Disproof of Arrow's Impossibility Theorem

by

John Clifton Lawrence
General Algorithm
POB 230351
Encinitas, CA 92023
Phone/fax: 760-633-3778
Web site: http://www.genalg.com
E-mail: jlawrence@genalg.com

July 15, 1998

Abstract

Arrow’s Social Choice Impossibility Theorem is disproved by demonstrating that Arrow’s treatment of tie situations was incorrect. Invalidating Arrow’s proof in itself does not prove that Social Choice is possible. The possibility of Social Choice is proven by presenting an algorithm which represents a social welfare function that maps the domain of all possible combinations of individual choices into corresponding social choices. It is proven that the algorithm produces correct solutions for any number of alternatives and any number of voters which meet Arrow’s criteria.

Key Words: social choice, Arrow, Condorcet, voting, paradox of voting, algorithm, social welfare function, impossibility theorem

List of Symbols

x^1

R_i

aR_ib

x_i^k

aP_ib

$N(a,b)$

aTb

aIb

$\rho(A)$

Z_1^1 (**abcd**)

$X[b_1^p]^1$

Introduction

In 1785 The Marquis de Condorcet (Granger [1989], McLean and Hewitt [1994]) published his Essai (1785) in which he pointed out the problems associated with an election in which there are three or more candidates. This has become known as the paradox of voting. Condorcet and the American President, Thomas Jefferson, were collaborators in the production of both the French and American Constitutions. Condorcet lost his life in the French Revolution because he left his secure hiding place when he learned that his host was subject to the death penalty for harboring him. He was preceded in the theory of elections by a few years by his friend Jean-Charles de Borda (1781) who proposed the rank-order count method of voting. The French Enlightenment philosophers hoped to “carry the methods of rigorous and mathematical thought beyond the physical and into the realms of the human sciences.” (Black[1958])

Over the years there have been various writers that have contributed to the theory of elections including E. J. Nanson (1907) and the Reverend C. L. Dodgson (Lewis Carroll) (1873, 1874, 1876). In 1951 Nobel Laureate Kenneth J. Arrow published *Social Choice and Individual Value*s in which he explored the question of whether or not individual preferences could be aggregated in some rational way in order to form a social choice. He postulated five rational and ethical criteria and two axioms that such a social welfare function should meet, and then proceeded to prove that no such social welfare function existed. This theorem is known as Arrow's Impossibility Theorem, and an impressive literature concerning itself with what has come to be known as Social Choice theory has developed in the last   forty -six years. At least one author considers that Arrow's Theorem “has a good claim to be considered the outstanding problem in the philosophy of economics” (MacKay [1980]).

Some of the literature has been concerned with finding a way around Arrow's basic result that no rational social choice is possible by relaxing one or more of his criteria (Sen [1970], Riley [1988], Murakami [1968]). Arrow's theorem has important political, economic and social implications since, if indeed no rational way to aggregate individual preferences is possible and Pareto optimality is the best that can be achieved, then a populist democracy which closely reflects the will of the people becomes impossible and free market capitalism acquires a theoretically endorsed superiority over any kind of populist socialistic or democratic economic system. Liberal or Madisonian democracy in which the purpose of voting is just to elect leaders and lawmakers becomes all that is attainable while populist or direct democracy in which social policies are decided upon directly by voting becomes theoretically unfeasible. The notion of electronic democracy in which voters vote directly on issues from computer terminals and then supercomputers tally the results [what might be called an Information Age Utopia] is not theoretically acceptable. These realizations have produced pessimism and even nihilism among proponents of welfare economics (Bergson, 1966). However, advocates of democratic voting systems should be equally concerned as Arrow's result tarnishes the validity of democratic elections as well (Riker [1982], Schofield [1985]).

In this paper we will present an algorithm which provides a solution for the social choice problem for any number of alternatives without diluting Arrow's five criteria and two axioms for social choice. In fact we strengthen them considerably. We also give a more rigorous statement of those criteria. We prove that the algorithm works for all values of m, the number of alternatives and for any number of voters, n. Our method is ordinal (rather than cardinal), based on pair-wise comparisons and independent of irrelevant alternatives. It is shown in this paper that the key to opening the door of social choice is the proper consideration of tie solutions.

Notation

We follow conventional notation. Let us assume that we have a society composed of n voters. For identification purposes, we can number them from 1 to n, $1< i \leq n$. We will refer to the ith individual. We assume an alternative set, S, consisting of m alternatives: a, b, c... . Let the set $\{x^1, x^2 ... x^m\}$ consist of some permutation of the alternative set {a,b,c...}. Arrow uses an R notation, which we will follow, which means "is preferred or is indifferent to." To indicate that voter i prefers a to b or is indifferent between a and b, we would write aR_ib. We assume that each voter has a "preference or indifference" relationship, R_i, over the alternative set as follows:

$$R_i = x_i^1 R_i x_i^2 R_i ... x_i^{m-1} R_i \, x_i^m$$

where

x_i^k represents the kth "preference or indifference" of the ith voter.

We will use a shorthand notation as follows: abcd for $aR_ib \, R_ic \, R_id$.

The Social Welfare Function

A function is a mapping from a set of elements called the domain to a set of elements called the range in such a way that each element of the domain is connected with not more than one element of the range. Now the mapping from domain to range can be in such a way that for every element of the range there is at most one corresponding element of the domain **(one-to-one or injective)**; for every element of the range there is one or more corresponding elements of the domain **(onto or surjective)** or for every element of the range there is one and only one corresponding element of the domain **(one-to-one correspondence or bijective)**.

The **Social Welfare Function (SWF)** maps the domain which consists of all possible combinations of R_i, $1\leq i \leq n$, votes onto the range, each element of which is a possible social "preference or indifference" relationship, R, which is one of the set of relationships, R_i, available to the individual voter. The domain can be represented as the set of all possible combinations
$\{ R_1, R_2,... R_n\}$ where each R_i can take on one of m! values. (If there are m alternatives,

there are m! permutations of those alternatives.) There are thus $(m!)^n$ elements in the domain. The corresponding range would be the set of values

$$\{R\} = \{x^1Rx^2R...x^{m-1}R\, x^m\}$$

where the set $\{x^1, x^2...x^m\}$ can take on m! possible values. The set {R}, the possible social choices, is identical to the set $\{ R_i\}$ available to any individual.

At this point we are in complete agreement with Arrow's definition of a SWF which states:

"By a social welfare function will be meant a process or rule which, for each set of individual orderings $R_1,...,R_n$ for alternative social states (one ordering for each individual), states a corresponding social ordering of alternative social states, R." (1951).

In addition, we consider the possibility of social choices which are tie sets. To motivate our discussion of tie sets, we take as an example the binary case of two alternatives, a and b, and n voters. This is the typical, traditional voting situation. The individual voters vote either aP_ib or bP_ia where aP_ib means voter i prefers a to b. The corresponding social choices are aPb and bPa. If n is an even number and n/2 voters vote aP_ib while the other n/2 voters vote bP_ia, then we have a tie which we indicate {aPb,bPa} or aTb. Note that aTb = bTa. Therefore, the set of range elements that can be considered social choices are aPb, bPa and the tie set, {aPb,bPa}.

Let N(a,b) be the number of voters who vote aP_ib , and N(b,a) be the number who vote bP_ia. The rule connecting domain and range elements is as follows: If N(a,b) > N(b,a), the social choice is aPb. If N(b,a) > N(b,a), the social choice is bPa. If N(a,b) = N(b,a) (which can only happen if n is even), the social choice is a tie {aPb,bPa}. Clearly, this element needs to be considered in the range as a distinct possibility.

Now let us consider preferences <u>and</u> indifferences. The individual indifference realationship is aI_ib which means the ith voter is indifferent between a and b while the social indifference relationship is aIb. The individual now can vote in one of three ways: aP_ib, bP_ia or aI_ib. The social choices are aPb, bPa, {aPb,bPa} ≡ aTb and aIb. Note that a distinction needs to be made between a social indifference and a social tie which are philosophically distinct. Now a particular SWF might map the case, N(a,b) = N(b,a), refered to above, into the social choice aIb while another SWF might map the same case into aTb. It is important to preserve the distinction between these two cases so that the same options are available in the world in which P and I are possible as are available in the world in which just P is possible.

Arrow (1951) claims to treat ties. He asserts: "...Axioms I and II do not exclude the possibility that for some distinct [alternatives] x and y, both xRy and yRx. A strong ordering, on the other hand, is a ranking in which no ties are possible." Arrow is implying here that a social choice could consist of the tie set {xRy, yRx}. Clearly, this would not apply to individual choice since each individual would submit his vote in the form xR_iy or yR_ix but not both. It should be pointed out that the tie set {xRy, yRx} ≡ xTy is not the same as indifference and does not imply the social choice xIy. Analagous to the case considered previously in which half the voters prefered a to b, half b to a and the social choice was {aPb,bPa}, the situation here might be that half the voters vote xR_iy and half vote xR_iy. The social choice {xRy, yRx} needs to be available as a distinct and logically separate possibility from the social choice xIy. In fact, xIy might only be appropriate if all the voters were indifferent between a and b but not appropriate if half the voters prefered a to b and half, b to a.

Arrow's (1951) proof that social choice is possible for two alternatives is questionable because he doesn't deal with the tie case, N(x,y) = N(y,x), properly. Arrow states: "DEFINITION 9: *By the method of* majority decision *is meant the social welfare function in which xRy holds if and only if the number of individuals such that* $xR_i\,y$ *is at least as great as the number of individuals such that* $yR_i\,x$*.*"

Therefore, the case in which N(x,y) = N(y,x) would be decided xRy. But this violates the principal of neutrality or self-duality that requires every alternative to be treated in exactly the same way. Murakami (1968) states: "As long as we are considering the world of two alternatives, self-duality can be regarded as impartiality or neutrality with respect to alternatives. A self-dual social decision function has exactly the same structure regarding issue x against y as it does regarding issue y against x." Self-duality is a stronger version of Arrow's Condition 3 — Citizen's Sovereignty, but one would think that, since Arrow provided for the possibility of the tie set, {xRy, yRx}, in Axiom I, it should be called for in this case. There is no reason to prefer x over y in this situation by calling for xRy as the solution in the tie case as opposed to yRx. You can't have it both ways. If you aren't going to allow the existence of tie sets as legitimate social choices, then there is no legitimate social choice in the binary case either. On the other hand, if tie sets are acceptable, then they must be admitted as potential social choice solutions for cases such that $n > 2$, and this will lead, as we shall show, to a disproof of Arrow's Impossibility Theorem.

In showing connectivity Arrow states: "Clearly, always either $N(x,y) \geq N(y,x)$ or $N(y,x) \geq N(x,y)$, so that, for all x and y, xRy or yRx." This is an incorrect statement. One could say correctly that 'either $N(x,y) \geq N(y,x)$ or $N(y,x) > N(x,y)$' or 'either $N(x,y) > N(y,x)$ or $N(y,x) \geq N(x,y)$' or 'either $N(x,y) > N(y,x)$ or $N(y,x) > N(x,y)$ or $N(y,x) = N(x,y)$.' The latter restatement then would suggest the conclusion that either xRy or yRx or {xRy, yRx}. However, Arrow's definition of majority rule would have to be changed to allow for the tie case. With these changes one could then go on to prove that social choice is indeed possible for the case of two alternatives, but not allowing the acceptance of the tie case leads to the conclusion that social choice is impossible for the tie case as well.

Arrow's statement that in a "strong ordering ... no ties are possible" violates the common sense notion considered above in which (when only preferences are considered) n/2 voters prefer a to b and n/2 voters prefer b to a. Clearly, this is a tie, and clearly we *cannot* have the social choice aIb since the indifference operator is not a part of the domain or the range. The social choice must be {aPb, bPa}.

In accordance with Arrow's Axiom I which states: "For all x and y, either xRy or yRx" and about which he states: "Note also that the word 'or' in the statement of Axiom I does not exclude the possibility of both xRy and yRx.", the social choice tie set, {xRy, yRx}, is made possible because of the assumption by Arrow of the *inclusive* or in Axiom I. If we would have had xRy AND yRx as a possibility in Axiom I, then indeed this would imply xIy. When the "inclusive or" interpretation of Axiom I is extended to three alternatives, we would have social choice solutions, for instance, of the form {aRbRc,bRaRc,cRbRa}. For example, let us imagine a situation in which there are 3 alternatives and 6 voters. There are 6 possible choices in the choice set: {aRbRc,aRcRb,bRaRc,bRcRa,cRaRb,cRbRa}. Let us assume that each voter votes for a different element of this set. There is then one vote for each possible social choice. The common sense solution is a tie among all the possible choices. Similarly, there are 24 possible choices for 4 alternatives, and, for the case of 24 voters each voting for a different choice, common sense would dictate a tie among all the possible social choices. A similar case can be made for m=5, 6, ... These are the broadest conceivable tie sets, and will be called maximal tie sets. Tie sets involving less than the total number of choices are also conceivable.

An important thing to keep in mind here is that a tie refers to elements of the range and not to alternatives. If there are just two alternatives in an election, we say, sloppily, that it's possible for there to be a tie between x and y when what we mean (considering just preference relationships) is that there is a possibility of a tie between xPy and yPx which are the social choices. In other words, xPy and yPx are the social choices for which a tie may exist not x and y which are the alternatives. Similarly, for xRy and yRx, the tie is between xRy and yRx.

Therefore, in general, we consider that the range consists of all possible elements, $\{R\} = \{x^1Rx^2R...x^{m-1}R\, x^m\}$, plus elements which represent *all possible combinations of these elements which are the tie solutions.*

We take as the range of the SWF the *power set* (Stoll[1979]) of the set of all possible rankings, ρ(A), where

$$A = \{R^1,R^2,...,R^q\}$$

The set $\{R^1,R^2,...,R^q\}$ represents every possible ranking of the alternatives a,b,c... . q = m!. ρ(A) is the set of all possible subsets of A. If the vote is split precisely equally among every possible ranking, then the social choice would be equal to the tie set A. If there is a singular solution, then the social choice is equal to one of the elements of the set A. Other subsets of A would represent tie solutions of varying degrees.

Paradox of Voting

According to the Condorcet method for determining the outcome of an election, we consider each of the alternatives in pairs, determine the winner for each pair and then determine the final social ordering by combining these results. For example, if there are 4 alternatives and 5 voters with votes abcd, abcd, adcb, cdab and acbd, clearly, aRb (since there are 5 votes for ab and none for ba), aRc (since there are 4 votes for ac and 1 vote for ca), aRd (4 votes for ad and 1 for da), cRb (3 votes for cb and 1 for bc), bRd (3 votes for bd and 1 for db) and cRd (4 votes for cd and 1 for dc). So the winner is acbd. However, there are cases for which this method will not work.

Consider the following: 3 alternatives and 3 voters. Voter 1 votes abc; voter 2 votes bca; and voter 3 votes cab. If we consider the alternatives pairwise we have 2 votes for ab and 1 for ba; 2 votes for bc and 1 vote for cb; 2 votes for ca and 1 for ac. Therefore, a is preferred to b is preferred to c is preferred to a, and we have the cycle discovered by Condorcet. This is called the "paradox of voting." Clearly, any of the choices, abc, bca or cab would be incorrect.

The heart of Arrow's analysis is the criterion known as the Independence of Irrelevant Alternatives. Arrow (1951) states that "...suppose that an election system has been devised whereby each individual lists all the candidates in order of his preference and then, by a preassigned procedure, the winning candidate is derived from these lists. ...Suppose an election is held, with a certain number of candidates in the field, each individual filing his list of preferences , and then one of the candidates dies. Surely the social choice should be made by taking each of the individual's preference lists, blotting out completely the dead candidate's name, and considering only the orderings of the remaining names in going through the procedure of determining a winner."

Now let's reconsider the voting paradox considered above, assume that one candidate dies and recompute from the individual lists. Clearly, if c dies, ab should be the winner since there are 2 abs to 1 ba. Similarly, if b dies, ca should be the winner, and, if a dies, bc should be the winner. Why shouldn't a similar demand be made of the social choice i.e. if a candidate dies, the new social choice is determined by blotting out the dead candidate's name from the social choice list? For example, if the social choice were abcd and c died, why shouldn't the new social choice be abd? This is precisely the case when the Condorcet criterion is used in a situation where it actually works. Let's consider the example considered previously in which there were 4 alternatives and 5 voters with votes abcd, abcd, adcb, cdab and acbd. Clearly, aRb, aRc, aRd, cRb, bRd and cRd. The winner is acbd. Let's say c dies. We have aRb, aRd and bRd from a consideration of the individual lists which leads by combination to the social choice abd, and we have the social choice abd by considering the social choice acbd and blotting out c. So we get the same social choice by building the solution from individual lists (blotting out the dead candidate) as we do by considering the social choice and blotting out the dead candidate.

Generalizing this notion, consider the tie solution {abc, bca, cab} for the voter's paradox case considered above. If c dies, we have the solution {ab, ba, ab}. Now we need some sort of rule for combining these elements in order to reduce this solution down to either ab or ba. This is not necessary when the solution is not a tie, but is necessary (although Arrow doesn't consider it) when there is a tie and the solution at the next lower stage contains fewer elements than the present stage solution. The rule that would produce correct results in the example under consideration would be: choose the element or elements whose number is greatest in the tie set after the appropriate alternative has been blotted out. There are 2 abs and 1 ba in the solution so we determine the reduced solution to be ab which matches with the solution which is built up from the individual binary choices. Similarly, we get bc and ca, respectively if a or b dies. So we can expand the 3 stage 2 solutions, ab, bc and ca to the stage 3 solution {abc, bca, cab} and we can reduce the stage 3 solution {abc, bca, cab} to the 3 stage 2 solutions ab, bc and ca.

An Algorithm which Generates Social Choices

We now consider a general algorithm for generating solutions to the social choice problem. In a real sense the algorithm is the SWF. We build the solution stage by stage starting at the binary level. We first determine all the social choices by pairwise comparisons of the m alternatives as determined by the individual voter lists. These are the social choice solution sets for stage 2, one set for each possible pair of alternatives. Then we build the stage 3 solution sets by taking a binary solution and expanding it by combining it with another alternative such that the expanded stage 3 solution reduces correctly to the stage 2 solution when each alternative is blotted out as in the above example. We do this for each possible combination of 3 alternatives. We continue in this way until, if there are m alternatives, we have generated the stage m solution.

Now for some more terminology. We call the members of a social choice tie set "elements." We say that a social choice i-ary element "covers" an (i-1)-ary element if a letter can be blotted out of the i-ary element in such a way that the reduced i-ary element is identical with the (i-1)-ary element. For example, abcd covers abc since, if a d is blotted out of abcd, we have abc.

A “combination rule” tells us how to combine terms when a letter is blotted out in a tie solution set, and the solution set at the next lower level contains fewer elements. We use the following combination rule when reducing an i-ary solution to an (i-1)-ary solution: 1) blot out a particular letter in each element of the tie solution set; 2) out of this set of elements, choose that set of elements as the reduced solution if, for each element in the reduced solution, there are more of them than there are of any element not in the reduced solution and there are the same number of them as there are for every other element in the reduced solution. Let's call this the “majority” combination rule.

For example, let us assume that the stage 3 and 4 solution sets are the following:

Letter Combination	Stage 3 Solution Sets	Stage 4 Solution Set
a,b,c	abc	{abcd, acdb,adbc}
a,b,d	adb	
a,c,d	acd	
b,c,d	{bcd, cdb, dbc}	

In this example, if we blot out a d at stage 4, we have the modified set, {abc, acb, abc}. There are 2 abcs and 1 acb. Therefore the reduced solution set at stage 3 is abc. If we blot out a c, we have the modified set, {abd, adb, adb}. There are 2 adbs and 1 abd. Therefore, the reduced solution set is adb. If we blot out a b, we have the modified set {acd, acd, adc}. There are 2 acds and 1 adc. Therefore, the reduced solution set is acd. Finally, if we blot out an a, we have the set {bcd, cdb, dbc} which is the solution set since all three elements occur the same number of times.

We generalize these notions to the following definition:

Definition 1: The Lawrence SWF is an algorithm which, for m alternatives and n voters, generates, for any stage i ($2<i\leq m$), solution sets such that, when any letter is blotted out and using the majority combination rule, the solution set reduces to a correct solution for stage i-1.

It should be pointed out that this is one very specific SWF which we will use to prove that social choice is possible by proving that it always (for any m,n) produces solutions which meet Arrow's five criteria and two axioms. Other SWFs may exist as well.

Examples

If m=3, there are 27 possible combinations of pairwise comparisons since all domain elements can be collapsed down to 27 different cases. At the binary level we have either xRy or yRx or xTy. We work out the solutions as follows for each case. Each possible solution is rated to see how well it covers the solution sets at stage 2 (1 point for each stage 2 element covered). Since there are 3 stage 2 solution sets each consisting of one element, a rating of 3 means that all stage 2 elements are covered by 1 stage 3 element and that element is, hence, the stage 3 solution. If there are no stage 3 elements with a 3 rating, then there will be more than 1 stage 3 element in the solution set.

<u>**Case 1:**</u> **aRb, aRc, bRc**

Stage 3 Choices	Rating
abc	3
acb	2
bac	2
bca	1
cab	1
cba	0

Solution: **aRbRcCheck:** Blot out a; Solution—bRc;
Blot out b; Solution—aRc;
Blot out c; Solution—aRb;

<u>**Case 2:**</u> **aRb, aRc, cRb**

Stage 3 Choices	Rating
abc	2
acb	3
bac	1
bca	0
cab	2

cba	1

Solution: **aRcRbCheck:** Blot out a; Solution—cRb;
Blot out b; Solution—aRc;
Blot out c; Solution—aRb;

__**Case 3:**__ **aRb, cRa, bRc**

Stage 3 Choices	Rating
abc	2
acb	0
bac	1
bca	2
cab	2
cba	1

Solution: **{aRbRc, bRcRa, cRaRb}** **Check:** Blot out a; Solution—bRc;
Blot out b; Solution—cRa;
Blot out c; Solution—aRb;

__**Case 4:**__ **aRb, cRa, cRb**

Stage 3 Choices	Rating
abc	1
acb	2
bac	0
bca	1
cab	3
cba	2

Solution: **cRaRbCheck:** Blot out a; Solution—cRb;
Blot out b; Solution—cRa;
Blot out c; Solution—aRb;

__Case 5:__ **bRa, aRc, bRc**

Stage 3 Choices	Rating
abc	2
acb	1
bac	3
bca	2
cab	0
cba	1

Solution: **bRaRcCheck:** Blot out a; Solution—bRc;
Blot out b; Solution—aRc;
Blot out c; Solution—bRa;

__Case 6:__ **bRa, aRc, cRb**

Stage 3 Choices	Rating
abc	1
acb	2
bac	2
bca	1
cab	1
cba	2

Solution: **{aRcRb, cRbRa, bRaRc}** **Check:** Blot out a; Solution—cRb;
Blot out b; Solution—aRc;
Blot out c; Solution—bRa;

Case 7: **bRa, cRa, bRc**

Stage 3 Choices	Rating
abc	1
acb	0
bac	2
bca	3
cab	1
cba	2

Solution: **bRcRaCheck:** Blot out a; Solution—bRc;
Blot out b; Solution—cRa;
Blot out c; Solution—bRa;

Case 8: **bRa, cRa, cRb**

Stage 3 Choices	Rating
abc	0
acb	1
bac	1
bca	2
cab	2
cba	3

Solution: **cRbRaCheck:** Blot out a; Solution—cRb;
Blot out b; Solution—cRa;
Blot out c; Solution—bRa;

Cases 9 through 27 are covered in Appendix 1.

For m=4, there are 64 cases not counting ties. These solutions are given in Appendix 2.

Proof that Algorithm Satisfies Arrow's Criteria

Axiom I: **Connectivity**

Either xRy or yRx or {xRy, yRx} by construction.

Axiom II: **Transitivity**

For all x, y and z, xRy and yRz imply xRz by construction; xRy and yTz imply xRz; xTy and yRz imply xRz; and xTy and yTz imply xTz. As long as a solution can be expressed in the form $aQ^1b\ Q^2c...y\ Q^{m-1}z$ where Q^i can be either R or T, the solution is transitive. Alternatively, any solution expressed in the form **ab(c,d)e(f,g,h)ij** where **(c,d)** denotes cTd is transitive.

Condition 1: **Existence of a free triple**

Arrow only required that some set of three alternatives be available for any logical ordering. Our algorithm assigns solutions for every logical ordering of every individual voter.

Condition 2: **Positive Association of Individual and Social Values**

This Condition requires that, if every individual voter raises some candidate in his "preference or indifferenve" list, that candidate must not be lowered in the social choice. The algorithm considered here satisfies an even stronger criterion which is, if any individual voter raises a candidate in his "preference or indifference" list, that candidate must not be lowered in the social choice.

Since the social choice is based on the choices made on binary pairs, let us consider only one voter and only two candidates, a and b. Let us say this voter originally preferred or was indifferent between a and b and then switched his vote to b over a. As long as the majority of voters still prefer or are indifferent between a and b after the switch, there will be no change in the social choice. However, there is the possibility that the change of one vote will change the majority to b over a. Then, at stage 2, bRa. At stage 3, if we originally had a unique solution, then it would have to be either abc or acb. If we originally had a tie solution, then it would have to be {abc, bca, cab} or {acb, cba, bac}. If we originally had a unique solution, then (after the change) we would either have a unique solution in which b is preferred to a or a tie solution in which, in at least one element, b is ranked higher than a. If we originally had a tie solution, then (after the change) we would have a unique solution in which b is ranked higher than a. In any case, if a switch between two candidates by one individual voter affects the social choice at the binary level, it will affect the social choice at any other level since those social choices are built up from the binary level.

Condition 3: **The Independence of Irrelevant Alternatives**

Since the solution is computed stage by stage from binary pairs, it will always be the same if one or more candidates dies or drops out. In fact the solution can be recomputed starting at stage m and going down in stage number as well as starting at stage 1 and going up.

Condition 4: **Citizens' Sovereignty**

The social choice is imposed if there is some pair of alternatives a and b such that the Social Choice will always be bRa even if, for every individual voter aR_ib. In the algorithm under consideration here, if the majority of voters prefers a to b, then aRb and vice versa by construction.

Condition 5: **The Condition of Nondictatorship**

There is no dictator by construction, if the majority prefers or is indifferent to a over b, then aRb and vice versa.

Formal Explication of the Algorithm

The following is a formal delineation of the steps involved in the algorithm. We assume we have the correct solutions for the (m-1)th stage
(m > 2) and want to develop the solution for the mth stage.

1) Label all the alternatives alphanumerically such as **a, b, c** etc.
2) List all the (m-1)th stage letter combinations in lexicographical order.

3) For each (m-1)th stage letter combination, list the (m-1)th stage solution next to it forming the (m-1)th stage winning matrix. The elements of each solution are written, for example, **ab(c,d)** etc.
4) Consider each element in the winning matrix in lexicographical order i.e. from left to right columnwise and from top to bottom rowwise.
5) For each element in order list the possible mth stage elements by inserting the remaining letter at the end of the element to form the first mth stage element and then moving that letter one place to the left to form the next element etc. This represents the lexicographical ordering of the mth stage elements. After this process has been completed, the remaining letter is inserted in the same way from right to left again forming elements with possible tie alternatives.
6) For each possible mth stage element assign a rating which is computed by calculating the number of (m-1)th stage elements that are "covered" by this element where "covered" has been defined previously.
7) Choose that mth stage element with the highest rating as a potential element of the mth stage winning set. If there is a tie in the ratings consider the first element of the tie in lexicographical order.
8) Keep a list of the (m-1)th stage elements that are covered as they occur as a result of the inclusion of a potential mth stage element in the winning set. For each "covered" element, keep a record as to how many times it has been covered.
8) Make sure that, when the potential element are considered to be part of the winning solution, no (m-1)th stage element is covered more than twice.
9) If a potential stage m element results in a (m-1)th stage element being covered more than twice, then consider the next element in lexicographical order of the same rating or next lower rating. Go back to step 1.
10) Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.
11) If there are elements not in the winning matrix that are covered more than once, the potential element must be thrown out. Consider the next element in lexicographical order of the same or next lower rating. Go back to step 1.
12) If all potential stage 5 elements have been considered and no suitable element has been found, then go back to the last element included in the winning set that could be changed in such a way as to result in the least number of changes to the

winning set. Change that element to another one thus allowing one of the presently considered elements to be used in the winning set.
13) Add the potential element to the winning set.
14) Go back to step 1 and continue until every (m-1)th stage element has been considered.
15) If some (m-1)th stage elements have not been covered twice, start over considering those particular elements in lexicographical order.
16) Continue until all (m-1)th stage elements have been covered exactly twice.

An example worked out for the case m=5 is given in Appendix 3.

The proof that the algorithm works in every case is given in Appendix 4.

New Directions

Since many of the solutions are ties, we may use an additional criterion to choose among them. In fact we could introduce the concept of "digital utility" which would be a measure of the "goodness of fit" of each of the elements of the tie set. We could measure for each individual voter the goodness of fit of his preference list with the social choice by measuring the "distance," for each alternative, between the position of that alternative in the voter's preference list and the position of that alternative in the social choice. For instance, if voter i places alternative **a** 2nd in his list and the social choice places **a** 4th, there is a distance of 2 between the individual choice and the social choice. Summing over all alternatives and all individuals, we could get a measure of the digital utility for each element of the tie set. The element with the lowest summation would be the one with the highest digital utility, and, therefore, could be chosen as the social choice.

There is reason to believe that the social choices produced by the algorithm we have presented are stable in that it doesn't pay for any voter to vote insincerely. We quote Murakami (1968): "Therefore, if a democracy is based on pairwise comparisons, the outcome of sincere individual decisions is, if it exists at all, always stable. Any insincere or strategic move cannot improve the situation for any individual. This is one of the essential features of democracy based on pairwise comparison. Therefore, insofar as a democracy is based on pairwise comparisons, a distinction between individual decisions and individual preferences may not be so important.”

The aspect of pairwise comparisons also opens another door: that of probabilistic voting systems based upon limited information from each individual. Instead of millions of voters exhaustively ranking hundreds of alternatives, we can envision a voting system in which different voters are assigned different pairs of candidates to be ranked on a pairwise or a partial list ranking basis. Then all this information can be integrated to form the social choice with the probability of error made as low as desired by increasing the number of pairwise or partially ordered lists considered. The results could be compared with non-probabilistic voting systems for accuracy of results and effort (on the voters’ parts) expended.

Conclusions

We have shown that Arrow’s Impossibility Theorem is flawed since it doesn’t handle tie solutions correctly. We have demonstrated an algorithm which generates social choices and, therefore, constitutes a SWF. We allow a social choice to consist of a set of tie elements. The algorithm is based on binary, pairwise comparisons and satisfies a strengthened version of Arrow’s conditions and axioms as originally expounded . The social choices for all combinations of two alternatives are first determined. Then the social choices for 3, 4, ... alternatives are built up stage by stage. We have proven that the algorithm provides solutions for all values of m (number of alternatives) and n (number of voters). Therefore, Condorcet’s “paradox of voting” has been resolved and Social Choice is possible.

Appendix 1

Social Choice Solutions for m=3

(Both “preferences or indifferences” and ties)

Case 9: **aRb, aRc, bTc**

Stage 3 Alternatives	Rating
abc	2
acb	2
bac	1
bca	0
cab	1
cba	0
a(b,c)	3
(b,c)a	1
b(a,c)	0
(a,c)b	1
c(a,b)	0
(a,b)c	1
(a,b,c)	1

Solution: **aRbIc [a(b,c)]** **Check:** Blot out **a**; Solution—**bIc**;
Blot out **b**; Solution—**aRc**;
Blot out **c**; Solution—**aRb**;

We will just present the rest of the solutions without giving the details.

<u>Case 10:</u>	**aRb, cRa, bTc**	**Solution:**	**cab, a(b,c), (b,c)a**
<u>Case 11:</u>	**bRa, aRc, bTc**	**Solution:**	**bac, a(b,c), (b,c)a**
<u>Case 12:</u>	**bRa, cRa, bTc**	**Solution:**	**(b,c)a**
<u>Case 13:</u>	**aRb, aTc, bRc**	**Solution:**	**abc, b(a,c), (a,c)b**
<u>Case 14:</u>	**aRb, aTc, cRb**	**Solution:**	**(a,c)b**
<u>Case 15:</u>	**bRa, aTc, bRc**	**Solution:**	**b(a,c)**
<u>Case 16:</u>	**bRa, aTc, cRb**	**Solution:**	**cba, b(a,c), (a,c)b**
<u>Case 17:</u>	**aTb, aRc, bRc**	**Solution:**	**(a,b)c**
<u>Case 18:</u>	**aTb, aRc, cRb**	**Solution:**	**acb, (a,b)c, c(a,b)**
<u>Case 19:</u>	**aTb, cRa, bRc**	**Solution:**	**bca, (a,b)c, c(a,b)**
<u>Case 20:</u>	**aTb, cRa, cRb**	**Solution:**	**c(a,b)**
<u>Case 21:</u>	**aRb, aTc, bTc**	**Solution:**	**a(b,c), (a,c)b, (a,b,c)**
<u>Case 22:</u>	**bRa, aTc, bTc**	**Solution:**	**b(a,c), (b,c)a, (a,b,c)**
<u>Case 23:</u>	**aTb, aRc, bTc**	**Solution:**	**a(b,c), (a,b)c, (a,b,c)**
<u>Case 24:</u>	**aTb, cRa, bTc**	**Solution:**	**c(a,b), (b,c)a, (a,b,c)**
<u>Case 25:</u>	**aTb, aTc, bRc**	**Solution:**	**b(a,c), (a,b)c, (a,b,c)**
<u>Case 26:</u>	**aTb, aTc, cRb**	**Solution:**	**c(a,b), (a,c)b, (a,b,c)**
<u>Case 27:</u>	**aTb, aTc, bTc**	**Solution:**	**(a,b,c)**

Appendix 2

Social Choice Solutions for m=4 (no ties considered)

Case 1:	aRb, aRc, aRd, bRc, bRd , cRd	Solution:	abcd
Case 2:	aRb, aRc, aRd, bRc, bRd, dRc	Solution:	abdc
Case 3:	aRb, aRc, aRd, bRc, dRb, cRd	Solution:	abcd, acdb, adbc
Case 4:	aRb, aRc, aRd, bRc, dRb, dRc	Solution:	adbc
Case 5:	aRb, aRc, aRd, cRb, bRd , cRd	Solution:	acbd
Case 6:	aRb, aRc, aRd, cRb, bRd , cRd	Solution:	abdc, acbd, adcb
Case 7:	aRb, aRc, aRd, cRb, dRb , cRd	Solution:	acdb
Case 8:	aRb, aRc, aRd, cRb, dRb , dRc	Solution:	adcb
Case 9:	aRb, aRc, dRa, bRc, bRd , cRd	Solution:	abcd, bcda, dabc
Case 10:	aRb, aRc, dRa, bRc, bRd , dRc	Solution:	abdc, bdac, dabc
Case 11:	aRb, aRc, dRa, bRc, dRb, cRd	Solution:	abcd, dabc, cdab
Case 12:	aRb, aRc, dRa, bRc, dRb, dRc	Solution:	dabc
Case 13:	aRb, aRc, dRa, cRb, bRd , cRd	Solution:	acbd, cbda, dacb
Case 14:	aRb, aRc, dRa, cRb, bRd , dRc	Solution:	acbd, bdac, dacb
Case 15:	aRb, aRc, dRa, cRb, dRb , cRd	Solution:	acdb, cdab, dacb
Case 16:	aRb, aRc, dRa, cRb, dRb , dRc	Solution:	dacb
Case 17:	aRb, cRa, aRd, bRc, bRd , cRd	Solution:	abcd, bcad, cabd
Case 18:	aRb, cRa, aRd, bRc, bRd , dRc	Solution:	abdc, bdca, cabd
Case 19:	aRb, cRa, aRd, bRc, dRb , cRd	Solution:	adbc, bcad, cadb
Case 20:	aRb, cRa, aRd, bRc, dRb , dRc	Solution:	adbc, dbca, cadb
Case 21:	aRb, cRa, aRd, cRb, bRd , cRd	Solution:	cabd
Case 22:	aRb, cRa, aRd, cRb, bRd , dRc	Solution:	cabd, abdc, dcab
Case 23:	aRb, cRa, aRd, cRb, dRb , cRd	Solution:	cadb
Case 24:	aRb, cRa, aRd, cRb, dRb , dRc	Solution:	cadb, dcab, adcb
Case 25:	aRb, cRa, dRa, bRc, bRd , cRd	Solution:	abcd, bcda, cdab
Case 26:	aRb, cRa, dRa, bRc, bRd , dRc	Solution:	abdc, bdca, dcab
Case 27:	aRb, cRa, dRa, bRc, dRb , cRd	Solution:	cdab, bcda, dabc
Case 28:	aRb, cRa, dRa, bRc, dRb , dRc	Solution:	dcab, dbca, dabc
Case 29:	aRb, cRa, dRa, cRb, bRd , cRd	Solution:	cabd, cbda, cdab
Case 30:	aRb, cRa, dRa, cRb, bRd , dRc	Solution:	cabd, bdca, dcab
Case 31:	aRb, cRa, dRa, cRb, dRb , cRd	Solution:	cdab
Case 32:	aRb, cRa, dRa, cRb, dRb , dRc	Solution:	dcab

Case 33:	**bRa, aRc, aRd, bRc, bRd , cRd**	**Solution:**	**bacd**
Case 34:	**bRa, aRc, aRd, bRc, bRd , dRc**	**Solution:**	**badc**
Case 35:	**bRa, aRc, aRd, bRc, dRb , cRd**	**Solution:**	**acdb, bacd, dbac**
Case 36:	**bRa, aRc, aRd, bRc, dRb , dRc**	**Solution:**	**adbc, badc, dbac**
Case 37:	**bRa, aRc, aRd, cRb, bRd , cRd**	**Solution:**	**acbd, bacd, cbad**
Case 38:	**bRa, aRc, aRd, cRb, bRd , dRc**	**Solution:**	**adcb, badc, cbad**
Case 39:	**bRa, aRc, aRd, cRb, dRb , cRd**	**Solution:**	**acdb, bacd, cdba**
Case 40:	**bRa, aRc, aRd, cRb, dRb , dRc**	**Solution:**	**adcb, badc, dcba**
Case 41:	**bRa, aRc, dRa, bRc, bRd , cRd**	**Solution:**	**bacd, bcda, bdac**
Case 42:	**bRa, aRc, dRa, bRc, bRd , dRc**	**Solution:**	**bdac**
Case 43:	**bRa, aRc, dRa, bRc, dRb , cRd**	**Solution:**	**bacd, cdba, dbac**
Case 44:	**bRa, aRc, dRa, bRc, dRb , dRc**	**Solution:**	**dbac**
Case 45:	**bRa, aRc, dRa, cRb, bRd , cRd**	**Solution:**	**acbd, bdac, cbda**
Case 46:	**bRa, aRc, dRa, cRb, bRd , dRc**	**Solution:**	**dacb, bdac, cbda**
Case 47:	**bRa, aRc, dRa, cRb, dRb , cRd**	**Solution:**	**acdb, dbac, cdba**
Case 48:	**bRa, aRc, dRa, cRb, dRb , dRc**	**Solution:**	**dacb, dbac, dcba**
Case 49:	**bRa, cRa, aRd, bRc, bRd , cRd**	**Solution:**	**bcad**
Case 50:	**bRa, cRa, aRd, bRc, bRd , dRc**	**Solution:**	**badc, bcad, bdca**
Case 51:	**bRa, cRa, aRd, bRc, dRb , cRd**	**Solution:**	**cadb, bcad, dbca**
Case 52:	**bRa, cRa, aRd, bRc, dRb , dRc**	**Solution:**	**adbc, bcad, dbca**
Case 53:	**bRa, cRa, aRd, cRb, bRd , cRd**	**Solution:**	**cbad**
Case 54:	**bRa, cRa, aRd, cRb, bRd , dRc**	**Solution:**	**badc, cbad, dcba**
Case 55:	**bRa, cRa, aRd, cRb, dRb , cRd**	**Solution:**	**cadb, cbad, cdba**
Case 56:	**bRa, cRa, aRd, cRb, dRb , dRc**	**Solution:**	**adcb, cbad, dcba**
Case 57:	**bRa, cRa, dRa, bRc, bRd , cRd**	**Solution:**	**bcda**
Case 58:	**bRa, cRa, dRa, bRc, bRd , dRc**	**Solution:**	**bdca**
Case 59:	**bRa, cRa, dRa, bRc, dRb , cRd**	**Solution:**	**bcda, cdba, dbca**
Case 60:	**bRa, cRa, dRa, bRc, dRb , dRc**	**Solution:**	**dbca**
Case 61:	**bRa, cRa, dRa, cRb, bRd , cRd**	**Solution:**	**cbda**
Case 62:	**bRa, cRa, dRa, cRb, bRd , dRc**	**Solution:**	**bdca, dcba, cbda**
Case 63:	**bRa, cRa, dRa, cRb, dRb , cRd**	**Solution:**	**cdba**
Case 64:	**bRa, cRa, dRa, cRb, dRb , dRc**	**Solution:**	**dcba**

Appendix 3

An Example of the Algorithm for m = 5

As an example we consider the following case which represents the social choices at the binary level as determined by the aggregate of the individual decisions:

aRb,	**aRc,**	**aTd, aRe**	
	bRc,	**bRd,**	**bRe**
		cRd,	**cTe**
			dRe

From the stage 2 solutions given above, we can generate the stage 3 and stage 4 solutions. We present the stage 4 solutions below and demonstrate how to construct the stage 5 solution from them. In order to simplify the notation, we introduce a shorthand for the R and T operators as follows: aRbRc...yRz becomes **abc...yz** and aTb becomes **(a,b)** so that, for example, aRbTcRdRe becomes **a(b,c)de**.

The stage 4 solutions form a matrix as follows:

	<u>Stage 4 Letter Combinations</u>	<u>Stage 4 Winning Matrix</u>
(1)	**a,b,c,d**	**abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b**
(2)	**a,b,c,e**	**ab(c,e)**
(3)	**a,b,d,e**	**abde, b(a,d)e, (a,d)be**
(4)	**a,c,d,e**	**acde, c(a,d)e, (a,d)(c,e), a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e)**
(5)	**b,c,d,e**	**bcde, bd(c,e), b(c,e)d**

The SWF algorithm proceeds as follows. Consider each letter combination in the matrix, M(j,k), in lexicographical order that has not been covered at least once$^{i.e.}$ M(1,1) through M(5,3). Find a stage 5 element which covers each such that no stage 4 element is covered more than twice and such that upon reduction from stage 5 to stage 4 there are no elements in the reduced set that are not part of the stage 4 solution that are covered more than once. When each element has been considered, go over the matrix again in lexicographical order and cover again those elements that have only been covered once. A stage 5 element "covers" a stage 4 element if a letter can be blotted out of the stage 5 element and the resultant element is identical to the stage 4 element.

Step 1

1) Consider M (1,1) = **abcd**. Insert an **e** in every possible position (starting to the right and working to the left) and compute the rating which is the number of stage 4 winners covered as follows:

Stage 4 Element	Potential Element of Stage 5 Winning Set		Rating
M(1,1) = **abcd**	**abcde**	4	
	abced	1	
	abecd	1	
	aebcd	1	
	eabcd	1	
	abc(d,e)		1
	ab(c,e)d		2
	a(b,e)cd		1
	(a,e)bcd		1

2) Pick highest rated one: **abcde**. Update list of covered elements. The number of times the element has been covered is indicated in parentheses.

Covering Element	Covered Elements
abcde	**abcd** (1), **abde** (1), **acde** (1), **bcde** (1)

3) Check to see that no element is covered more than twice. If it is, don't add covering element to winning set and go back to (1). If it is not, go to (4).

4) Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

(1) Blot out an **e**: We get **abcd**. **abcd** in winning set.
(2) Blot out a **d**: We get **abce**. **abce** not in winning set, but covered only once.
(3) Blot out a **c**: We get **abde**. **abde** in winning set.
(4) Blot out a **b**: We get **acde**. **acde** in winning set.
(5) Blot out an **a**: We get **bcde**. **bcde** in winning set.

Winning set is now {**abcde**}.
Element not in stage 4 winning matrix, but covered only once: **abce**

5) Go back to (1) and proceed with next element.

Step 2

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set			Rating
M(1,2) = **acbd**	**acbde**	3		
	acbed		1	
	acebd		1	
	aecbd		1	
	eacbd		1	
	acb(d,e)			1
	ac(b,e)d			1
	a(c,e)bd			1
	(a,e)cbd			1

Pick highest rated one: **acbde**. Update list of covered elements.

Covering Element	Covered Elements
acbde	**abcd** (1), **abde** (2), **acde** (2), **bcde** (1) **acbd** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:

{**abcde, acbde**}

(1) Blot out an **e**: We get **abcd**, **acbd**. In winning set: **abcd**, **abdc.**
(2) Blot out a **d**: We get **abce**, **acbe**. Not in winning set: **abce**, **acbe**.
(3) Blot out a **c**: We get **abde**, **abde**. In winning set: **abde**, **abde.**
(4) Blot out a **b**: We get **acde**, **acde**. In winning set: **acde**, **acde.**
(5) Blot out an **a**: We get **bcde**, **cbde**. In winning set: **bcde.** Not in winning set: **cbde**

Winning set is now:
{**abcde, acbde**}

Elements not in stage 4 winning matrix, but covered only once:
abce, **acbe, cbde**

Step 3

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,3) = **bc(a,d)**	**bc(a,d)e**	4
	bc(a,d,e)	2
	bce(a,d)	1
	bec(a,d)	1
	ebc(a,d)	1
	b(c,e)(a,d)	3
	(b,e)c(a,d)	1

Pick highest rated one: **bc(a,d)e**. Update list of covered elements.

Covering Element	Covered Elements
bc(a,d)e	**abcd** (1), **abde** (2), **acde** (2), **bcde** (2), **acbd** (1), **bc(a,d)** (1), **b(a,d)e** (1), **c(a,d)e** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde, bc(a,d)e**}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d)**. In winning set: **abcd, abdc, bc(a,d).**
(2) Blot out a **d**: We get **abce, acbe, bcae.** Not in winning set: **abce, acbe, bcae.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e**. In winning set: **abde, abde, b(a,d)e.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e.** In winning set: **acde, acde, c(a,d)e.**
(5) Blot out an **a**: We get **bcde, cbde, bcde**. In winning set: **bcde, bcde.** Not in winning set: **cbde**

Winning set is now:
{abcde, acbde, bc(a,d)e}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, cbde

Step 4

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,5) = **(a,d)bc**	**(a,d)bce**	2
	(a,d)bec	2
	(a,d)ebc	1
	(a,d,e)bc	2
	e(a,d)bc	1
	(a,d)b(c,e)	4
	(a,d)(b,e)c	1

Pick highest rated one: **(a,d)b(c,e)**. Update list of covered elements.

Covering Element	Covered Elements
(a,d)b(c,e)	**abcd** (1), **abde** (2), **acde** (2), **bcde** (2), **acbd** (1) **bc(a,d)** (1), **b(a,d)e** (1), **c(a,d)e** (1), **ab(c,e)** (1), **(a,d)bc** (1), **(a,d)be** (1), **(a,d)(c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde, bc(a,d)e, (a,d)b(c,e)**}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e).** In winning set: **ab(c,e).** Not in winning set: **abce, acbe, bcae.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be**. In winning set: **abde, abde, b(a,d)e, (a,d)be.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e).**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e)**. In winning set: **bcde, bcde.** Not in winning set: **cbde, db(c,e)**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e)}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, cbde, db(c,e)

Step 5

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,5) = **b(a,d)c**	**b(a,d)ce**	2
	b(a,d)ec	2
	b(a,d,e)c	2
	be(a,d)c	1
	eb(a,d)c	1
	b(a,d)(c,e)	4
	(b,e)(a,d)c	1

Pick highest rated one: **b(a,d)(c,e)**. Update list of covered elements.

Covering Element	Covered Elements
b(a,d)(c,e)	**abcd** (1), **abde** (2), **acde** (2), **bcde** (2), **acbd** (1), **bc(a,d)** (1), **b(a,d)e** (2), **c(a,d)e** (1), **ab(c,e)** (1), **(a,d)bc** (1), **(a,d)be** (1) **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e)**}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e).** In winning set: **ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e).**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e.**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e).**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e)**. In winning set: **bcde, bcde, bd(c,e).** Not in winning set: **cbde, db(c,e)**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e)}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cbde, db(c,e)

Step 6

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,6) = **c(a,d)b**	**c(a,d)be**	3
	c(a,d)eb	2
	c(a,d,e)b	1
	ce(a,d)b	1
	ec(a,d)b	1
	c(a,d)(b,e)	2
	(c,e)(a,d)b	2

Pick highest rated one: **c(a,d)be**. Update list of covered elements.

Covering Element	Covered Elements
c(a,d)be	**abcd** (1), **abde** (2), **acde** (2), **bcde**(2), **acbd** (1), **bc(a,d)** (1), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (1), **(a,d)bc** (1), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be**}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe.** In winning set: **ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e.**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe**. In winning set: **bcde, bcde, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cabe, cbde, db(c,e), cdbe.

Step 7

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(2,1) = **ab(c,e)**	**ab(c,e)d**	4
	ab(c,d,e)	1
	abd(c,e)	3
	adb(c,e)	1
	dab(c,e)	1
	a(b,d)(c,e)	1
	(a,d)b(c,e)	4

Pick highest rated one: **ab(c,e)d**. Update list of covered elements.

Covering Element	Covered Elements
ab(c,e)d	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (1), **bc(a,d)** (1), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (1), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1), **a(c,e)d** (1), **b(c,e)d** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e).** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d.**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d.** Not in winning set: **cbde, db(c,e), cdbe.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cabe, abed, cbde, db(c,e), cdbe.

The next stage 4 winner that has not already been covered twice is **a(c,e)d.**

<u>Step 8</u>

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(4,4) = **a(c,e)d**	**a(c,e)db**	1
	a(c,e)bd	2
	a(b,c,e)d	1
	ab(c,e)d	4
	ba(c,e)d	2
	a(c,e)(b,d)	1
	(a,b)(c,e)d	2

ab(c,e)d doesn't work since **abcd** has already been covered twice. Try **a(c,e)bd**. Update list of covered elements.

Covering Element	Covered Elements
a(c,e)bd	**abcd** (2), **abde**(2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (1), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (1), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd.**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d.**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d.** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, abed, aebd, cbde, db(c,e), cdbe, (c,e)bd.

Step 9

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(4,5) = **(c,e)(a,d)**	**(c,e)(a,d)b**	2
	(c,e)(a,b,d)	1
	(c,e)b(a,d)	1
	(b,c,e)(a,d)	1
	b(c,e)(a,d)	3

Pick highest rated one: **b(c,e)(a,d)**

Covering Element	Covered Elements
b(c,e)(a,d)	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (1), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d)}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d)**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d).**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d).**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d).**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d.** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd.**

Winning set is now:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d)}

Elements not in stage 4 winning matrix, but covered only once:

abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, abed, aebd, be(a,d), cbde, db(c,e), cdbe, (c,e)bd.

Step 10

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(4,6) = **(a,d,e)c**	**(a,d,e)cb**	1
	(a,d,e)bc	2
	(a,b,d,e)c	1
	b(a,d,e)c	2
	(a,d,e)(b,c)	1

Pick highest rated one: **(a,d,e)bc**

Covering Element	Covered Elements
(a,d,e)bc	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1), **(a,d,e)c** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c.**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d.** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, abed, aebd, be(a,d), (a,d,e)b, cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc.

<u>Step 11</u>

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,7) = **d(a,c,e)**	**d(a,c,e)b**	1
	d(a,b,c,e)	1
	db(a,c,e)	1
	bd(a,c,e)	2
	(b,d)(a,c,e)	1

Pick highest rated one: **bd(a,c,e)**

Covering Element	Covered Elements
bd(a,c,e)	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (2), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1), **(a,d,e)c** (1), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e)}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c)**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc.** Not in winning set: **bd(a,c).**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e).** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e).**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e).**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e).**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e)**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e)}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), abed, aebd, be(a,d), (a,d,e)b, bd(a,e), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc.

Now we go back and make a second pass over the stage 4 elements covering those that have only been covered once again.

Step 12

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,3) = **b(a,d)c**	**b(a,d)ce**	2
	b(a,d)ec	2
	b(a,d,e)c	2
	be(a,d)c	1
	eb(a,d)c	1
	b(a,d)(c,e)	4
	(b,e)(a,d)c	1

Next 1-coverer in lex order: **be(a,d)c.** This does not work since **be(a,d)** has already been covered once. Try **eb(a,d)c.**

Covering Element	Covered Elements
eb(a,d)c	**abcd** (2), **abde**(2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1), **(a,d,e)c** (1), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c.** Not in winning set: **bd(a,c).**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac.**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d).**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e).** Not in winning set: **e(a,d)c**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)c, cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc.

Step 13

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,6) = **c(a,d)b**	**c(a,d)be**	3
	c(a,d)eb	2
	c(a,d,e)b	1
	ce(a,d)b	1
	ec(a,d)b	1
	c(a,d)(b,e)	2
	(c,e)(a,d)b	2

Next 1-coverer in lex order: **c(a,d,e)b.** This does not work since **(a,d,e)b** has already been covered once. Try **ce(a,d)b.**

Covering Element	Covered Elements
ce(a,d)b	**abcd** (2), **abde**(2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (2), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1), **(a,d,e)c** (1), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c, c(a,d)b**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.** Not in winning set: **bd(a,c).**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c, ce(a,d).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e).** Not in winning set: **e(a,d)c, ce(a,d)**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc, cedb**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb.**

Winning set is now:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b}

Elements not in stage 4 winning matrix, but covered only once:

bd(a,c), abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d)c, ce(a,d), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb

Step 14

<u>Next Stage 4 Winner</u>	<u>Potential Element of Stage 5 Winning Set</u>	<u>Rating</u>
M(4,5) = **(c,e)(a,d)**	**(c,e)(a,d)b**	2
	(c,e)(a,b,d)	1
	(c,e)b(a,d)	1
	(b,c,e)(a,d)	1
	b(c,e)(a,d)	3

Next 1-coverer in lex order: **(c,e)(a,b,d).**

Covering Element	Covered Elements
(c,e)(a,b,d)	**abcd** (2), **abde**(2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (2), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (2), **(a,d,e)c** (1), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,b,d)}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c, c(a,d)b, c(a,d,b)**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.** Not in winning set: **bd(a,c), c(a,d,b).**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b).** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b).**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b).**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c, ce(a,d), (c,e)(a,d).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), (c,e)(a,d).** Not in winning set: **e(a,d)c, ce(a,d)**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc, cedb, (c,e)(d,b)**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b).**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,d,b)}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), c(a,d,b), abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), e(a,d)c, ce(a,d), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b)

Step 15

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(4,6) = **(a,d,e)c**	**(a,d,e)cb**	1
	(a,d,e)bc	2
	(a,b,d,e)c	1
	b(a,d,e)c	2
	(a,d,e)(b,c)	1

Next 1-coverer in lex order: **(a,d,e)cb.** This doesn't work since **(a,d,e)b** has already been covered once. Try: **(a,b,d,e)c.**

Covering Element	Covered Elements
(a,b,d,e)c	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (2), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (2), **(a,d,e)c** (2), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,b,d), (a,b,d,e)c}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c, c(a,d)b, c(a,d,b), (a,b,d)c**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.** Not in winning set: **bd(a,c), c(a,d,b), (a,b,d)c.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e).**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c, ce(a,d), (c,e)(a,d), (a,d,e)c.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), (c,e)(a,d), (a,d,e)c.** Not in winning set: **e(a,d)c, ce(a,d)**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc, cedb, (c,e)(d,b), (b,d,e)c**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b), (b,d,e)c.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,d,b), (a,b,d,e)c}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), c(a,d,b), (a,b,d)c, abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e), e(a,d)c, ce(a,d), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b), (b,d,e)c

Step 16

Potential Element of

Next Stage 4 Winner	Stage 5 Winning Set	Rating
M(4,7) = **d(a,c,e)**	**d(a,c,e)b**	1
	d(a,b,c,e)	1
	db(a,c,e)	1
	bd(a,c,e)	2
	(b,d)(a,c,e)	1

Next 1-coverer in lex order: **d(a,c,e)b.**

Covering Element	Covered Elements
d(a,c,e)b	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (2), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (2), **(a,d,e)c** (2), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,b,d), (a,b,d,e)c, d(a,c,e)b}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c, c(a,d)b, c(a,d,b), (a,b,d)c, d(a,c)b**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.** Not in winning set: **bd(a,c), c(a,d,b), (a,b,d)c, d(a,c)b.**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c, (a,c,e)b.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c, (a,c,e)b.**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e), d(a,e)b**. In winning set:

abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be. Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e), d(a,e)b.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c, ce(a,d), (c,e)(a,d), (a,d,e)c, d(a,c,e).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), (c,e)(a,d), (a,d,e)c, d(a,c,e).** Not in winning set: **e(a,d)c, ce(a,d)**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc, cedb, (c,e)(d,b), (b,d,e)c, d(c,e)b**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b), (b,d,e)c, d(c,e)b.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,d,b), (a,b,d,e)c, d(a,c,e)b}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), c(a,d,b), (a,b,d)c, d(a,c)b, abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c, (a,c,e)b, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e), d(a,e)b, e(a,d)c, ce(a,d), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b), (b,d,e)c, d(c,e)b

This completes the solution for stage 5.

Appendix 3

Proof that Algorithm Works in Every Case

We do a proof by induction. We assume that the algorithm provides solutions which are correct for stage m-1. Then we prove that the solutions are correct for stage m. We also know that the algorithm provides correct solutions for stage 3 as presented earlier.

Step 1:

Any two (m-1)ary solutions will reduce to the same (m-2)ary solution for the m-2 letters they have in common.

Proof

a) We have assumed that the solutions at stage m-1 are correct.
b) There are m solutions at stage m-1, one for each of m combinations of m letters taken m-1 at a time.

e.g. for m=5, the solutions are

Letter Combination	Solution
a,**b**,**c**,**d**	Z_1^1 (**abcd**), Z_2^1 (**abcd**), Z_3^1 (**abcd**)
a,**b**,**c**,**e**	Z_1^2 (**abce**), Z_2^2 (**abce**), Z_3^2 (**abce**)
a,**b**,**d**,**e**	Z_1^3 (**abde**), Z_2^3 (**abde**), Z_3^3 (**abde**)
a,**c**,**d**,**e**	Z_1^4 (**acde**), Z_2^4 (**acde**), Z_3^4 (**acde**)
b,**c**,**d**,**e**	Z_1^5 (**bcde**), Z_2^5 (**bcde**), Z_3^5 (**bcde**)

where Z_i^j (**wxyz**) is a permutation of **wxyz**.

c) We know that when a letter is "blotted out" of a (m-1)ary solution, the solution reduces to the correct (m-2)ary solution.

d) Any two (m-1)ary solutions have m-2 letters in common.
e) Therefore, if the uncommon letter is removed from each of two (m-1)ary solutions, both will reduce to the same (m-2)ary solution.

Step 2:

For any two (m-1)ary solutions, there are elements in both solutions which have m-2 letters which are the same and in the same order. In fact and by construction, there are 2n elements in each solution which have elements with the same letters in the same order where n is the number of elements in the (m-2)ary solution.

Proof

By construction

e.g. for m=6 and the following case

aRb	aRc	aRd	eRa	fRa
	bRc	bRd	bRe	fRb
		cRd	cRe	cRf
			dRe	dRf
				eRf

we have fifth stage solutions as follows:

a,b,c,d,e: **abcde**, **bcdea**, **eabcd**, **abdce**, **cbdea**, **eacbd**, **bcead**, **deabc**, **acdeb**
a,b,c,d,f: **abcdf**, **cdfab**, **fabcd**, **abdcf**, **bacdf**, **cdfba**, **dcfab**, **facbd**, **fadbc**, **fbcda**

The fourth stage solution for **a**,**b**,**c**,**d** is **abcd**.

When we reduce the above solution for **a**,**b**,**c**,**d**,**e** we get **abcde, eabcd**, and when we reduce the above solution for **a**,**b**,**c**,**d**,**f** we get **abcdf** and **fabcd**. Both solutions reduce correctly to **abcd**.

Step 3:

Any two (m-1)ary elements with (m-2) letters in common and in the same order can be covered by one m-ary element.

e.g. the two elements **abc(d,e)** and **abc(d,f)** can be covered by **abc(d,e,f)**.

Without loss of generality, let $a_1a_2\cdots a_{m-2}$ be the m-2 letters that each element has in common. Let's say that the (m-1)th letter is X for the first element and Y for the second.

Therefore, not considering ties at the binary level, we have

$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-2}$

where a_i is a distinct member of the set, {**a**,**b**,**c**···}, for $1 < i < m-1$, and $a_i =1$ for i=1 and i=m-1.

and $a_1a_2\cdots a_{j-1}Ya_j\cdots a_{m-2}$

where a_j is a distinct member of the set, {**a**,**b**,**c**···}, for $1< j < m-1$, and $a_j =1$ for j=1 and j=m-1.

We construct the m-ary element by taking the element, $a_1a_2\cdots a_{m-2}$, and inserting X between a_{i-1} and a_i and Y between a_{j-1} and a_j as follows:

$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{j-1}Ya_j\cdots a_{m-2}$.

Clearly, if i=j, we may have either

$a_1a_2\cdots a_{i-1}XYa_i\cdots a_{m-2}$

or $a_1a_2\cdots a_{i-1}YXa_i\cdots a_{m-2}$.

When ties at the binary level are considered, we have

$a_1a_2\cdots(a_{i-1},X)a_i\cdots a_{m-2}$ or $a_1a_2\cdots(a_{i-1},X,a_i)\cdots a_{m-2}$ or $a_1a_2\cdots a_{i-1}(X,a_i)\cdots a_{m-2}$

and

$a_1a_2\cdots(a_{j-1},Y)a_j\cdots a_{m-2}$ or $a_1a_2\cdots(a_{j-1},Y,a_j)\cdots a_{m-2}$ or $a_1a_2\cdots a_{j-1}(Y,a_j)\cdots a_{m-2}$

We construct the m-ary element in the same way using parentheses as appropriate.
e.g. for

$a_1a_2\cdots(a_{i-1},X)a_i\cdots a_{m-2}$ and $a_1a_2\cdots(a_{j-1},Y)a_j\cdots a_{m-2}$

and for i=j, we have

$a_1a_2\cdots(a_{i-1},X,Y)a_j\cdots a_{m-2}$

The proof is not substantially changed if some of the alternatives are tied:

e.g. the two elements **a(b,c)de** and **a(b,c)df** can be covered by **a(b,c)def** and **a(b,d)e** and **a(c,d)e** can be covered by **a(b,c,d)e.**

Step 4:

Each element of the (m-1)ary winning matrix can combine with at least one other element of the winning matrix in such a way as to form an m-ary element that covers those elements so combined. There are enough such elements to cover all elements in the (m-1)ary winning matrix at least once. We will call such m-ary elements *primary elements.*

Proof

a) Since any two (m-1)ary rows have to reduce to the same solution at stage m-2 for the m-2 letters they have in common, they will have 2n elements in common where n is the number of elements in the particular row at stage m-2.

e.g.

row p: $X[b_1^p]^1,X[b_1^p]^2,X[b_2^p]^1,X[b_2^p]^2,\cdots,X[b_n^p]^1,X[b_n^p]^2,X[b_{n+1}^p],\cdots,X[b_t^p]$

row q: $Y[b_1^q]^1,Y[b_1^q]^2,Y[b_2^q]^1,Y[b_2^q]^2,\cdots,Y[b_n^q]^1,Y[b_n^q]^2,Y[b_{n+1}^q],\cdots,Y[b_s^q]$

where

row x is that row of the winning matrix whose letter combination does not include x.
b_z^p and b_z^q are (m-1)ary elements such as **badc...t**. Each element contains m-1 letters.
$b_z^p = b_z^q$ for $1 \le z \le n$.
$b_z^p \neq b_z^q$ for $n+1 \le z \le \min(t,s)$, where min(t,s) is the minimum of t and s.
X and Y are letters and the brackets represent an operator such that X[AB...P] = XAB...P or AXB...P or ... or AB...XP or AB...PX.
$X[b_z^v]^1$ represents a different permutation of X and b_z^v than does $X[b_z^v]^2$

b) $X[b_z^p]^j$ and $Y[b_z^q]^j$ can be covered at stage m by $X[Y[b_i^p]]$ for $1 \le z \le n$.

c) That leaves the elements $X[b_{n+1}^p], \cdots, X[b_t^p]$ and $Y[b_{n+1}^q], \cdots, Y[b_s^q]$. By construction these elements cover (m-2)ary elements in rows other than p and q. So for each of these elements there exists an element in another row of the (m-1)ary winning matrix such that they each have m-2 letters in common and in the same order. Therefore, by **Step (3)** there is an m-ary element that covers each of these elements and at least one other.

<u>Example</u>

Let the (m-1)ary winning matrix be

$$\left|\begin{array}{lll}
\mathbf{acde} & \mathbf{cdea} & \mathbf{eacd} \\
\mathbf{acdf} & \mathbf{cdfa} & \mathbf{facd} \\
\mathbf{acef} & \mathbf{cefa} & \mathbf{efac} \\
\mathbf{adef} & \mathbf{defa} & \mathbf{efad} \\
\mathbf{cdef} & &
\end{array}\right|$$

The 5-ary set {**acdef**, **cdefa**, **efacd**} consists of primary elements and covers the 4-ary winning matrix exactly once.

Definition: Interference—when there are any two elements in a potential m-ary winning set that, when reduced to the (m-1)ary level, generate the same element which is not in the (m-1)ary winning matrix.

Step 5:

Two primary elements cannot interfere with each other.

Proof:

The only way interference can occur is if there are at stage m two elements such that, when a letter is blotted out, both elements reduce to the same (m-1)ary element and this element is not in the (m-1)ary winning matrix.

Without loss of generality, let the two m-ary primary elements be

$$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-2} \text{ and } a_1a_2\cdots a_{j-1}Xa_j\cdots a_{m-2}$$

When X is blotted out these both reduce to

$$a_1a_2\cdots a_{m-2}$$

so that there are two such elements at stage m-1. We assume that this element is not in the winning matrix and prove the assertion that two primary elements cannot interfere by contradiction.

Because both m-ary elements under consideration are primaries, they were both formed by merging two elements from the (m-1)ary winning matrix. Let these elements be

$$a_1a_2\cdots a_{k-1}\,a_{k+1}\cdots a_{i-1}Xa_i\cdots a_{m-2},\ a_1a_2\cdots a_{l-1}\,a_{l+1}\cdots a_{i-1}Xa_i\cdots a_{m-2}$$

and

$a_1a_2\cdots a_{k-1}\, a_{k+1}\cdots a_{j-1}Xa_j\cdots a_{m-2}$, $a_1a_2\cdots a_{l-1}\, a_{l+1}\cdots a_{j-1}Xa_j\cdots a_{m-2}$,

respectively.

This implies that at the (m-2) stage there is an element

$$a_1a_2\cdots a_{k-1}\, a_{k+1}\cdots a_{i-1}a_i\cdots a_{m-2}$$

and an element

$$a_1a_2\cdots a_{l-1}\, a_{l+1}\cdots a_{i-1}a_i\cdots a_{m-2}$$

since there are two of each of them at the (m-1)ary stage when an X is blotted out and we know, by assumption, that the (m-1)ary solution is correct. Therefore, there must be an element on row X (where row X is the row in the (m-1)ary winning matrix which does not contain an X in its letter combination), stage (m-1) that reduces to

$$a_1a_2\cdots a_{k-1}\, a_{k+1}\cdots a_{i-1}a_i\cdots a_{m-2}$$

when a K is blotted out and to

$$a_1a_2\cdots a_{l-1}\, a_{l+1}\cdots a_{i-1}a_i\cdots a_{m-2}$$

when an L is blotted out since every row at stage (m-1) must reduce correctly. The only element for which this is possible is

$$a_1a_2\cdots a_{m-2}$$

and, therefore, this element must be in the (m-1)ary winning matrix which contradicts the assumption and the assertion is proven.

Example

Consider the following stage 4 winning matrix:

abcd		
abce	**bcea**	**eabc**
abde	**bdea**	**eabd**
acde	**cdea**	**eacd**
bcde		

We can form the stage 5 primary **abcde** from **abce** and **abde** and the stage 5 primary **bcdea** from **bcea** and **bdea**, respectively. Then on blotting out an **a** at stage 5 we will have two **bcde**s. Therefore, **bcde** must be in the stage 4 winning matrix or else interference would occur. Since at stage 4 there are two **bce**s if an **a** is blotted out of row d and two **bde**s if an **a** is blotted out of row c, this implies that there is a **bcde** on row a.

<u>Step 6:</u>

For each (m-1)ary element there are m permutations of that element and the last remaining letter. (There are m letters altogether.) One of them is the primary element. So there are m-1 other permutations. Some of these cover two (m-1)ary elements and some cover one. We call these other permutations secondaries and we say they are related to the primary element from which they are derived.

e.g.

Let

$a_1a_2\cdots a_{m-1}$

be the (m-1)ary element. Then we have the possible set of m-ary permutations as follows:

$\{X\, a_1a_2\cdots a_{m-1},\ a_1X\, a_2\cdots a_{m-1},\ \cdots,\ a_1a_2\cdots a_{m-1}X\}$

Let

$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-1}$

be the primary element which covers two or more (m-1)ary elements. One of the covered elements is $a_1a_2\cdots a_{m-1}$ by construction.

Consider

$$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-1}$$

If a_i or a_{i-1} is blotted out, the resultant (m-1)ary elements might be in the (m-1)ary winning matrix. If $a_1a_2\cdots a_{i-1}Xa_{i+1}\cdots a_{m-1}$ is in the winning matrix, for example, then the element $a_1a_2\cdots a_{i-1}a_iX\cdots a_{m-1}$ covers two (m-1)ary elements. Every other permutation of X and $a_1a_2\cdots a_{m-1}$ results in an element in which the X is out of position from its place in the primary element and, hence, the resultant element can only be a 1-coverer and only when the X is blotted out. Therefore, a secondary can cover one or in two cases possibly two elements.

<u>Step 7:</u>

There are at least two secondaries derived from any given m-ary primary that will not interfere with any other m-ary primary or secondary.

<u>Proof</u>

Any primary differs from any other primary or secondary by having at least one letter in a different place. Let's consider a given primary or secondary element

$$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{j-1}Ya_j\cdots a_{m-2}$$

Then we consider a second primary element such as

$$a_1a_2\cdots a_{k-1}Xa_k\cdots a_{l-1}Ya_l\cdots a_{m-2}$$

in which the letter Y is the one letter definitely in a different position from the preceding element. This second primary is unrelated to the first element since it is not derived from it. The letter X "slides" along the second element forming different permutations at different positions, $(0 < k < m, a_0 = a_{m-1} = 1)$, and different related secondaries. In the first element, X is fixed. Now when $k=i$, there is possible interference when a Y is blotted out since both elements reduce down to

$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{j-1}a_j\cdots a_{m-2}$.

In all other positions of X (or values of k), there are two letters out of synch for the two elements so they will not reduce down to the same (m-1)ary element and hence there will be no interference.

If X and Y are adjacent in the first element

$a_1a_2\cdots a_{j-1}XYa_j\cdots a_{m-2}$

then there are two positions of X which could cause interference as follows

$a_1a_2\cdots a_{l-1}XYa_l\cdots a_{m-2}$, l=j

and

$a_1a_2\cdots a_{l-1}YXa_l\cdots a_{m-2}$, l=j

Therefore, there are at most two positions that could cause interference between a secondary and an unrelated element (primary or secondary).

If there are ties in the first element as follows,

$a_1a_2\cdots(a_{i-1},X,a_i,\cdots a_{j-1},Y,a_j)\cdots a_{m-2}$,

then the second element will produce interference only for those positions inside the parentheses. All other kinds of ties (not involving X and Y tied together) do not alter the above analysis.

A secondary cannot interfere with the primary it is derived from since for the two elements

$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-1}$

and

$a_1a_2\cdots a_{k-1}Xa_k\cdots a_{m-1}$

when a_j is blotted out for any value of j except a_j=X, the two reduced elements will not be identical since each X will be in a different position and when X is blotted out the reduced element

$$a_1a_2\cdots a_{m-2}$$

is in the stage m-1 winning matrix by definition.

Therefore, the assertion is proved true.

Step 8:

There are, therefore, m-2 other secondary elements which are non-interfering not considering ties for the moment. Choose one of these if necessary (derived from each primary) to be the second m-ary element to cover each (m-1)ary element. Each (m-1)ary element is then covered twice in such a way that, when the solution is reduced from m-ary to (m-1)ary, every other element in the reduced solution is covered at most once. Therefore, we have proven that, if there is a correct solution at stage m-1, it is possible to find a correct solution for stage m. We know all the solutions for m=3. Therefore, a solution exists for m= 4⋯∞.

When ties are considered, the number of possible non-interfering elements is reduced when both X and Y are in the tie by the number of tied alternatives. At least, when every alternative is tied, there are still two non-interfering elements as shown by the following. Assume the first element is as follows:

$$(a_1,a_2,\cdots a_{i-1},X,a_i,\cdots a_{j-1},Y,a_j,\cdots,a_{m-2})$$

Then the second element would be non-interfering for the following positions of X:

$$X(a_1,a_2,\cdots a_{i-1},a_i,\cdots a_{j-1},Y,a_j,\cdots,a_{m-2})$$

and

$$(a_1,a_2,\cdots a_{i-1},X,a_i,\cdots a_{j-1},Y,a_j,\cdots,a_{m-2})X$$

Therefore, there are at least two non-interfering elements. Since we know all the solutions for m=3, solutions exist for m= 4⋯∞.

References

1. K. J. Arrow, “Social Choice and Individual Values,” John Wiley & Sons Inc., New York, 1951.
2. A. Bergson, “Essays in Normative Economics,” Harvard University Press, Cambridge, Mass., 1966.
3. D. Black, “The Theory of Committees and Elections,” Cambridge University Press, London, 1958.
4. J-C de Borda, Mémoire sur les élections par scrutin, *Mémoires de l'Academie Royale des sciences année 1781,* (1781), 657-65.
5. M. J. A. N. marquis de Condorcet, “Essai sur l'application de l'analyse á la probabilité des décisions rendues á la pluralité des voix”, Imprimerie Royale, Paris, 1785.
6. Rev. C. L. Dodgson, “A Discussion of the Various Methods of Procedure in Conducting Elections”, 1873 ; “Suggestions as to the Best Method of Taking Votes, Where More than Two Issues are to be Voted on,” 1874; “A Method of Taking Votes on More than Two Issues,” 1876.
7. G.-G. Granger, “*La mathématique sociale du marquis de Condorcet*,” (Éditions Odile Jacob, Paris, 1989.
8. A. F. MacKay, “Arrow's Theorem,” Yale University Press, 1980.
9. I. McLean and F. Hewitt, “Condorcet,” Edward Elgar, 1994.
10. Y. Murakami, “Logic and Social Choice,” Routledge & Kegan Paul Ltd., London, 1968.
11. E. J. Nanson, Methods of Election, in British Government blue book, *Misc. No. 3,* Cd. 3501, 1907.
12. W.H. Riker, “Liberalism Against Populism: A Confrontation Between the Theory of Democracy and the Theory of Social Choice,” W. H. Freeman, San Francisco,1982.
13. J. Riley, “Liberal Utilitarianism,” Cambridge University Press, Cambridge, 1988.
14. N. Schofield, “Social Choice and Democracy,” Springer-Verlag, 1985.
15. A. K. Sen, “Collective Choice and Social Welfare,” Holden-Day, San Francisco, 1970.
16. R. R. Stoll, “Set Theory and Logic,” Dover Publications Inc., New York, 1979.