Breaking Up is Hard to Do: An Investigation of Decomposition for Assume-Guarantee Reasoning

Jamieson M. Cobleigh
George S. Avrunin
Lori A. Clarke

Department of Computer Science
University of Massachusetts Amherst

Finite-State Verification (FSV)

- FSV techniques are used to check properties of a system
  - A property describes either a desired or undesired system behavior
- FSV techniques work on a finite model of a system

Finite-State Verification (FSV)

- FSV techniques are used to check properties of a system
  - A property describes either a desired or undesired system behavior
- FSV techniques work on a finite model of a system
- Verifying properties using FSV tools can become intractable due to the state-explosion problem
  - The cost for verification can be exponential in the size of the system being analyzed

Compositional Analysis

- A divide-and-conquer approach based on:
  - Breaking the system into pieces
  - Analyzing the pieces individually
  - Composing the results to determine if the desired global system property holds

Assume-Guarantee Reasoning

- Assume-guarantee reasoning is a compositional approach
  - Originally proposed by Jones [1983] and Pnueli [1984]
- Many assume-guarantee reasoning frameworks have been proposed:
  - e.g., Shurek and Grumberg [1990], Grumberg and Long [1994], Abadi and Lamport [1995]

Assume-Guarantee Reasoning

- Properties are triples: <*A*> *S* <*P*>
  - *S* is the subsystem under analysis
  - *P* is the property to be verified
  - *A* is the assumption that *S* makes about the subsystems with which it interacts

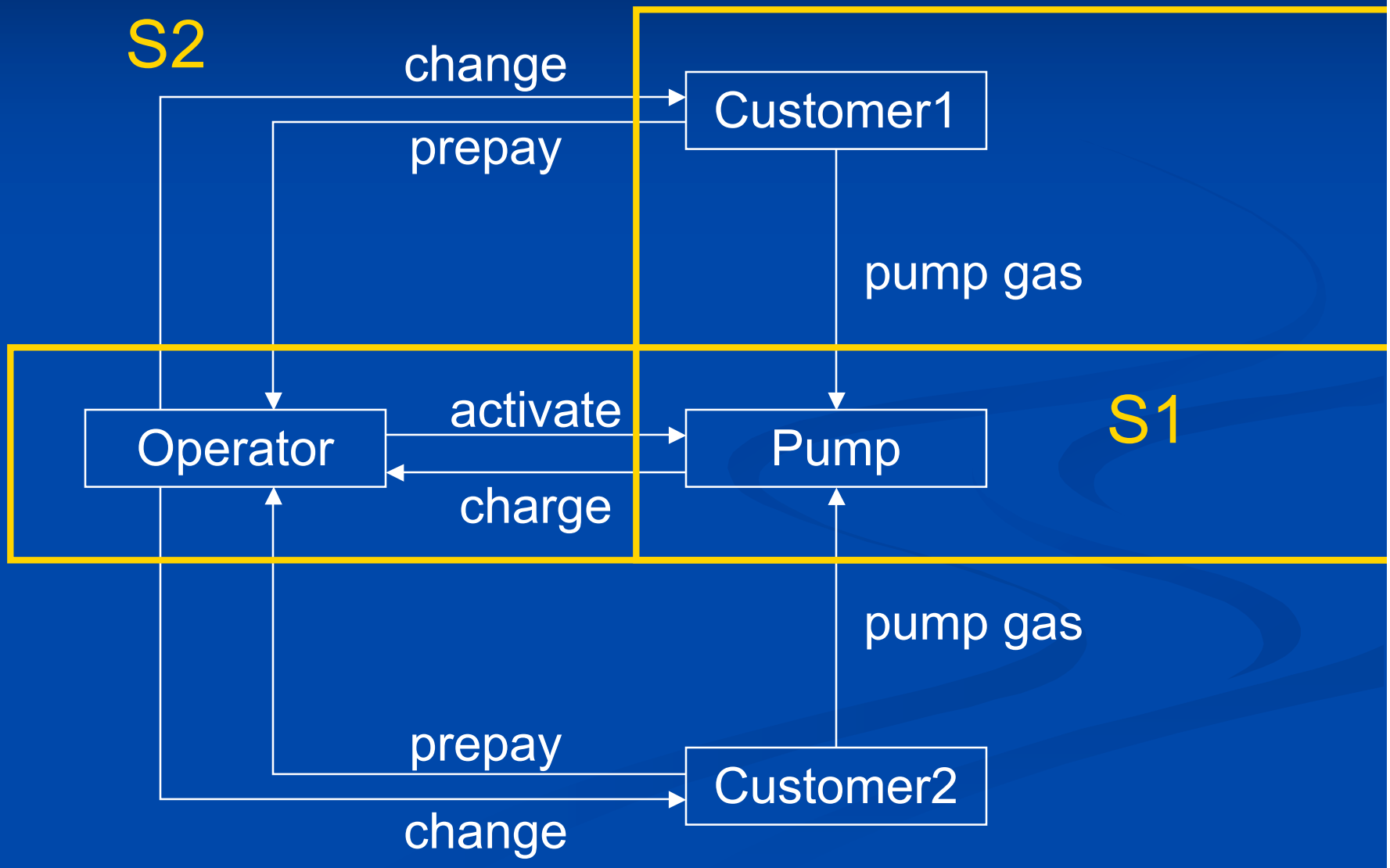
Assume-Guarantee Reasoning

- Properties are triples: <*A*> *S* <*P*>
  - *S* is the subsystem under analysis
  - *P* is the property to be verified
  - *A* is the assumption that *S* makes about the subsystems with which it interacts
- Simplest assume-guarantee rule:

$$\frac{\langle A \rangle\ S1\ \langle P \rangle \qquad \langle true \rangle\ S2\ \langle A \rangle}{\langle true \rangle\ S1 \parallel S2\ \langle P \rangle}$$

S2 discharges the assumption A

Gas Station


S2
change
Customer1
prepay
pump gas
activate
Operator
Pump
S1
charge
pump gas
prepay
Customer2
change

Evaluation of Assume-Guarantee Reasoning

- While many assume-guarantee reasoning techniques have been proposed, we know of no extensive evaluation using FSV
  - There are several obstacles to using assume-guarantee reasoning that have made evaluation difficult

Obstacles to Using Assume-Guarantee Reasoning

1. If the system is made up of more than two subsystems, those subsystems need to be decomposed into S1 and S2
   - How this is done can affect the time and memory usage by over an order of magnitude

$$\frac{\langle A \rangle\ S1\ \langle P \rangle \qquad \langle true \rangle\ S2\ \langle A \rangle}{\langle true \rangle\ S1 \parallel S2\ \langle P \rangle}$$

Obstacles to Using Assume-Guarantee Reasoning

2. After the subsystems are decomposed, an assumption is needed
   - Finding such assumptions can be difficult

$$\frac{\langle A \rangle\ S1\ \langle P \rangle \qquad \langle true \rangle\ S2\ \langle A \rangle}{\langle true \rangle\ S1 \parallel S2\ \langle P \rangle}$$

Obstacles to Using Assume-Guarantee Reasoning

2. After the subsystems are decomposed, an assumption is needed
   - Finding such assumptions can be difficult

$$\frac{\langle A \rangle\ S1\ \langle P \rangle \qquad \langle true \rangle\ S2\ \langle A \rangle}{\langle true \rangle\ S1 \parallel S2\ \langle P \rangle}$$

But, several automated approaches to assumption generation have recently been proposed

Automated Assumption Generation

- Makes it easier to use and evaluate assume-guarantee reasoning
- We performed an preliminary investigation using an automated assumption generation technique

Preliminary Observations

- We observed a large variation in performance depending on the decomposition selected
- We found it difficult to select good decompositions
  - In more than half of our examples, the initial decompositions we selected did not save memory over monolithic FSV

- Goal:
  - Undertake an investigation to discover how to best select decompositions
- Approach:
  1. Examine all decompositions at the smallest reasonable system size to find the one that uses the least memory
  2. Generalize the best decomposition from the smallest reasonable system size to make it applicable at larger system sizes

Metrics

- To determine whether or not assume-guarantee reasoning offers an advantage over monolithic FSV:
  1. Compare the time used
  2. Compare the memory used
  3. Compare the size of the systems that can be verified

Methodology

- Used the assumption learning technique developed by Cobleigh, Giannakopoulou, and Păsăreanu [2003]
- Implemented the learning algorithm for two FSV tools
  - FLAVERS [Dwyer et al., 2004]
  - LTSA [Magee and Kramer, 1999]
- Looked at a number of scalable systems and properties drawn from the literature
  - Chiron, Gas Station, Peterson's mutual exclusion protocol, Relay, Smokers
    - At smallest size, systems have between 125 and 915 LOC
    - Up to 20,000 LOC at the largest system size

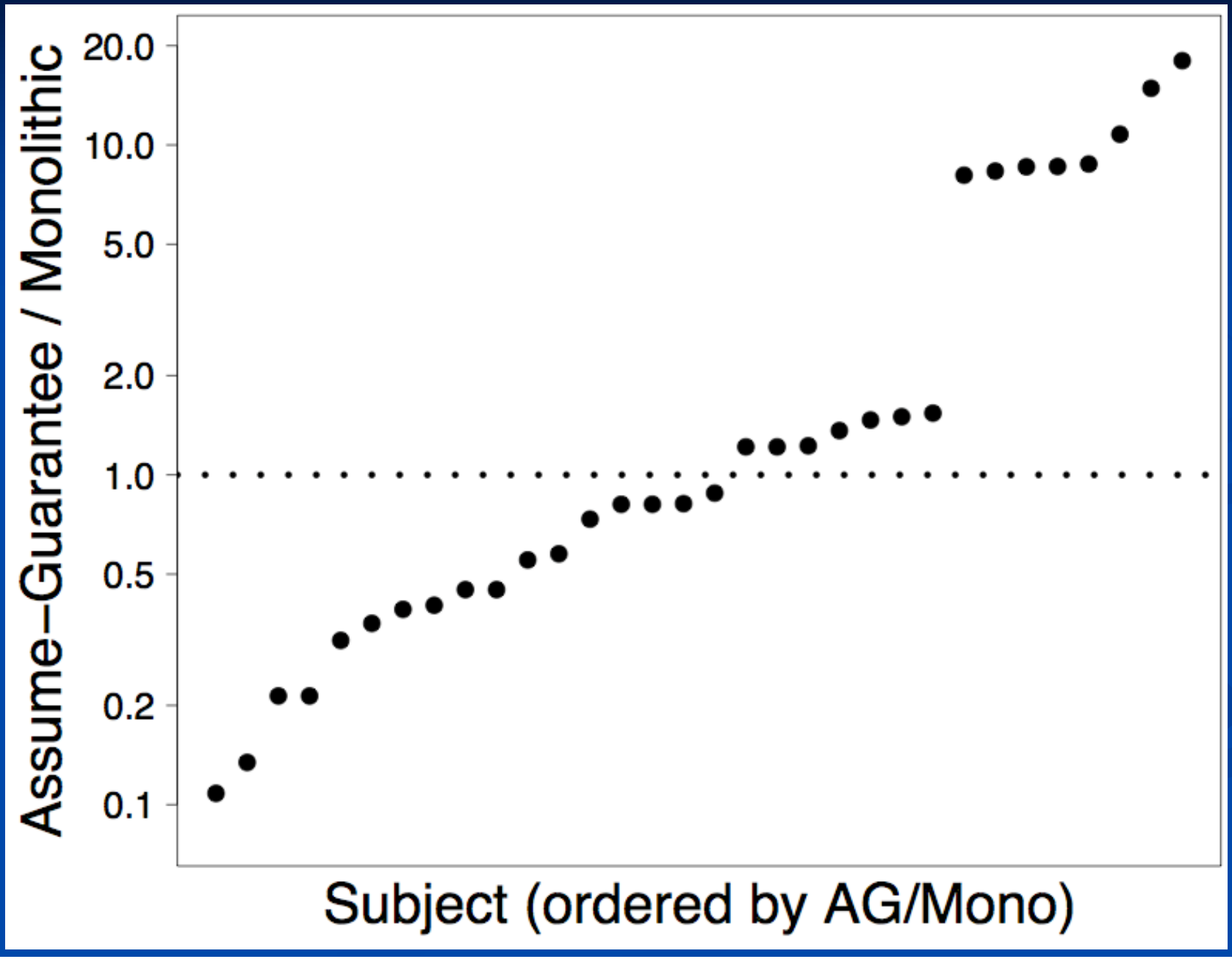
Methodology

- For each subject (a property/system pair) and verifier, examined all two-way decompositions at the smallest reasonable system size (size 2) to find the best one with respect to memory
- The systems are scaled by creating more instances of one particular task (e.g., customers in the Gas Station system)
  - The size is the number of occurrences of that task

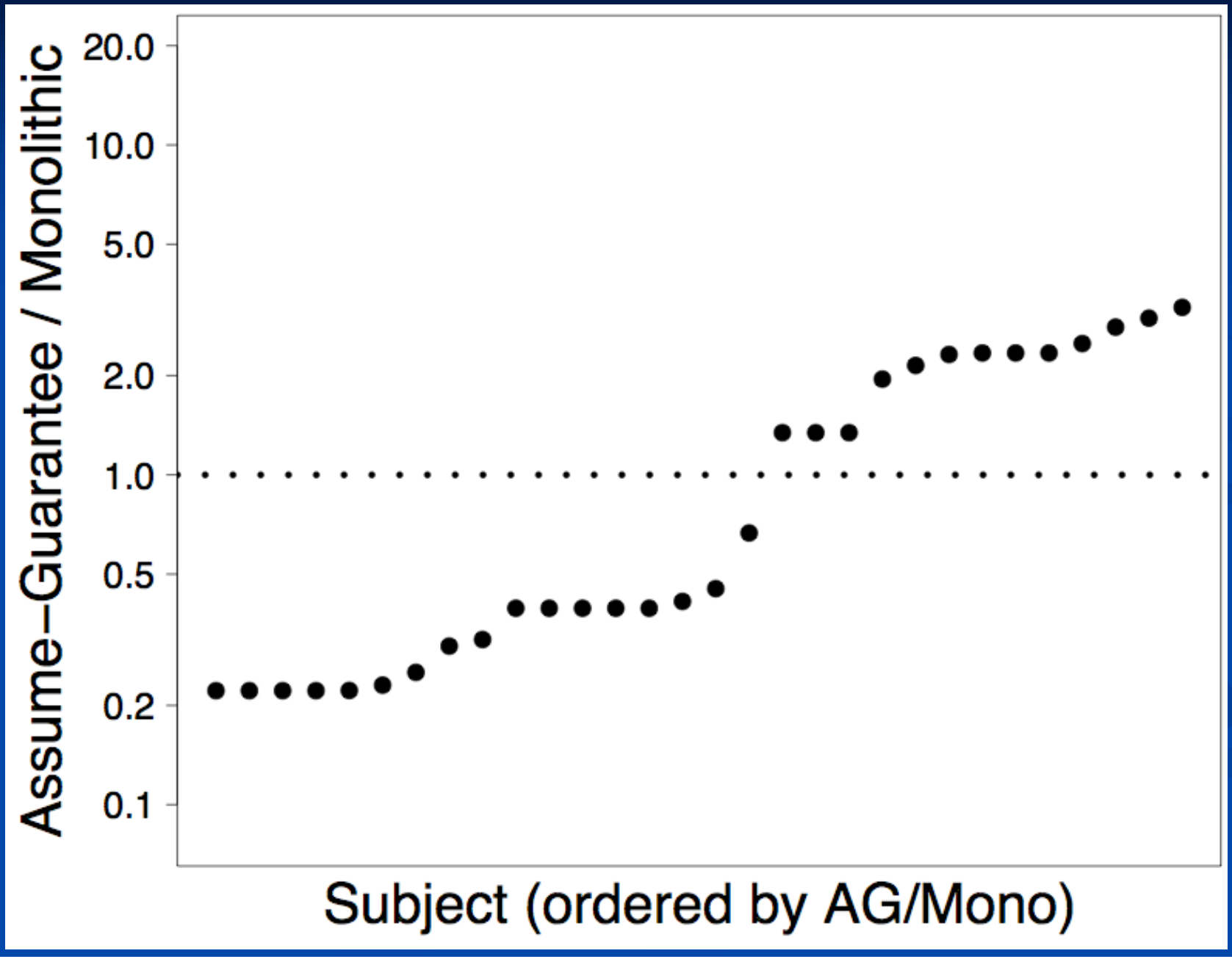
Results - Size 2

- For the vast majority of decompositions, assume-guarantee reasoning did not save memory over monolithic verification
- Looking at just the best decomposition (the decomposition that used the least memory) for each subject, the results are better

FLAVERS Memory Usage - Size 2



LTSA Memory Usage - Size 2



Summary - Best Size 2 Decomposition

- Assume-guarantee reasoning with the best decomposition is better than monolithic:
  - FLAVERS: 53% of the subjects
  - LTSA: 57% of the subjects
- For those subjects where the best decomposition saved memory, it used on average:
  - FLAVERS: 48% of the memory of monolithic
  - LTSA: 34% of the memory of monolithic

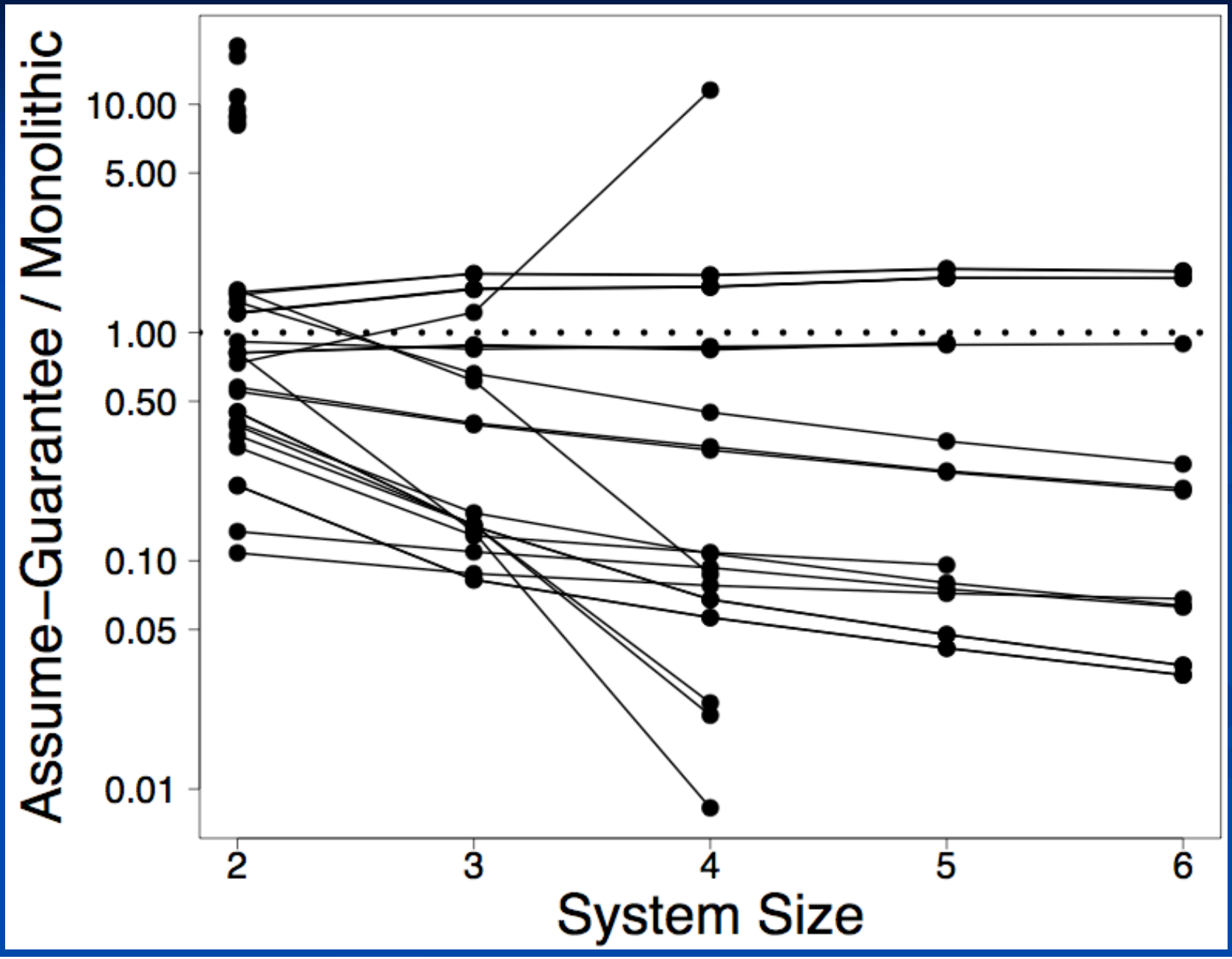
Another Way to View the Data

- Every decomposition uses more memory than monolithic on:
  - FLAVERS: 47% of the subjects
  - LTSA: 43% of the subjects
- We could not find heuristics to select a good decomposition or to determine when one exists
  - Needed to look at all decompositions

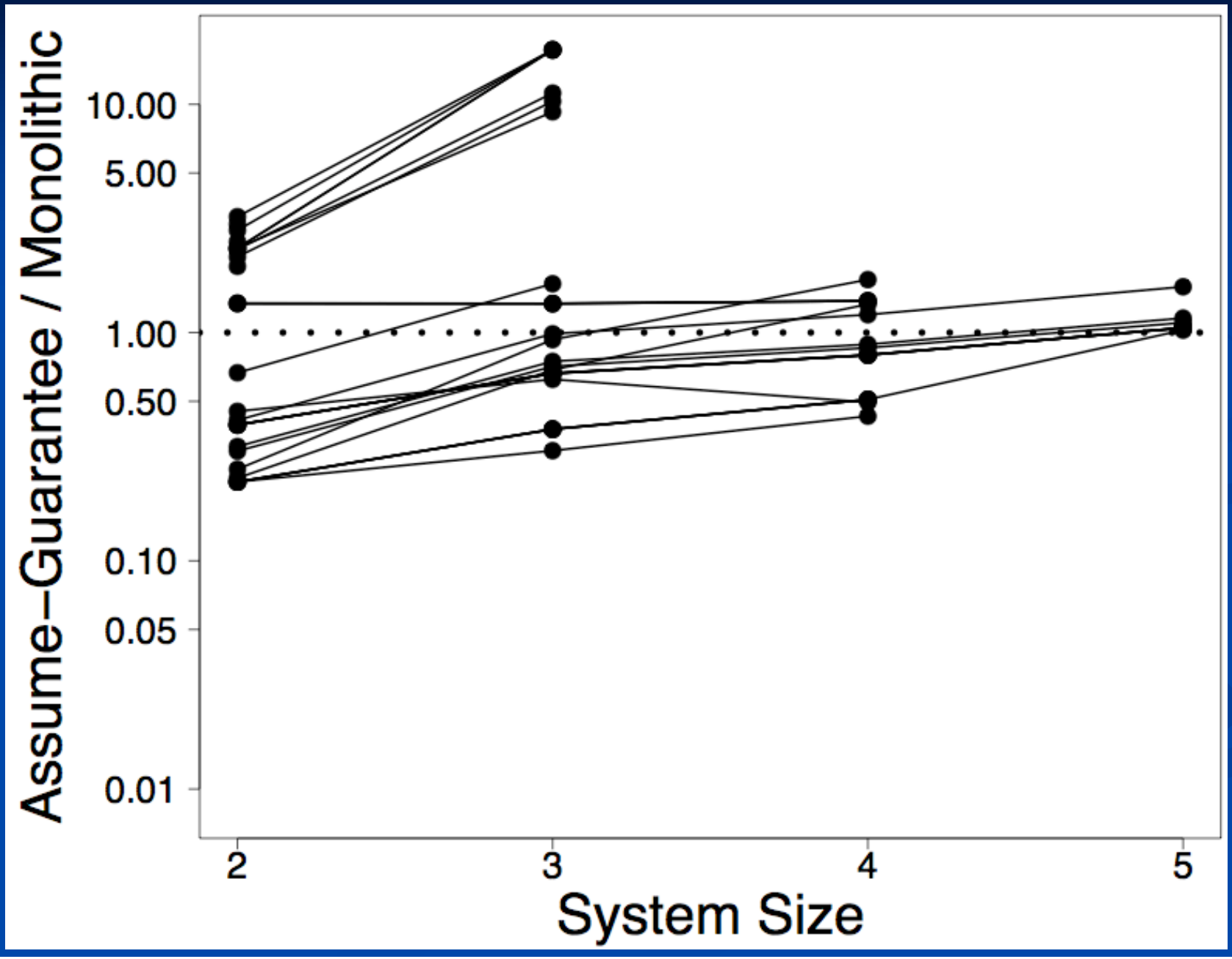
What Happens at Larger Sizes?

- Examining all decompositions at larger sizes quickly becomes infeasible
  - Generalized the best decomposition from size 2 to make it applicable for larger system sizes

FLAVERS Memory Usage - Larger Sizes



LTSA Memory Usage - Larger Sizes



Summary - Larger Sizes

- Assume-guarantee reasoning with a generalized decomposition is better than monolithic at the largest size a comparison could be made:
  - FLAVERS: 56% of the subjects
  - LTSA: 17% of the subjects
- When assume-guarantee reasoning with a generalized decomposition saved memory, at the largest size a comparison could be made, it used on average:
  - FLAVERS: 24% of the memory
  - LTSA: 49% of the memory

Scaling to Larger Sizes

- Is this memory savings enough to allow larger systems to be verified?
- Using generalized decompositions, assume-guarantee reasoning could scale further on:
  - FLAVERS: 7 of 32 subjects
    - 1 size larger on 5 subjects
    - 2 sizes larger on 1 subject
    - At least 3 sizes larger on 1 subject
  - LTSA: 0 of 30 subjects

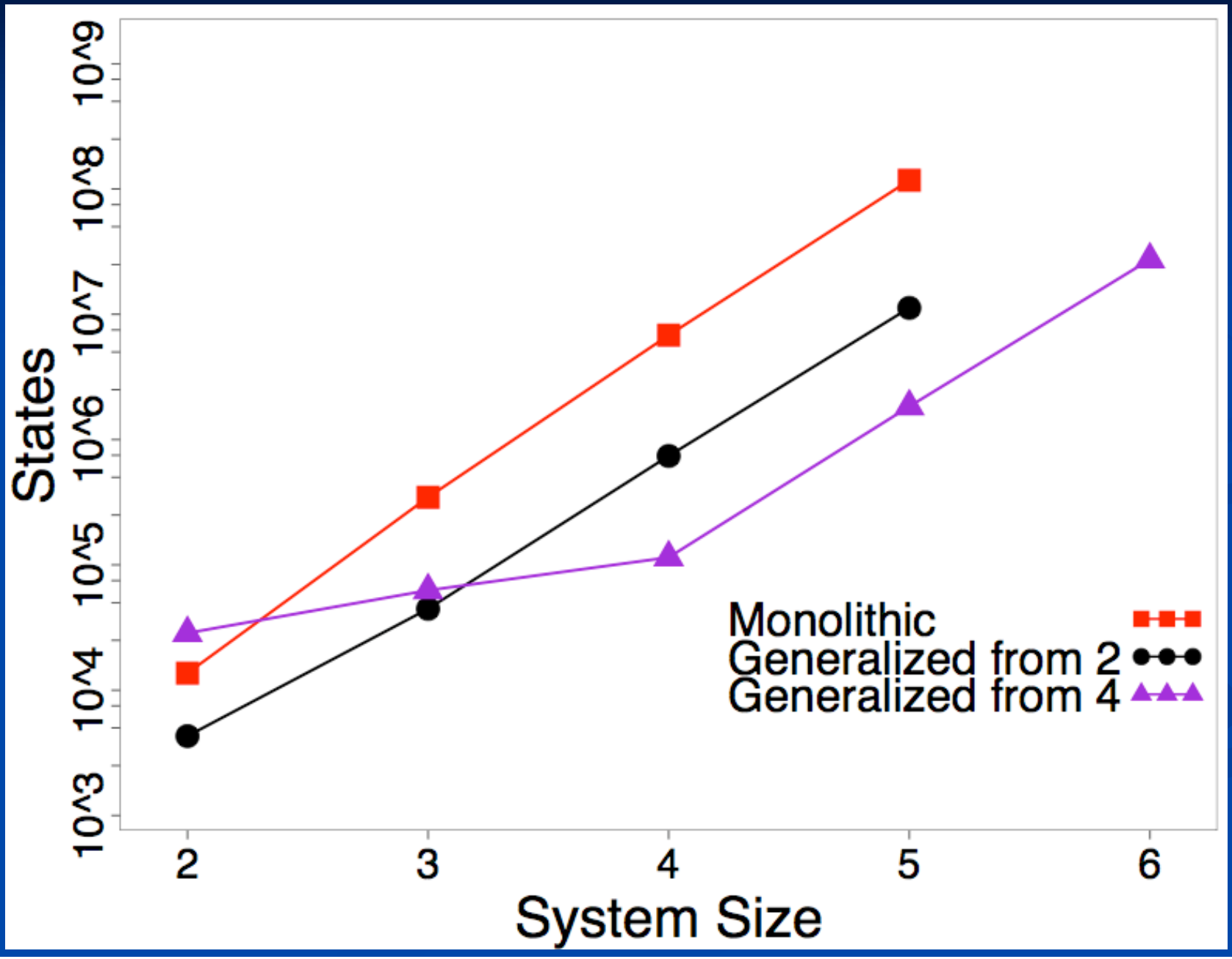
What About Other Decompositions?

- Previously, used just one decomposition, generalized from the best decomposition at size 2
- For all subjects, looked at all decompositions for at least one larger system size
  - As at size 2, the vast majority of decompositions used more memory than monolithic verification

What About Other Decompositions? (continued)

- For a little more than half of the subjects, there exists at least one decomposition that is better than the generalized one
- Compared to the decompositions generalized from size 2:
  - Three of these, when generalized, allow larger systems to be verified
  - These three only scale one size larger
- Other decompositions might do even better
  - But, we do not know of a good way to find them

Gas Station - FLAVERS



Results of Empirical Study

- Assume-guarantee reasoning usually uses more time
- Assume-guarantee reasoning sometimes saves memory
- Assume-guarantee reasoning rarely allows properties to be verified on larger systems
  - In these cases, only one or two sizes larger
- Finding the best decompositions at small system sizes usually did not help in finding the best decomposition at larger system sizes

Threats to Validity

- The scope of these experiments is limited, only looked at:
  - One assume-guarantee rule
  - One automated assumption generation technique
  - Two FSV tools
  - A small number of somewhat artificial systems

But, performed a larger empirical study than has been done in the past

Conclusions

- The vast majority of decompositions use more memory than monolithic verification
  - Considering just the best decomposition assume-guarantee reasoning saved memory for about half the subjects
- We did not find good heuristics for selecting a decomposition
- In only 7 out of 62 subjects could assume-guarantee reasoning verify larger systems than monolithic verification

Conclusions

- Our results raise concerns about the usefulness of assume-guarantee reasoning
  - More experimentation is needed
  - Future work in this area should provide convincing experimental evidence

What Went Wrong?

$$\frac{\langle A \rangle\ S1\ \langle P \rangle \qquad \langle true \rangle\ S2\ \langle A \rangle}{\langle true \rangle\ S1 \parallel S2\ \langle P \rangle}$$

- The assumption A is usually smaller than S2
  - The assumption A usually allows more behaviors than S2
  - Causes the memory needed to check <A> S1 <P> to be greater than memory need to check <true> S1 || S2 <P>
- The assumption A is usually larger than the property P
  - This increases the memory needed to check <true> S2 <A>