those for model-2. This becomes particularly important when the derived expressions are used repeatedly within an algorithm to guide the problem-solving process (see the local heuristics developed in section 9 of [57]). Being easier to derive, model-0 and model-1 (in that order) expectations were in fact the first to be obtained — and their ability to approximate model-2 results would have been particularly important if the model-2 results had proven to actually be intractable. In different domains where the sought expectation *is* intractable, an analogous manner of approximating it may be applicable.

Let us now be explicit about how model-2 expectations are approximated by those for model-1. We first repeat from (32) and (33), the expressions for the expected value of a quantity $Q(clp)$ under models 1 and 2 respectively:

$$\bar{Q}_\beta^1(n\ \mathbf{m}\ c\ \mathbf{Z}\ ;\ \mathbf{p}) = \sum_{clp \,\in\, \text{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})} Q(clp)\, P_\beta^1(clp\ ;\ \mathbf{p}) \tag{114}$$

$$\bar{Q}_\sigma^2(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S}) = \sum_{clp \,\in\, \text{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})} Q(clp)\, P_\sigma^2(clp) \tag{115}$$

Probabilities $P_\beta^1(clp\ ;\ \mathbf{p})$ and $P_\sigma^2(clp)$ under the two models are given by (56) and (66) for the instance $clp = clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T})$. In the following we refer to a fixed big-class $\text{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ so that arguments n, $\mathbf{m}$, c and $\mathbf{Z}$ may be dropped where convenient. In particular we write the left sides of (114) and (115) as $\bar{Q}_\beta^1(\mathbf{p})$ and $\bar{Q}_\sigma^2(\mathbf{S})$ respectively and we abbreviate small-class $\text{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$ as $\text{CLP}_\sigma(\mathbf{S})$. The sum in (114) over the instances of the big-class $\text{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$ can be broken up into several sums over the instances of the small-classes $\text{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$ that are subsumed by the big-class — see section 2.5 — giving

Table 3-10: Probabilities of the Events in Table 3-2
(Combination of Tables 3-5 and 3-6)

Prob. Model	1	2
Ω	$\Omega_\beta = \mathrm{CLP}_\beta(n\ \mathbf{m}\ c\ \mathbf{Z})$	$\Omega_\sigma = \mathrm{CLP}_\sigma(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$
$P(E)$	$P_\beta^1(E\,;\mathbf{p})$	$P_\sigma^2(E)$
$P(j\ \bar{Z}_j)$	p_j	R_j
$P(j\ \bar{X})$	p_j	R_j
$P(J\ \bar{X})$	$\prod_{j \in J} p_j$	$\prod_{j \in J} R_j$
$P(J\ f\ \ t\ \bar{X})$	$\binom{m_f}{t} \left[\prod_{j \in J} p_j \right]^t \left[1 - \prod_{j \in J} p_j \right]^{m_f - t}$	$\dfrac{\binom{m_f}{t} \sum_{l=0}^{m_f - t} (-1)^l \binom{m_f - t}{l} \prod_{j \in J} \binom{M_j - t - l}{S_j - t - l}}{\prod_{j \in J} \binom{M_j}{S_j}}$
$P(J\ f\ \geq 1\ \bar{X})$	$1 - \left[1 - \prod_{j \in J} p_j \right]^{m_f}$	$1 - \dfrac{\sum_{l=0}^{m_f} (-1)^l \binom{m_f}{l} \prod_{j \in J} \binom{M_j - l}{S_j - l}}{\prod_{j \in J} \binom{M_j}{S_j}}$
$P(j\ T_j)$	$p_j^{S_j} (1 - p_j)^{M_j - S_j}$	$\binom{M_j}{S_j}^{-1}$
$P(\mathbf{T}) =$ $P(clp(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{T}))$	$\prod_{j=1}^{c} p_j^{S_j} (1 - p_j)^{M_j - S_j}$	$\prod_{j=1}^{c} \binom{M_j}{S_j}^{-1}$
$P(j\ S_j)$	$\binom{M_j}{S_j} p_j^{S_j} (1 - p_j)^{M_j - S_j}$	1.0
$P(\mathbf{S})$	$\prod_{j=1}^{c} \binom{M_j}{S_j} p_j^{S_j} (1 - p_j)^{M_j - S_j}$	1.0

$$\overline{Q}_{\beta}^{1}(\mathbf{p}) = \sum_{\mathbf{S}=(0\,\ldots\,0)}^{(M_1\,\ldots\,M_c)} \sum_{clp \in \mathrm{CLP}_{\sigma}(\mathbf{S})} Q(clp)\, P_{\beta}^{1}(clp\ ;\ \mathbf{p}) \tag{116}$$

Now by (56), (58) and (66) we have the following relationship between probabilities under models 1 and 2:

$$P_{\beta}^{1}(clp\ ;\ \mathbf{p}) = P_{\beta}^{1}(\mathbf{S}\ ;\ \mathbf{p})\, P_{\sigma}^{2}(clp) \tag{117}$$

where $\mathbf{S}$ is the satisfiability vector of instance clp. This allows (116) to be rewritten as

$$\overline{Q}_{\beta}^{1}(\mathbf{p}) = \sum_{\mathbf{S}=(0\,\ldots\,0)}^{(M_1\,\ldots\,M_c)} P_{\beta}^{1}(\mathbf{S}\ ;\ \mathbf{p}) \sum_{clp \in \mathrm{CLP}_{\sigma}(\mathbf{S})} Q(clp)\, P_{\sigma}^{2}(clp) \tag{118}$$

Using (115) we have finally

$$\overline{Q}_{\beta}^{1}(\mathbf{p}) = \sum_{\mathbf{S}=(0\,\ldots\,0)}^{(M_1\,\ldots\,M_c)} \overline{Q}_{\sigma}^{2}(\mathbf{S})\, P_{\beta}^{1}(\mathbf{S}\ ;\ \mathbf{p}) \tag{119}$$

which says that the expectation of $Q(clp)$ for a big-class under model-1 equals the sum of the expectations of $Q(clp)$ for each subsumed small-class weighted by the model-1 probability of the corresponding small-class.

We see then from (119) that a model-1 expectation is a weighted average of *many* model-2 expectations. However, from (58), the weighting probability $P_{\beta}^{1}(\mathbf{S}\ ;\ \mathbf{p})$ is a product of Binomial distributions and is sharply peaked about its modal vector $\mathbf{S} = \mathbf{S}^{mode}(\mathbf{p})$, which depends on the model-1 parameter vector $\mathbf{p}$. If the peaking of $P_{\beta}^{1}(\mathbf{S}\ ;\ \mathbf{p})$ about its mode is sufficiently sharp and if, away from the mode, $\overline{Q}_{\sigma}^{2}(\mathbf{S})$ does not grow so large as to offset its tiny weighting there, then the contribution of $\overline{Q}_{\sigma}^{2}(\mathbf{S})$ terms in (119) will be virtually zero at all $\mathbf{S}$ vectors except those close (in the Euclidean metric) to the mode vector $\mathbf{S}^{mode}(\mathbf{p})$. If also the $\overline{Q}_{\sigma}^{2}(\mathbf{S})$ terms in (119) for such near-modal $\mathbf{S}$ vectors are nearly equal then we might expect the following approximate equality to hold

$$\overline{Q}_{\beta}^{1}(\mathbf{p}) \approx \overline{Q}_{\sigma}^{2}(\mathbf{S}^{mode}(\mathbf{p})) \tag{120}$$

In this way, given a vector $\mathbf{p}$, the model-1 expectation over a big-class could be used to approximate the model-2 expectation over the subsumed small-class corresponding to the modal $\mathbf{S}$ vector induced by $\mathbf{p}$. Conversely, given an $\mathbf{S}$ vector characterizing a certain small-class, the model-2 expectation for that small-class could be approximated using the model-1 result for a $\mathbf{p}$ vector that makes $\mathbf{S}$ modal. The question then becomes: what $\mathbf{p}$ vector makes a given $\mathbf{S}$ modal in (58)? Generalizing Feller's result for an individual Binomial distribution, page 151 [18], we propose using for the multi-dimensional Binomial case[23]

$$\mathbf{p} = \mathbf{R} = (R_1\, R_2\, \ldots\, R_c) = \left(\frac{S_1}{M_1}\ \frac{S_2}{M_2}\ \ldots\ \frac{S_c}{M_c}\right) \tag{121}$$

With this value for $\mathbf{p}$, the model-1 expectation $\overline{Q}_{\beta}^{1}(n\ \mathbf{m}\ c\ \mathbf{Z}\ ;\ \mathbf{p})$ over big-class $\mathrm{CLP}_{\beta}(n\ \mathbf{m}\ c\ \mathbf{Z})$ should provide a useful approximation for the model-2 expectation $\overline{Q}_{\sigma}^{2}(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$ over any subsumed small-class $\mathrm{CLP}_{\sigma}(n\ \mathbf{m}\ c\ \mathbf{Z}\ \mathbf{S})$. Of course, this approximating ability is contingent on the validity of the assumptions made above regarding the nature of the quantity $Q(clp)$ being averaged. Empirically, these assumptions have been found to hold for the several clp-dependent quantities to be studied here. The model-1 and model-2 expected number of solutions $\overline{S}$ derived in the next section are in fact always exactly

[23] Note that this is also the maximum-likelihood estimate for $\mathbf{p}$ based on a single instance of satisfiability vector $\mathbf{S}$ arising under model-1 in $\mathrm{CLP}_{\beta}(n\ \mathbf{m}\ c\ \mathbf{Z})$. This however is not relevant here since we are not attempting to estimate the distribution parameter $\mathbf{p}$ of any real model-1 distribution. Model-1 is being used here merely as a technical device for estimating model-2 expectations.

equal when using (121). So also are all the model-1 and corresponding model-2 results for gBT derived in [52]. Exact equality however does not hold for gFC model-1 and model-2 results, although agreement is usually excellent.

Note, that exact equality will always hold no matter what the quantity $Q(clp)$, in the special case of small-classes all of whose satisfiabilities S_j equal 0 or M_j. (Such small-classes contain only a single CLP instance.) That exact equality then holds in (120) can be seen from (58) and (119) and the fact that (121) causes all p_j parameter values in model-1 to equal 0 or 1.

3.5.2. The Expected Number of Solutions

This section presents our first application of the above probability models in finding an expected value — the expectation $\bar{S}$ of the number $S(clp)$ of solutions of a CLP instance. This expectation is particularly easy to derive, since it is algorithm-independent. For this reason $\bar{S}$ is derived in this section, as opposed to in [52] where algorithm-dependent expectations are obtained. In spite of its relative simplicity, or perhaps because of it, the derivation here of $\bar{S}$ is important as a model for these later, more complicated derivations.

It should be noted that much of the reasoning in obtaining the expectations of [52] will be equally valid for all three probability models. A single unified derivation can therefore be used, at least up to the point where model-dependent differences can no longer be ignored. At this point the unified approach in split into separate streams. An example of this is seen in this section.

Let $S(clp)$ denote the number of solutions that exist for a given CLP instance clp. Specifically, it is the number of labelings $\bar{Z} \epsilon D$ of all n problem variables in Z, such that $\bar{Z}$ satisfies all the m problem constraints. The set $E(J\ \bar{X})$ of table 3-2 is useful here. It is the set of all instances for which value-tuple $\bar{X}$ satisfies all the constraints indexed by the set J. Thus, with $J = J_1^c$ and $\bar{X} = \bar{Z}$, we have that $E(J_1^c\ \bar{Z})$ is the set of all instances having $\bar{Z}$ as a solution. Letting $\delta(J_1^c\ \bar{Z}\ clp)$ be the characteristic function of this set, we can then express $S(clp)$ as

$$S(clp) = \sum_{\bar{Z} \epsilon D} \delta(J_1^c\ \bar{Z}\ clp) \tag{122}$$

For a fixed labeling $\bar{Z}$, summing the value of the characteristic function over all instances $clp \ \epsilon\ \Omega$ gives the number of instances for which $\bar{Z}$ is a solution. On the other hand, for a particular instance clp, summing over all possible labelings $\bar{Z} \epsilon D$, as in (122), gives the number of solutions for that instance. The expected number $\bar{S}$ of solutions is then found as follows

$$\bar{S} = \sum_{clp\ \epsilon\ \Omega} S(clp)\, P(clp) \quad \text{by } (25) \tag{123}$$

$$= \sum_{clp \epsilon \Omega} \sum_{\bar{Z} \epsilon D} \delta(J_1^c\ \bar{Z}\ clp)\, P(clp) \quad \text{by } (122) \tag{124}$$

$$= \sum_{\bar{Z} \epsilon D} \sum_{clp \epsilon \Omega} \delta(J_1^c\ \bar{Z}\ clp)\, P(clp) \tag{125}$$

$$= \sum_{\bar{Z} \epsilon D} P(J_1^c\ \bar{Z}) \quad \text{by } (27) \tag{126}$$

$$= \sum_{\bar{Z} \epsilon D} \prod_{j=1}^{c} P(j\ \bar{Z}) \quad \text{by } (44) \tag{127}$$

The above derivation has been equally applicable for obtaining $\bar{S}$ under either model-1 or model-2. This model-independence has been taken as far as possible and we now specialize to each case individually. Probability $P(j\ \bar{Z})$ under the models 1 and 2 is given by (51) and (61) respectively, with $\bar{X} = \bar{Z}$. In either case, the probability is independent of $\bar{Z}$. Thus the product in (127) may taken outside the sum, giving respectively the final results

$$\bar{S}_\beta^1 = \Big(\prod_{z_i \in Z} m_{z_i}\Big) \prod_{j=1}^{c} p_j \quad \text{by (51)} \tag{128}$$

$$\bar{S}_\sigma^2 = \Big(\prod_{z_i \in Z} m_{z_i}\Big) \prod_{j=1}^{c} R_j \quad \text{by (61)} \tag{129}$$

where we have collapsed the sum over $\bar{Z}$ by use of

$$\sum_{\bar{Z} \in D} 1 = |\ D\ | = |\ \bigtimes_{z_i \in Z} d_{z_i}\ | = \prod_{z_i \in Z} |\ d_{z_i}\ | = \prod_{z_i \in Z} m_{z_i} \tag{130}$$

Note that the above two expectations (128) and (129) provide a good example of the phenomenon discussed in the previous section. By setting each p_j of the model-1 parameter vector **p** equal to the corresponding $R_j = S_j / M_j$, a model-1 expectation over big-class CLP$_\beta$(n **m** c **Z**) may provide a good approximation for the corresponding model-2 expectation over small-class CLP$_\sigma$(n **m** c **Z S**). In the present case, the approximation is in fact perfect, the two expectations being always exactly the same when $p_j = R_j$. Although this will not always be the case for expectations of other quantities, we have found that the approximation is usually excellent. For some expectations found in [52], the asymptotic equality (113) will be found to help in this regard.

3.5.3. Expected Number of Surviving Domain Values

One of the probabilities derived earlier was the probability $P(J\ f\ t\ \bar{X})$ for event $E(J\ f\ t\ \bar{X})$ of table 3-2. This was in fact the most complex probability derived to date. It will prove very useful in [52] where expected complexities of CLP problem-solving are derived. In that context, we will also require a simplified expression for $\sum_{0 \le t \le m_f} t\ P(J\ f\ t\ \bar{X})$ — the expected value of t under distribution $P(J\ f\ t\ \bar{X})$. This expectation can be re-expressed in the form of (25) by introducing the instance-dependent quantity $t(J\ f\ \bar{X}\ clp)$ which is, for the given instance clp, the number of values from the domain of variable f that survive all the constraints indexed by set J when the other argument variables take their corresponding values as in $\bar{X}$. It is the expectation of this quantity that is expressed above and we have that

$$\bar{t}(J\ f\ \bar{X}) = \sum_{t=0}^{m_f} t\ P(J\ f\ t\ \bar{X}) = \sum_{clp \in \Omega} t(J\ f\ \bar{X}\ clp)\ P(clp) \tag{131}$$

The values under models 1 and 2 are obtained by using the corresponding versions of $P(J\ f\ t\ \bar{X})$ or $P(clp)$. In the present case it is simplest to work with $P(J\ f\ t\ \bar{X})$. Under model-1 this is given by (73) which is just the Binomial probability for the number t of successes in m_f Bernoulli trials of success rate $\prod_{j \in J} p_j$. The expected value of t is thus just the usual product of the number of trials by the success rate, so we have that the required expectation in (131) is

$$\sum_{t=0}^{m_f} t\ P_\beta^1(J\ f\ t\ \bar{X}\ ;\ \mathbf{p}) = m_f \prod_{j \in J} p_j = m_f\ P_\beta^1(J\ \bar{X}\ ;\ \mathbf{p}) \tag{132}$$

Under model-2 we obtain the expectation in (131) by using probability $P_\sigma^2(J\ f\ t\ \bar{X})$ of (110). Evaluating (131) for this distribution seems rather forbiding. However there is a convenient shortcut. Using the identity $\binom{a}{b} = \frac{a}{b}\binom{a-1}{b-1}$ one can readily show that

$$\sum_{t=0}^{m_f} t\ P_\sigma^2(J\ f\ t\ \bar{X}) = m_f\ \Big[\prod_{j \in J} \frac{S_j}{M_j}\Big] \sum_{t'=0}^{m_{f'}} P_\sigma^2(J'\ f'\ t'\ \bar{X}) \tag{133}$$

where f' is a variable with domain size $m_{f'} = m_f - 1$, J' indexes $|\ J\ |$ constraints of size $M_j - 1$ from the j-th of which we choose a subset of size S_j-1. Now since $P_\sigma^2(J'\ f'\ t'\ \overline{X})$ is a probability distribution for t', we have $\sum_{t'=0}^{m_{f'}} P_\sigma^2(J'\ f'\ t'\ \overline{X}) = 1$ and therefore

$$\sum_{t=0}^{m_f} t\, P_\sigma^2(J\ f\ t\ \overline{X}) = m_f \prod_{j \epsilon J} R_j = m_f\ P_\sigma^2(J\ \overline{X}) \tag{134}$$

APPENDIX A
ALTERNATIVE CHARACTERIZATIONS of the GOAL of a CLP INSTANCE

This appendix gives several alternative characterizations to that in (5) for the solution set T, which is the goal sought in solving a CLP instance.

Characterization 2

Note that the solution set T may be considered as the relation induced by a constraint $C = (Z\ T)$ on the full set Z of problem variables — and it is for this reason that we use the symbol T to denote the solution set, paralleling the use of T_j for the relations of the original problem constraints $C_j = (Z_j\ \ T_j)$. Since the value-tuples in T are precisely those that satisfy all c instance constraints, this induced constraint $C = (Z\ T)$ is just the canonical form for the logical conjunction of the c instance constraints

$$C = \bigwedge_{j=1}^{c} C_j \,.$$

Finding the solution set T of a CLP instance thus amounts to synthesizing the induced conjunctive constraint C.

Characterization 3

If for each constraint C_j we define its **extension** onto Z as $C_j^e = (Z\ T_j^e)$ where $T_j^e = \{\ \bar{Z} : \bar{Z}_j(\bar{Z}) \in T_j\ \}$, then the solution set is given by

$$T = \bigcap_{j=1}^{c} T_j^e$$

The sets T_j^e are "prisms" with cross-section T_j. The solution set T is thus the intersection of the c prisms whose cross-sections are the problem constraint-relations T_j.

Characterization 4

Any set can be characterized as the union of all its subsets. This is useful here because the subsets of the solution set T can themselves be described independently of T, in terms of the givens of the problem. Specifically, if we denote by $\pi_j(A)$ the projection onto Z_j of a set $A \subseteq D$, so that

$$\pi_j(A) = \{\bar{Z}_j(\bar{Z})\ \mid\ \bar{Z} \in A\}$$

then the subsets of T can be characterized by

$$A \subseteq T \ \text{ iff } \pi_j(A) \subseteq T_j \quad \forall\, j \in J_1^c$$

The solution set can then be expressed as the union of all its subsets, as follows

$$T = \bigcup_{A \text{ s.t. } \pi_j(A) \subseteq T_j\ \forall j \in J_1^c} A = \operatorname*{Sup}_{A \text{ s.t. } \pi_j(A) \subseteq T_j\ \forall j \in J_1^c} (A)$$

The solution set is thus the largest subset of D whose projections onto the various Z_j fall inside the regions given by the corresponding T_j. This and characterization 3 above correspond closely to the "volume-reconstruction" problem mentioned in section 1, in connection with figure 1-1.

Characterization 5

Conversely to the previous approach, any set may also be characterized as the intersection of all its supersets. Thus we have that

$$T = \bigcap_{A \supseteq T} A = \operatorname*{Inf}_{A \supseteq T} (A)$$

This however is not particularly useful since the supersets of T are here expressed directly in terms of T itself. There does not appear to be a simple characterization of these supersets in terms of the givens of the problem, as there was above for the subsets of T.

Characterization 6

For those familiar with the "join" operator $\bowtie$ of relational database theory [48], the solution set T may be directly characterized as

$$T = \bowtie_{j=1}^{c} T_j$$

(where we assume that $Z = \bigcup_{1 \leq j \leq c} Z_j$). This is further discussed in chapter 7 of [51] in dealing with database applications of CLP.

REFERENCES

[1] Ball W. W. R., *Mathematical Recreations and Essays, 11-th ed.,* Macmillan, New York, 1960.

[2] Barrow H. G. and Tenenbaum J. M., "MYSYS: A system for reasoning about scenes," Report TR-121, SRI International, March, 1976.

[3] Bernard S. R., "An urn model study of variability within a compartment," *Bulletin Mathematical Biology,* vol. 39, pp. 463-470, 1977.

[4] Bitner J. R., "Backtrack programming techniques," *Communications ACM,* vol. 18, pp. 651-656, 1975.

[5] Borning A., *Thinglab - A constraint-oriented simulation laboratory,* Stanford University, 1979, Ph.D. dissertation. A preliminary description appears in 5-th Proc. Int. Joint Conf. Artificial Intelligence, 1977.

[6] Brooks R. A., "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intelligence,* vol. 17, pp. 285-348, 1981.

[7] Brown C. A. and Purdom P. W. Jr., "How to search efficiently," *Proc 7-th International Joint Conf. on Artificial Intelligence,* pp. 588-594, Vancouver, B.C., Canada, August 1981.

[8] Brown C. A. and Purdom P. W. Jr., "An average time analysis of backtracking," *SIAM J. Computing,* vol. 10, pp. 583-593, 1981.

[9] Brown C. A. and Purdom P. W. Jr., "An empirical comparison of backtracking algorithms," *I.E.E.E. Trans. Pattern Analysis and Machine Intelligence,* vol. PAMI-4, pp. 309-316, 1982.

[10] Brown T., "A note on 'Instant Insanity'," *Mathematics Magazine,* vol. 41, no. 4, pp. 167-169, 1968.

[11] Burstall R. M., "A program for solving word sum puzzles," *Computer Journal,* vol. 12, pp. 48-51, 1969.

[12] Cook S. A., "The complexity of theorem proving procedures," *Proc. 3-rd Annual ACM Symp. on Theory of Computing,* pp. 151-158, New York, 1971.

[13] Davis L. S. and Rosenfeld A., "Hierarchical relaxation for waveform parsing," in *Computer Vision Systems,* Hanson and Riseman, Ed Acadamic Press, New York, pp. 101-109, 1978.

[14] Davis L. S. and Rosenfeld A., "Cooperating processes for low-level vision: a survey," *Artificial Intelligence (Special Issue on Computer Vision),* vol. 17, pp. 245-263, 1981.

[15] Davis L. S. and Thomas C. H., "Hierarchical constraint processes for shape analysis," *I.E.E.E. Trans. Pattern Analysis and Machine Intelligence,* vol. PAMI-3, no. 3, pp. 265-277, 1981.

[16] Deutsch J. P. A., "A short cut for certain combinatorial problems," *British Joint Computer Conference*, 1966.

[17] Faugeras O. D. and Berthod M., "Improving consistency and reducing ambiguity in stochastic labeling: An optimization approach," *I.E.E.E. Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-3, no. 4, pp. 412-424, 1981.

[18] Feller W., *An Introduction to Probability Theory and Its Applications*, Wiley, New York, 1968.

[19] Fikes R. E., "REF-ARF: A system for solving problems stated as procedures," *Artificial Intelligence*, vol. 1, pp. 27-120, 1970.

[20] Fillmore J. and Williamson S., "On Backtracking: A combinatorial description of the algorithm," *SIAM J. Computing*, vol. 3, no. 1, pp. 41-55, 1974.

[21] Floyd R., "Nondeterministic algorithms," *J. ACM*, vol. 14, pp. 636-644, 1967.

[22] Franco J. and Paull M., "Probabilistic analysis of the Davis Putnam procedure for solving the Satisfiability problem," Report CES-81-3, Case western Reserve University, June. 1981.

[23] Freuder E. C., "Synthesizing constraint expressions," *Comm. ACM*, vol. 21, pp. 958-966, 1978.

[24] Garey M. R. and Johnson D. S., *Computers and Intractability*, Freeman, San Francisco, 1979.

[25] Gaschnig J., "A constraint satisfaction method for inference making," *Proc. 12-th Annual Allerton Conf. on Circuit System Theory*, pp. 866-874, U. Illinois, 1974.

[26] Gaschnig J., *Performance Measurement and Analysis of Certain Search Algorithms*, Dept. Computer Science, Carnegie-Mellon University, May 1979, Ph.D. dissertation.

[27] Gibbons G. D., "POPS: An application of heuristic serach methods to the processing of a nondeterministic programming language," *Proc. 3-rd International Joint Conf. on Artificial Intelligence.*, Stanford U., Stanford, California., 1973.

[28] Goldberg A., "Average case complexity of the satisfiability problem," *Proc. 4-th Workshop on Automated Deduction*, pp. 1-6, Austin, Texas, 1979.

[29] Golomb S. W. and Baumert L. D., "Backtrack programming," *J. ACM*, vol. 12, pp. 516-524, 1965.

[30] Grimson W. E. L., "The combinatorics of local constraints in model-based recognition and localization from sparse data," *JACM*, 1986, To appear.

[31] Grossman R. W., "Some data base applications of constraint expressions," Report TR-158, Lab. Computer Science, M.I.T., Feb. 1976, M.Sc. dissertation.

[32] Hammersley J. M. and Handscomb D. C., *Monte Carlo Methods*, Wiley, New York, 1964.

[33] Haralick R. M., Davis L. S., and Rosenfeld A., "Reduction Operations for Constraint Satisfaction," *Information Sciences*, vol. 14, pp. 199-219, 1978.

[34] Haralick R. M. and Shapiro. L. G., "The consistent labeling problem: part I," *I.E.E.E. Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 173-184, 1979.

[35] Haralick R. M., "Scene matching methods," in *Issues in Digital Image Processing*, Haralick, R. M. and Simon J. C., Ed Sijthoff and Noordhoff, Alphen aan den Rijn, Netherlands, pp. 221-243, 1980.

[36] Haralick R. M. and Elliot G. L., "Increasing tree search efficiency for constraint satisfaction problems," *Artificial Intelligence*, vol. 14, pp. 263-313, 1980.

[37] Haralick R. M. and Shapiro. L. G., "The consistent labeling problem: part II," *I.E.E.E. Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 3, pp. 193-203, 1980.

[38] Hardy G. H. and Littlewood J. E., "Some problems of Diophantine aproximation: the lattice points of a right-angled triangle," *Proc. London Mathematical Soc.*, vol. 20, pp. 15-36, 1920.

[39] Hayes K. C., Jr., "Reading handwritten words using hierarchical relaxation," *Computer Graphics and Image Processing*, vol. 14, pp. 344-364, 1980.

[40] Johnson N. L. and Kotz S., *Urn Models and Their Applications: An Approach to Modern Discrete Probability Theory*, Wiley, New York, 1977.

[41] Knuth D. E., "Estimating the efficiency of Backtrack programs," *Mathematics of Computation*, vol. 29, no. 129, pp. 121-136, January, 1975.

[42] Kowalski R., "A proof procedure using connection graphs," *J. ACM*, vol. 22, no. 4, pp. 572-595, 1975.

[43] Kowalski R., *Logic for Problem Solving*, North Holland, New York, 1979.

[44] Lauriere J., "A language and a program for stating and solving combinatorial problems," *Artificial Intelligence*, vol. 10, pp. 29-127, 1978.

[45] Lieberherr K., "Algorithmic extremal problems in combinatorial optimization," *Journal of Algorithms*, vol. 3, no. 3, pp. 225-244, 1982.

[46] Mackworth A. K., "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, pp. 99-118, 1977.

[47] Mackworth A. K., "On reading sketch maps," *Proc. 5-th Int. Joint Conf. on Artificial Intelligence*, pp. 598-606, M.I.T., Cambridge, Mass., August, 1977.

[48] Maier D., *The Theory of Relational Databases*, Computer Science Press, Rockville, Maryland, 1983.

[49] McGregor J. J., "Relational consistency algorithms and their application in finding subgraph and graph isomorphisms," *Information Sciences*, vol. 19, pp. 229-250, 1979.

[50] Montanari U., "Networks of constraints: Fundamental properties and applications to picture processing," *Information Sciences*, vol. 7, pp. 95-132, 1974.

[51] Nadel B. A., *The Consistent Labeling Problem and its Algorithms: Towards Exact-Complexities and Theory-Based Heuristics,* Computer Science Dept., Rutgers University, New Brunswick, N.J., 1986, Ph.D. dissertation, to appear.

[52] Nadel B. A., "Three consistent labeling algorithms and their complexities: search-order dependent and effectively instance-specific results," Report DCS-TR-171, Computer Science Dept., Rutgers University, New Brunswick, N.J., 1986, Also appears as Report CRL-TR-3-86, Dept. Electrical Engineering and Computer Science, U. of Michigan, Ann Arbor, MI, 1986.

[53] Newell A. and Simon H. A., *Human Problem Solving,* Prentice-Hall, Englewood Cliffs, N.J., 1972.

[54] Nijenhuis A. and Wilf H. S., *Combinatorial Algorithms,* Academic Press, New York, 1975.

[55] Nudel B. A., "Improved constraint satisfaction algorithms using inter-variable compatibilities," Report DCS-TR-109, Computer Science Dept., Rutgers University, New Brunswick, N.J., 1981.

[56] Nudel B. A., "Consistent Labeling Problems and their algorithms," *Proc Nat. Conf. on Artificial Intelligence (AAAI),* pp. 128-132, Pittsburgh, August 1982.

[57] Nudel B. A., "Consistent-labeling problems and their algorithms: expected-complexities and theory based heuristics," *Artificial Intelligence (Special Issue on Search and Heuristics),* vol. 21, no. 1 and 2, pp. 135-178, March 1983, Also in book: Search and Heuristics, North-Holland, Amsterdam 1983.

[58] Nudel B. A., "Solving the general consistent labeling (or constraint satisfaction) problem: Two algorithms and their expected complexities," *Proc Nat. Conf. on Artificial Intelligence (AAAI),* pp. 292-296, Washigton D.C., August 1983, Also available as report DCS-TR-128, Computer Science Dept., Rutgers University, New Brunswick, N.J., 1983.

[59] Papoulis A., *Probability, Random Variables and Stochastic Processes,* McGraw-Hill, New York, 1965.

[60] Peleg S., "Ambiguity reduction in handwriting with ambiguous segmentation and uncertain interpretation," *Computer Graphics and Image Processing,* vol. 10, pp. 235-245, 1979.

[61] Peleg S. and Rosenfeld A., "Breaking substitution ciphers using a relaxation algorithm," *Communications ACM,* vol. 22, pp. 598-605, 1979.

[62] Peleg S., "A new probabilistic relaxation scheme," *I.E.E.E. Trans. Pattern Analysis and Machine Intelligence,* vol. PAMI-2, no. 4, pp. 362-369, 1980.

[63] Polya G., *Mathematical Discovery, volume 1.,* Wiley, New York, 1962.

[64] Reingold E. M., Nievergelt J., and Deo N., *Combinatorial Algorithms,* Prentice Hall, New Jersey, 1977.

[65] Rosenfeld A., Hummel R. A., and Zucker S. W., "Scene labeling by relaxation operations," *I.E.E.E. Trans. Systems, Man and Cybernetics,* vol. SMC-6, no. 6, pp. 420-433, 1976.

[66] Rutkowski W. S., Peleg S., and Rosenfeld A., "Shape segmentation using relaxation," *I.E.E.E. Trans. Pattern Analysis and Machine Intelligence,* vol. PAMI-3, no. 4, pp. 368-375, 1981.

[67] Schaefer T. J., "The complexity of satisfiability problems," *Proc. 10-th Annual ACM Symp. on the Theory of Computing*, pp. 216-226, 1978.

[68] Shapiro L. G. and Haralick R. M., "Structural Descriptions and Inexact Matching," *I.E.E.E. Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-3, no. 5, pp. 504-519, 1981.

[69] Shenton L. R., "A reinforcement-depletion urn problem - I. Basic theory," *Bulletin Mathematical Biology*, vol. 43, no. 3, pp. 327-340, 1981.

[70] Stallman R. M. and Sussman G. J., "Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis," *Artificial Intelligence*, vol. 9, pp. 135-196, 1977.

[71] Steele G., "The definition and implementation of a computer language based on constraints," AI TR-595, M.I.T., 1980.

[72] Steels L., "The constraint machine," Research AI-memo 1, Schlumberger-Doll, 1980.

[73] Steels L., "Constraints as consultants," in *Progress in Artificial Intelligence. Selected and updated papers from the proceedings of the 1982 European conference (Orsay, France, 1982)*, Steels L. and Campbell J., Ed Wiley, pp. 146-165, New York, 1985.

[74] Sussman G. J. and Steele G. L. Jr., "CONSTRAINTS - a language for expressing almost-heirarchical descriptions," *Artificial Intelligence*, vol. 14, pp. 1-39, 1980.

[75] Tucker A., *Applied Combinatorics*, Wiley, New York, 1980.

[76] Ullman S., "Relaxation and constrained optimization by local processes," *Computer Graphics and Image Processing*, vol. 10, pp. 115-125, 1979.

[77] Walker R. J., "An enumerative technique for a class of combinatorial problems," in *Combinatorial Analysis (Proc. Symp. Applied Math., Vol. X)* American Mathematical Society, Providence, R. I., 1960.

[78] Waltz D., "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*, Winston P. H., Ed McGraw-Hill, New York, 1975.

[79] Winston P. H., *Artificial Intelligence*, Addison-Wesley, Reading, Mass, 1977.

UNIVERSITY OF MICHIGAN
3 9015 09911 4897

PLEASE RETURN TO
COMPUTER SCIENCE DEPARTMENT ARCHIVES
3440 BOELTER HALL

0123456 A4 Page 6543210

AIIM SCANNER TEST CHART # 2

Spectra

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Times Roman

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Century Schoolbook Bold

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

News Gothic Bold Reversed

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Bodoni Italic

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Greek and Math Symbols

4 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋫⋪⋧⋜≡

6 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋫⋪⋧⋜≡

8 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋫⋪⋧⋜≡

10 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋫⋪⋧⋜≡

White Black

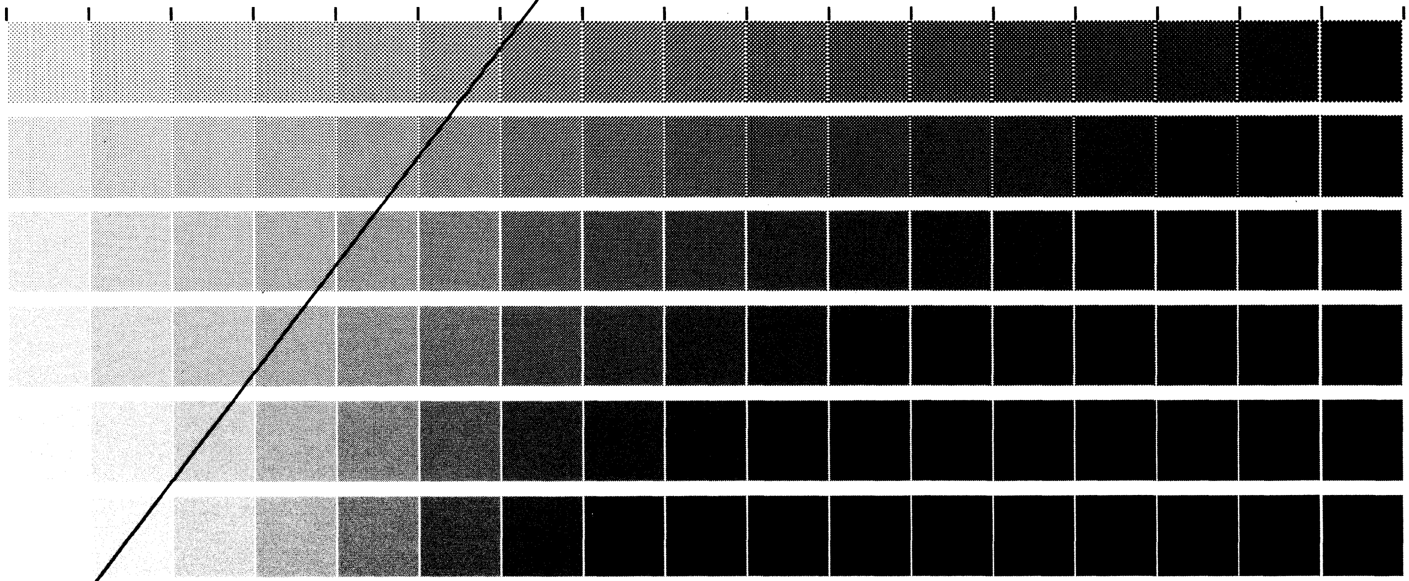
Isolated Characters

e	m	1	2	3	a
4	5	6	7	o	°
8	9	0	h	l	B

0123456 6543210

A4 Page 6543210

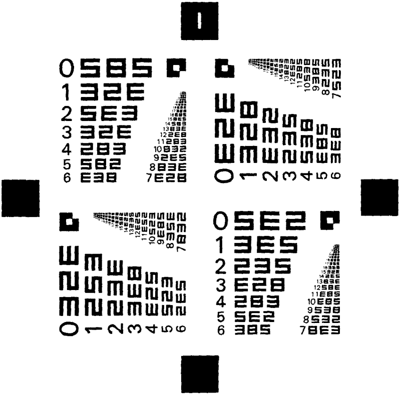
MESH HALFTONE WEDGES



A4 Page 6543210

0123456 6543210

RIT ALPHANUMERIC RESOLUTION TEST OBJECT, RT-1-71



PRODUCED BY GRAPHIC ARTS RESEARCH CENTER

ROCHESTER INSTITUTE OF TECHNOLOGY, ONE LOMB

MEMORIAL DRIVE, ROCHESTER, NEW YORK 14623