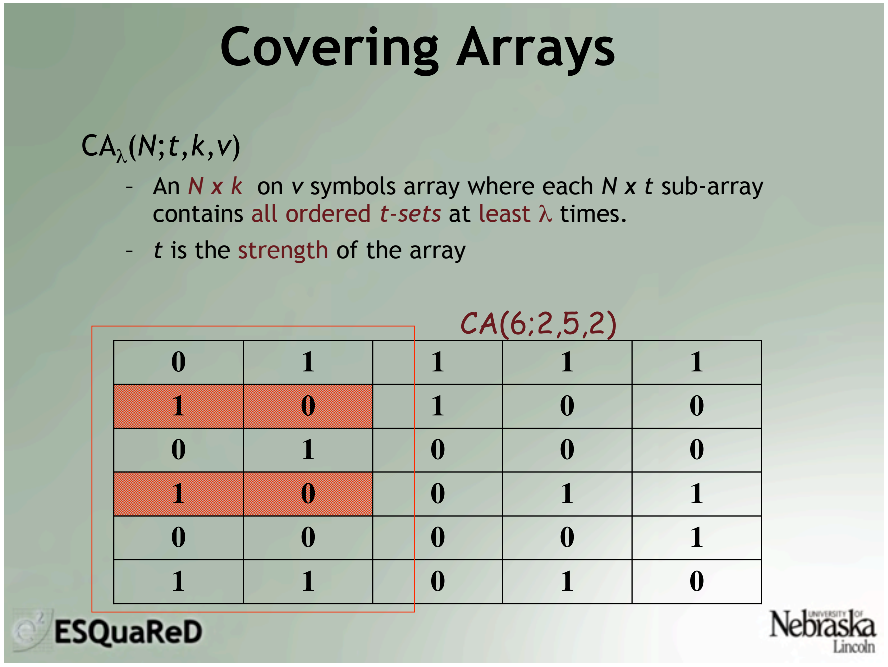
Variable Strength Covering Arrays
Applications and Challenges

Myra Cohen

Laboratory for Empirically-based Software Quality Research and Development



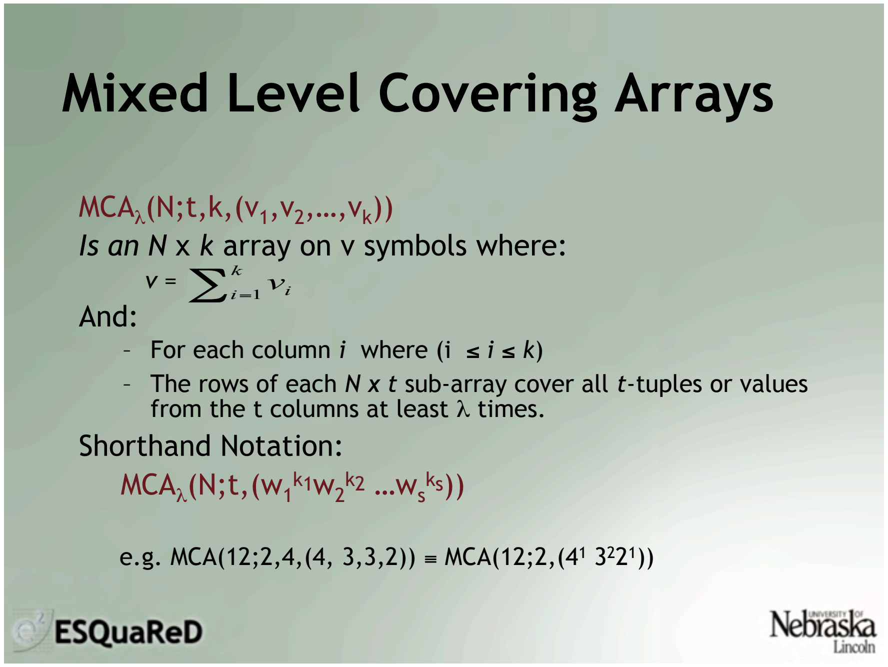
Covering Arrays

$CA_\lambda(N;t,k,v)$

- An *N x k* on *v* symbols array where each *N x t* sub-array contains all ordered *t-sets* at least λ times.
- *t* is the strength of the array

CA(6;2,5,2)

0	1	1	1	1
1	0	1	0	0
0	1	0	0	0
1	0	0	1	1
0	0	0	0	1
1	1	0	1	0



Mixed Level Covering Arrays

$MCA_\lambda(N;t,k,(v_1,v_2,\ldots,v_k))$

Is an N x *k* array on v symbols where:

$v = \sum_{i=1}^{k} v_i$

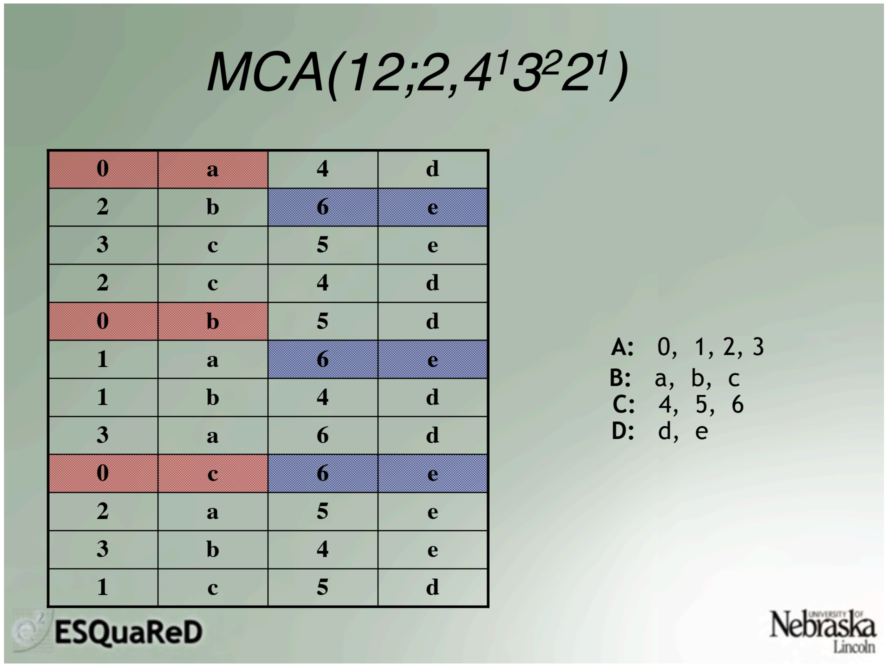
And:

- For each column *i* where (i ≤ *i* ≤ *k*)
- The rows of each *N x t* sub-array cover all *t*-tuples or values from the t columns at least λ times.

Shorthand Notation:

$MCA_\lambda(N;t,(w_1^{k_1} w_2^{k_2} \ldots w_s^{k_s}))$

e.g. $MCA(12;2,4,(4, 3,3,2)) \equiv MCA(12;2,(4^1\ 3^2 2^1))$



MCA(12;2,4¹3²2¹)

0	a	4	d
2	b	6	e
3	c	5	e
2	c	4	d
0	b	5	d
1	a	6	e
1	b	4	d
3	a	6	d
0	c	6	e
2	a	5	e
3	b	4	e
1	c	5	d

A: 0, 1, 2, 3
B: a, b, c
C: 4, 5, 6
D: d, e

Limitation

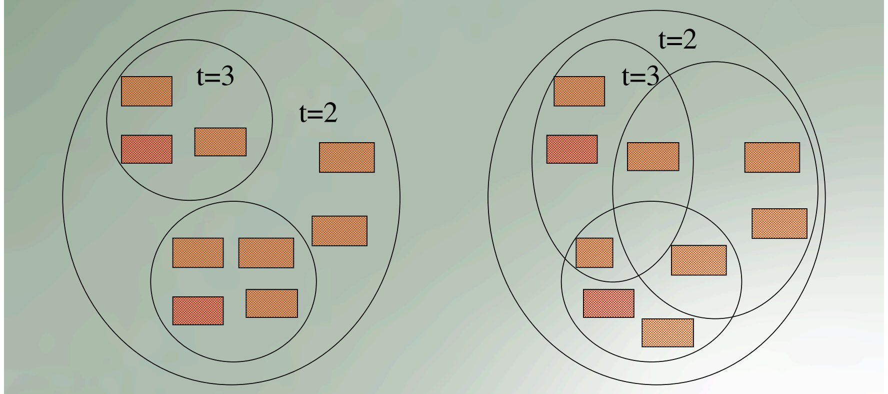
- Mixed level covering arrays have practical applications in software testing.
- But they view a system “flatly”. They force a (perhaps arbitrary) restriction on the importance of various parts of the system.

Motivation

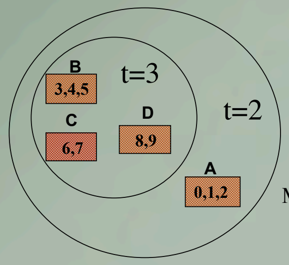
Scenarios:

- When testing a software system certain components may be closely interrelated
- Operational profiles give us information that certain areas of the system are used more often than others
- In modifying a system only certain regions are changed therefore we want to test more strongly in this area
- Failures in certain parts of a system are costlier than in others

Possible Models



Variable Strength Covering Arrays

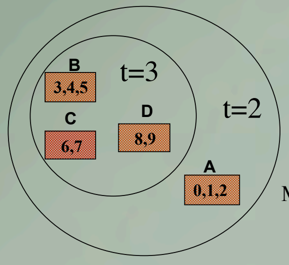


MCA($2,3^2 2^2$)

A	B	C	D
0	4	7	8
1	3	7	9
2	3	6	8
0	5	6	9
1	4	6	8
2	4	7	9
2	5	7	9
1	5	7	8
0	3	6	9

A 3 way array would have 18 rows

Variable Strength Covering Arrays



MCA(2,$3^2 2^2$)

A	B	C	D
0	4	7	8
1	3	7	9
2	3	6	8
0	5	6	9
1	4	6	8
2	4	7	9
2	5	7	9
1	5	7	8
0	3	6	9
0*	3	7	8
2*	4	6	9
1*	5	6	8

Additional rows to guarantee subset of 3-way coverage

Variable Strength Covering Array

- A VCA(N;t,k,(v_1,v_2,…v_k), C) is a t-way mixed level covering array on v symbols with a vector, C, of covering arrays each with strength > t and defined on a subset of the k columns of the VCA.

Variable Strength Arrays Using SA[1]

VCA	C	Size
VCA(2,3^{15},C)	-	16
	CA(3,3^3)	27
	CA(3,3^4)	27
	CA(3,3^5)	33
	CA(3,3^6)	33
	CA(3,3^9)	51
	CA(3,3^{15})	68
VCA(2,3^{20},10^2,C)	-	100
(22 factors: 20 have 3 values,	CA(3,3^{20})	100
2 have 10 values)	MCA(3,$3^{20}10^2$)	305

1. [C,C,C,G,M -2003]

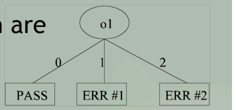
An Empirical Study[1]

- Distributed Quality Assurance (Skoll)
  - Distribute instances of the system configuration for testing in the field
  - Results can be returned to a centralized location
  - Includes fault localization techniques

1. [Yilmaz, Cohen, Porter - 2006]

Fault Characterization Process

Identifies configuration options and their settings which are responsible for the manifestation of failures



Config			Result
o1	o2	o3	
0	0	0	PASS
0	0	1	PASS
0	0	2	PASS
0	1	0	PASS
0	1	1	PASS
0	1	2	PASS
0	2	0	PASS
0	2	1	PASS
0	2	2	PASS

Config			Result
o1	o2	o3	
1	0	0	ERR #1
1	0	1	ERR #1
1	0	2	ERR #1
1	1	0	ERR #1
1	1	1	ERR #1
1	1	2	ERR #1
1	2	0	ERR #1
1	2	1	ERR #1
1	2	2	ERR #1

Config			Result
o1	o2	o3	
2	0	0	ERR #2
2	0	1	ERR #2
2	0	2	ERR #2
2	1	0	ERR #2
2	1	1	ERR #2
2	1	2	ERR #2
2	2	0	ERR #2
2	2	1	ERR #2
2	2	2	ERR #2

Fault Characterization

- Helps developers quickly pinpoint the root causes of failures
- Fundamental downside of the approach shown is that it requires testing ALL combinations of options: It does not scale

Covering Array Approach

- Systematically sample the configuration space, test only the selected configurations, and conduct fault characterization on the resulting data
- How good are the resulting characterizations?

Software System

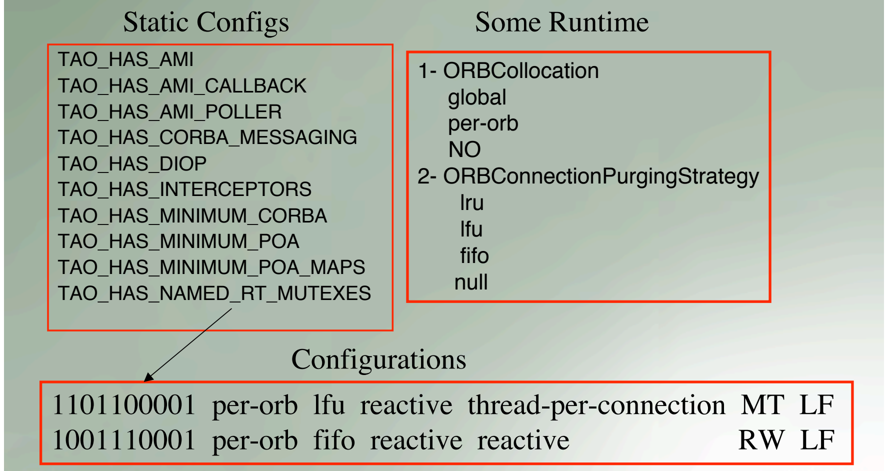
- ACE+TAO: an open source distributed CORBA middleware system
  - Large code base – 2M+ lines of C++ code
  - Over 500 configuration options
  - Dozens of OS, compiler and hardware platform combinations

Mappings

- 10 Static Binary Options (constraints reduce this to 92 feasible static configurations)
  - Only 29 of these compile successfully
    - Our model aggregates all of these into a single static option with 29 values
- 6 run time options with 2-4 values each

The Covering Array: $MCA(N;t,29^{1}4^{1}3^{4}2^{1})$

Mappings



Static Configs

TAO_HAS_AMI
TAO_HAS_AMI_CALLBACK
TAO_HAS_AMI_POLLER
TAO_HAS_CORBA_MESSAGING
TAO_HAS_DIOP
TAO_HAS_INTERCEPTORS
TAO_HAS_MINIMUM_CORBA
TAO_HAS_MINIMUM_POA
TAO_HAS_MINIMUM_POA_MAPS
TAO_HAS_NAMED_RT_MUTEXES

Some Runtime

1- ORBCollocation
global
per-orb
NO
2- ORBConnectionPurgingStrategy
lru
lfu
fifo
null

Configurations

1101100001 per-orb lfu reactive thread-per-connection MT LF
1001110001 per-orb fifo reactive reactive RW LF

Software System

- Test suite: 96 regression tests
  - Each designed to emit an error message in the case of failure
  - The error messages were captured, indexed, and recorded
- Almost a year of machine time for the exhaustive testing of 18,792 configurations – just a small portion of actual space

Constructing Covering Arrays

- Created 5 different t-way covering arrays for
  $2 \le t \le 6$
- Size of covering arrays:
  $MCA(N;t,29^{1}4^{1}3^{4}2^{1})$

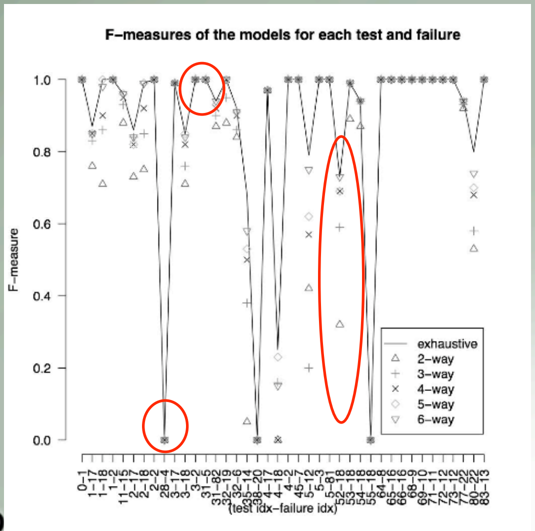
t	# of configurations	% reduction
2	116	99.4
3	348	98.2
4	1229-1236	93.5-93.4
5	3369-3372	82.1-82.0
6	9433-9453	49.8-49.7

Fault Localization

- Covering arrays performed better than random arrays of same size.
- Did almost as well as full configuration space at a reduced cost

(Use F Measure to determine how good our characterization is. It combines precision and recall)

Fault Localization



But

Given:

- Many of the faults were localized in the runtime options
- We had a large number of options for the one static factor

Question:

- Can we improve our fault localization by using VCAs?

VCAs Created

	Size
MCA t=2	116
2c4r VCA$(2,29^{1}4^{1}3^{4}2^{1}$,MCA$(4,4^{1}3^{4}2^{1}))$	116
2c5r	324
MCA t=3	348
3c5r	367-368

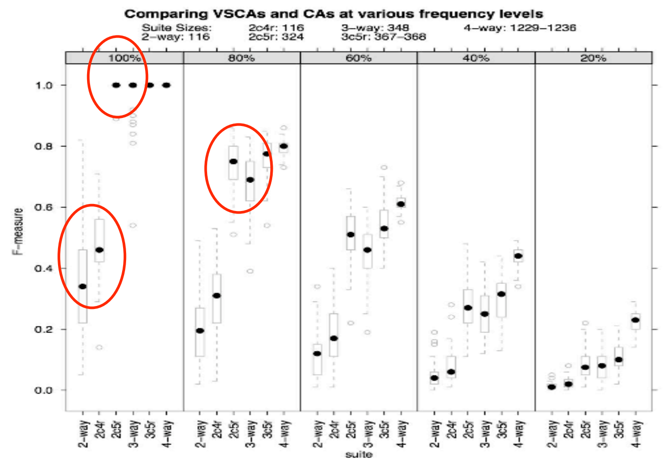
Results

Failure	OS	2-way	2c4r	2c5r	3-way	3c5r	4-way
2-17	Linux	.78	.81	.83	.81	.83	.81
80-22	Linux	.34	.51	.65	.61	.65	.67
4-18	Win	.69	.79	.83	.84	.85	.88

Simulation

- Real data was encouraging but inconclusive
- Most of our characterizations were almost perfect at lower strengths - we may not have many high order faults.
- We performed a simulation of 4 way (runtime) faults in our system at varying levels of determinism.

Results

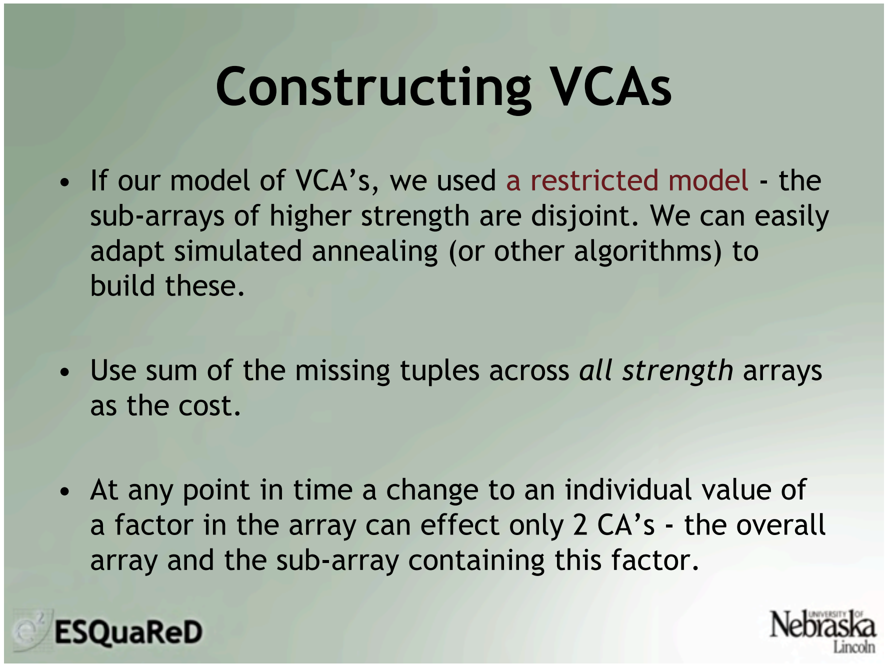


Constructing VCAs

- Gargano, L., Körner, J., and Vaccaro, U., Capacities: from information theory to extremal set theory, *Journal of Combinatorial Theory Series A*, 68, 2 (1994), 296--316.
- (Biyani) - IBM internal tool (tofu)
- Simulated Annealing - M.B. Cohen et. al
- Constructions: new: C. Cheng 2006

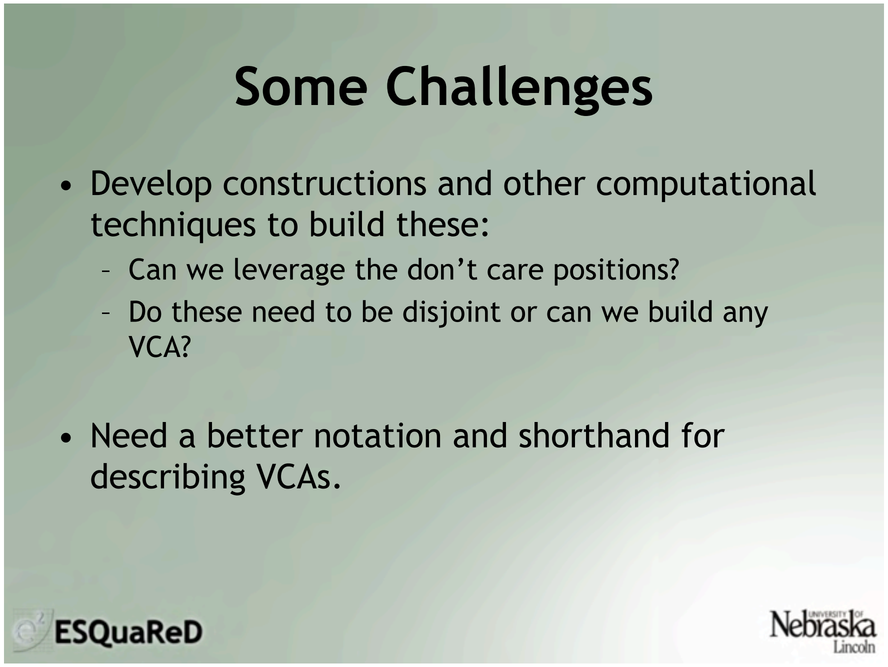
Roux Reference:

- Roux, Gilbert, *k*-Propriétés dans des tableaux de *n* colonnes; cas particulier de la *k*-surjectivité et de la k-permutivite, PhD dissertation, University of Paris, Department of Mathematics, 1987.



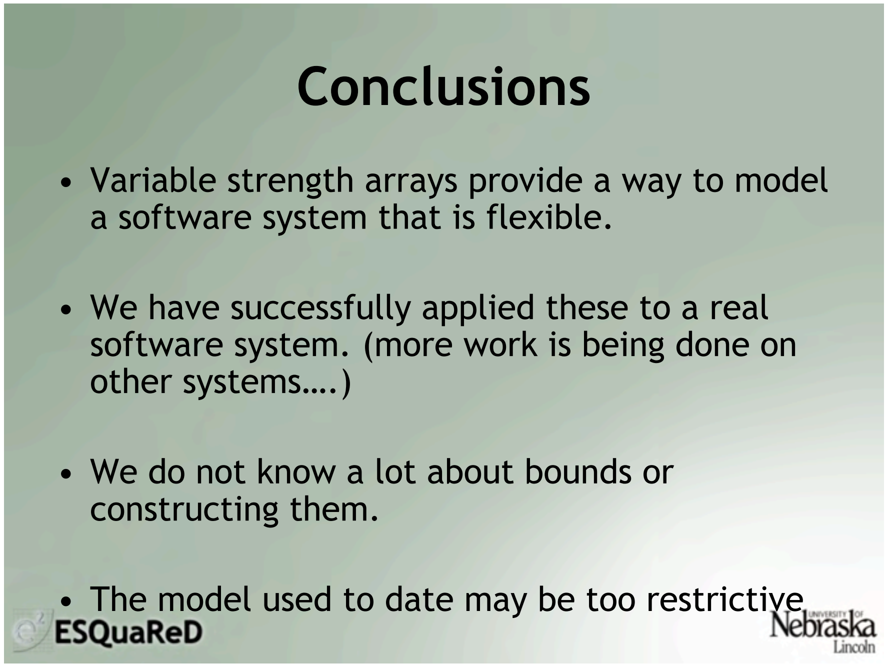
Constructing VCAs

- If our model of VCA's, we used a restricted model - the sub-arrays of higher strength are disjoint. We can easily adapt simulated annealing (or other algorithms) to build these.
- Use sum of the missing tuples across *all strength* arrays as the cost.
- At any point in time a change to an individual value of a factor in the array can effect only 2 CA's - the overall array and the sub-array containing this factor.



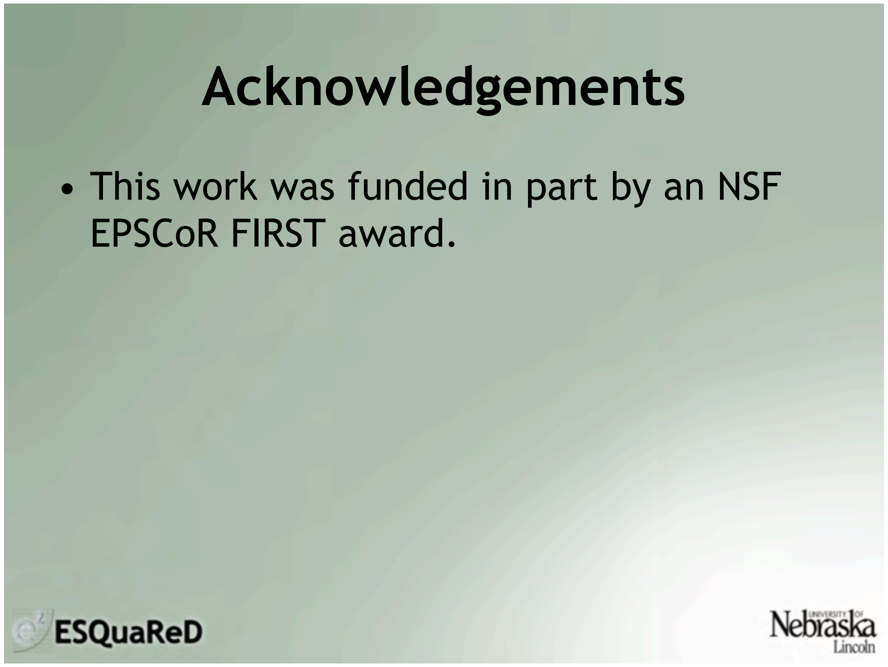
Some Challenges

- Develop constructions and other computational techniques to build these:
  - Can we leverage the don't care positions?
  - Do these need to be disjoint or can we build any VCA?
- Need a better notation and shorthand for describing VCAs.



Conclusions

- Variable strength arrays provide a way to model a software system that is flexible.
- We have successfully applied these to a real software system. (more work is being done on other systems….)
- We do not know a lot about bounds or constructing them.
- The model used to date may be too restrictive



Acknowledgements

- This work was funded in part by an NSF EPSCoR FIRST award.