On Uniformly Sampling Traces of a Transition System

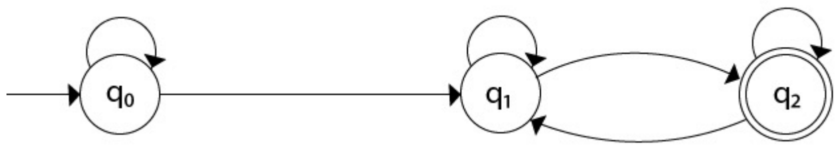
Supratik Chakraborty, **Aditya A. Shrotri**, Moshe Y. Vardi

Model-Counting Workshop 2021

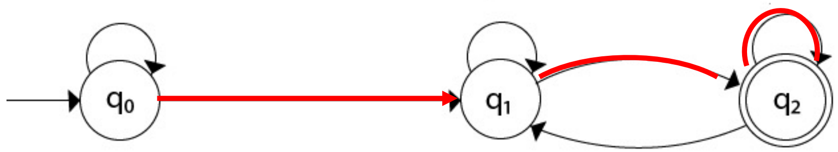
Appeared in proceedings of ICCAD 2020

Uniform Trace Sampling

- **Given:** State Transition Graph, $n \in \mathbb{N}$

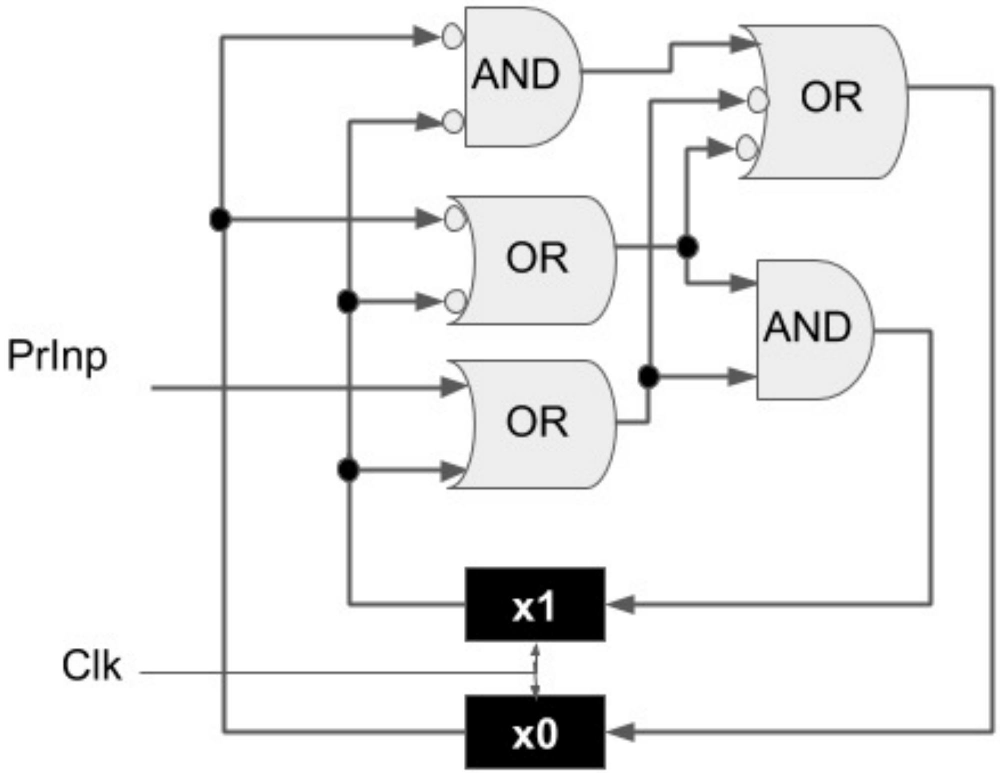


- **Decision:** Is there a path of length ‘n’ from initial state to final state?
  - n=3

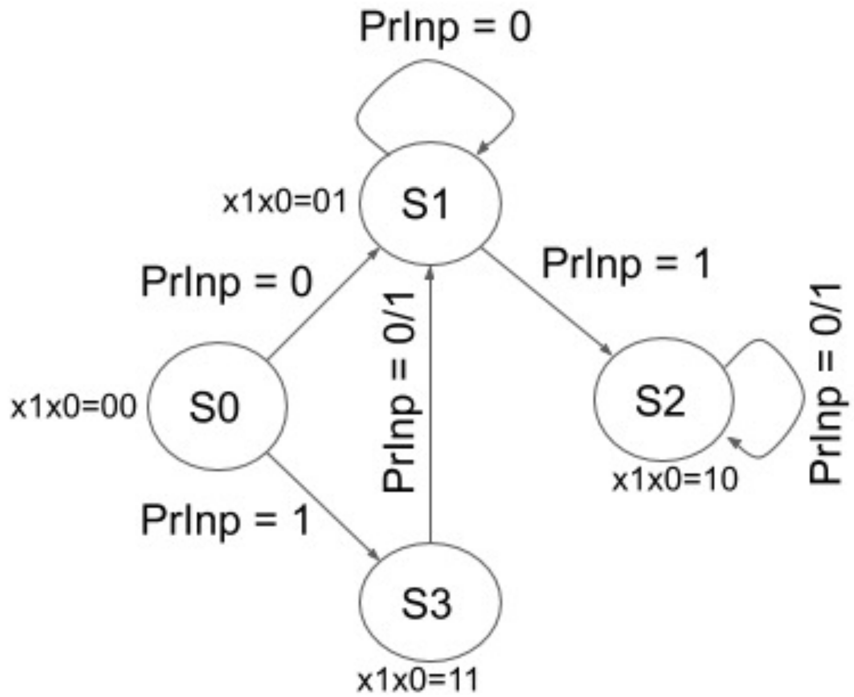


- **Counting:** How many paths of length ‘n’?
  - 3 paths of length 3
- **Sampling:** Generate a random path of length 3 with Pr = 1/3
- **Applications:** Verification of Sequential Circuits

Example: States, Traces and Uniformity

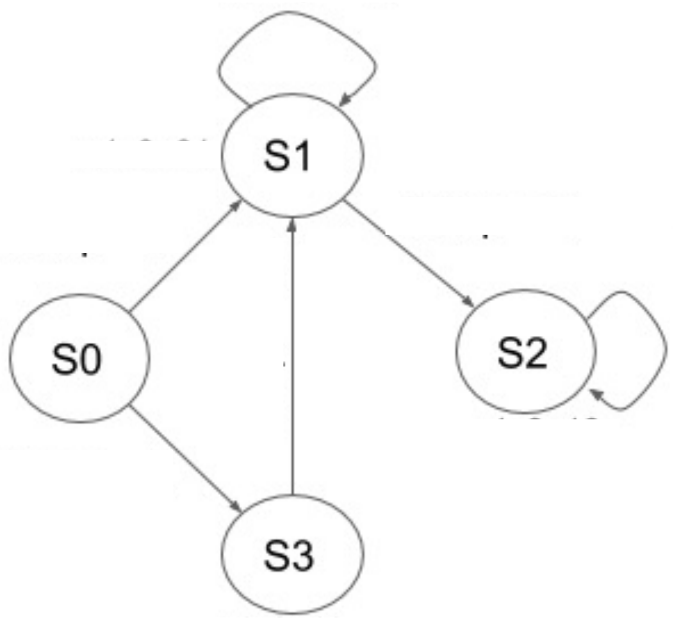


(a)



(b)

Example: States, Traces and Uniformity



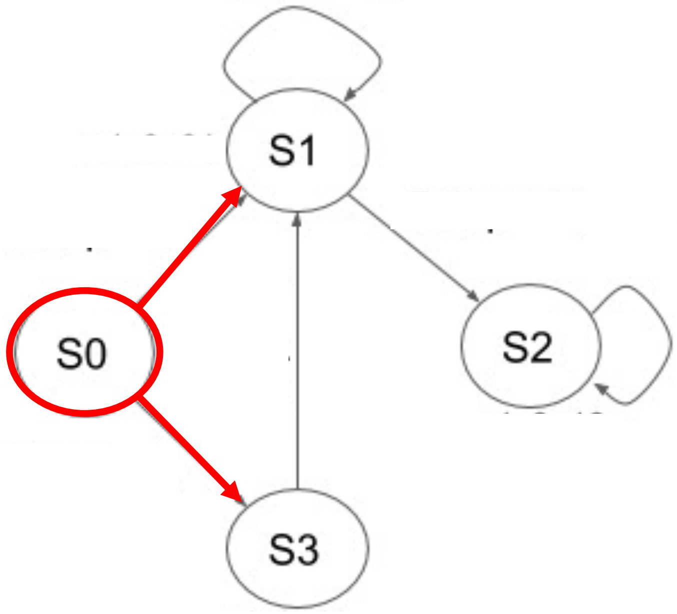
(b)

Traces with N = 4 transitions (5 states):

1. $s_0 s_1 s_1 s_1 s_1$
2. $s_0 s_1 s_1 s_1 s_2$
3. $s_0 s_1 s_1 s_2 s_2$
4. $s_0 s_1 s_2 s_2 s_2$
5. $s_0 s_3 s_1 s_1 s_1$
6. $s_0 s_3 s_1 s_1 s_2$
7. $s_0 s_3 s_1 s_2 s_2$

Uniformity: Sample each trace with probability 1/7

Example: Insufficiency of Local Uniformity



(b)

Current State: S_0

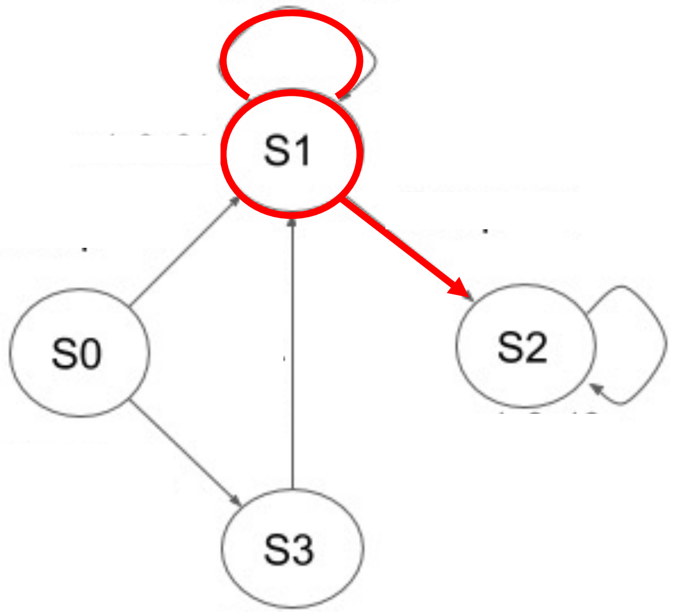
Trace: S_0

Probability: 1

Next State Probabilities:

S_3	**0.5**
S_1	**0.5**

Example: Insufficiency of Local Uniformity



(b)

Current State: S_0

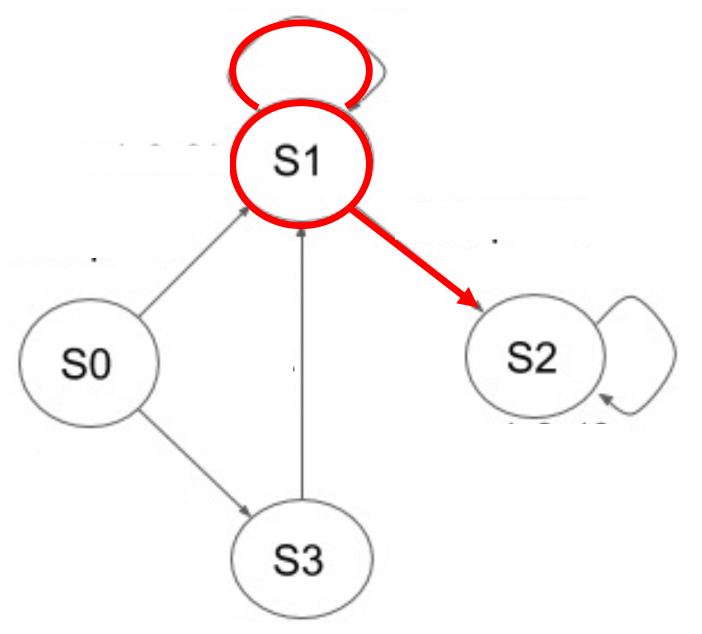
Trace: S_0 S_1

Probability: 1*0.5

Next State Probabilities:

S_2	0.5
S_1	0.5

Example: Insufficiency of Local Uniformity



(b)

Current State: S_0

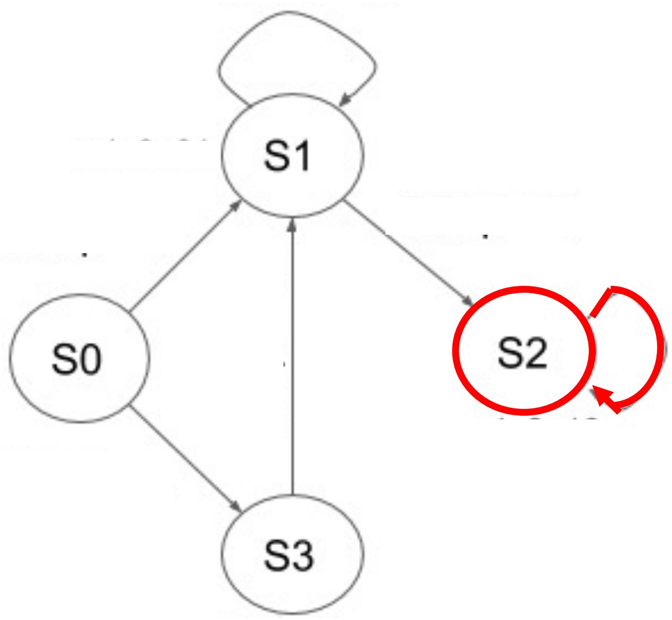
Trace: S_0 S_1 S_1

Probability: 1*0.5*0.5

Next State Probabilities:

S_2	0.5
S_1	0.5

Example: Insufficiency of Local Uniformity



(b)

Current State: S_0

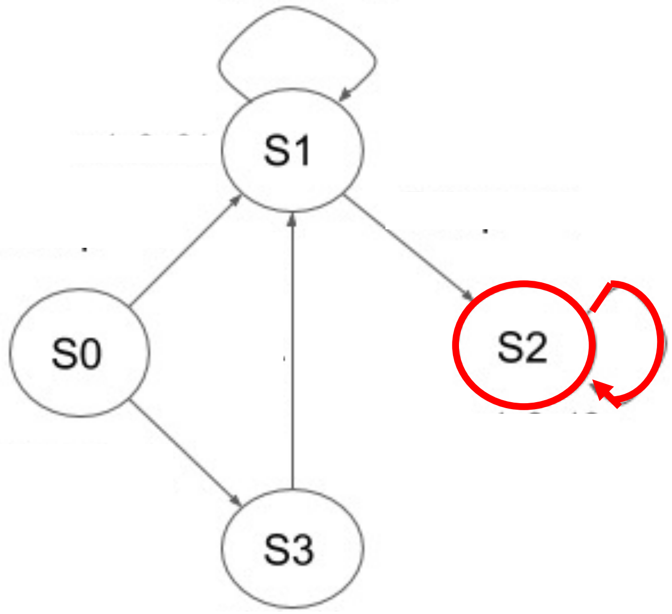
Trace: $S_0\ S_1\ S_1\ S_2$

Probability: 1*0.5*0.5*0.5

Next State Probabilities:

S_2	**1**

Example: Insufficiency of Local Uniformity



(b)

Current State: S_0

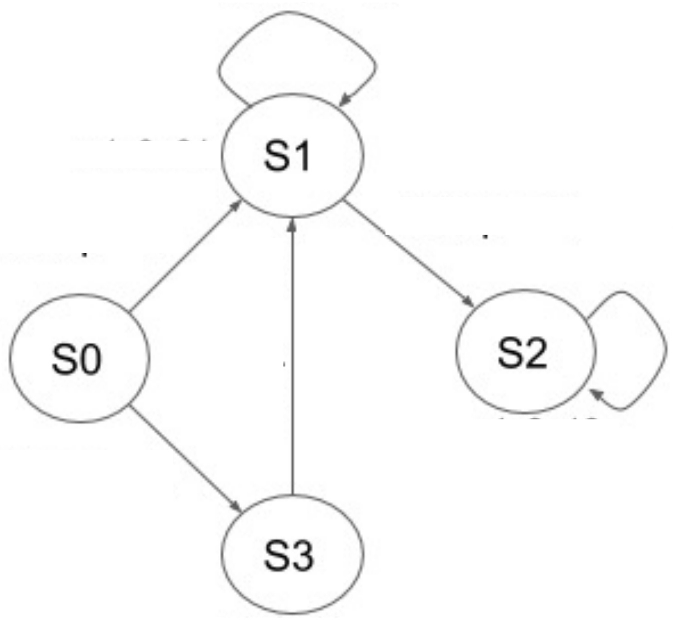
Trace: S_0 S_1 S_1 S_2 S_2

Probability: 1*0.5*0.5*0.5*1 = 0.125

Next State Probabilities:

S_2	**1**

Example: Insufficiency of Local Uniformity



(b)

Current State: S_0

Trace: S_0 S_1 S_1 S_2 S_2

Probability: 1*0.5*0.5*0.5*1 = 0.125

Fact: Pr = 1/7 not possible for **any** assignment of local probabilities

Existing Approaches

1. CNF-Sampling
   - SAT solvers are best tools for reachability (decision problem)
   - Surprisingly CNF Samplers fail to scale
     - Top-Down approach cannot exploit structure in underlying graph
     - Inadequate for global reasoning
2. Adjacency Matrix Squaring

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \qquad A^2 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 3 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \qquad \ldots \; A^n \rightarrow \; \#Paths\ of\ Length\ n$$

#Paths of Length 1

#Paths of Length 2

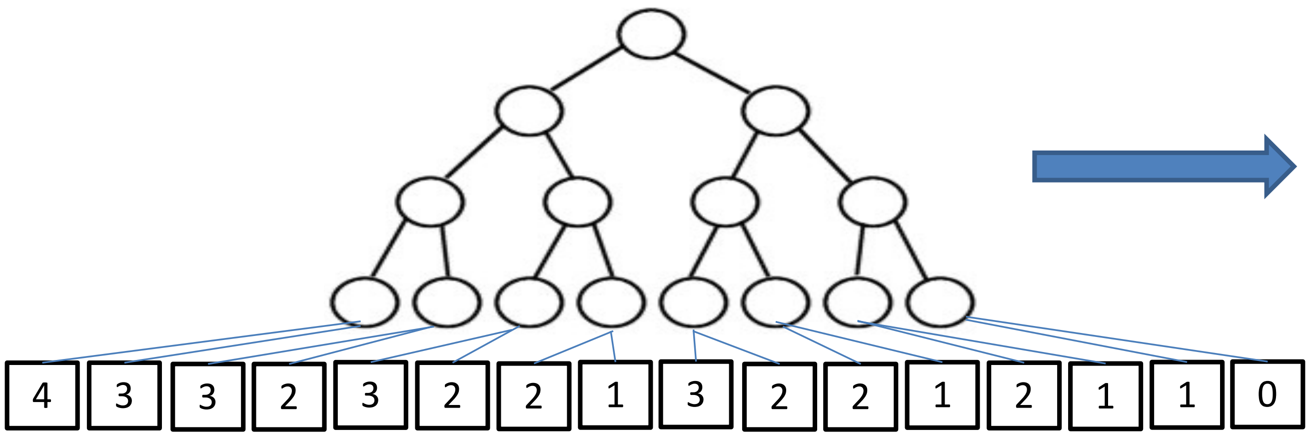
Our Contributions

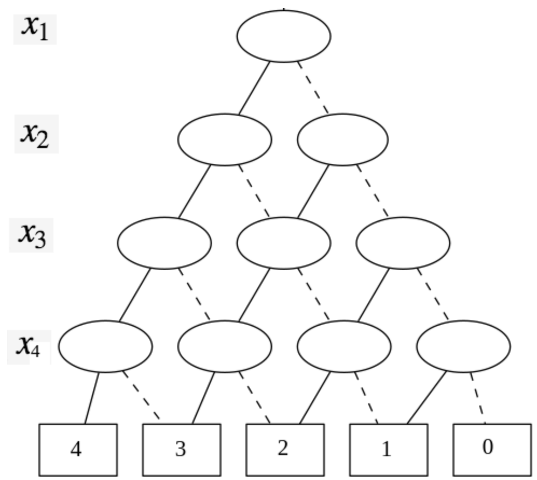
- **TraceSampler:** 1st dedicated algorithm + tool for uniformly sampling traces of a transition system
  - Uses Algebraic Decision Diagrams (ADDs) & enhanced iterative-squaring
  - Easily extensible to weighted sampling

- Empirical comparison to generic samplers based on SAT/CDCL
  - TraceSampler is fastest on ~90% of benchmarks
  - Solves 200 more benchmarks than nearest competitor

Algebraic Decision Diagrams

- Data Structures for compact representation of Real-valued Boolean functions $f: \{0,1\}^n \rightarrow \mathbb{R}$
  - DAGs with fixed variable order and node-sharing
  - Operations: Sum, Product, Additive Quantification ($\sum$), ITE
- Ex: $f = x_1 + x_2 + x_3 + x_4$





Benefits of ADDs

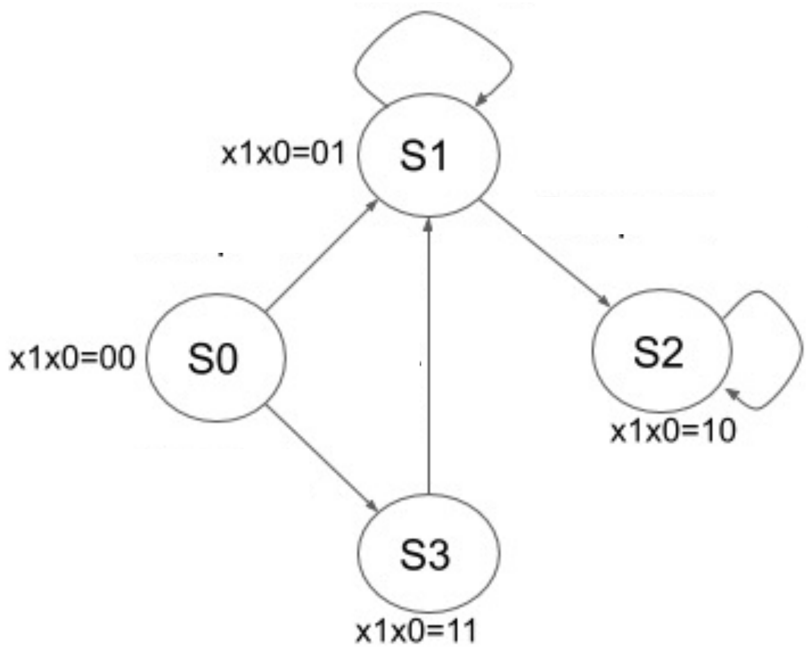
- ✓ Compact and canonical
  - Naturally embody dynamic programming
- ✓ Polynomial-time operations (addition, multiplication, ITE etc)
  - Enable bottom-up compositional approach
- ✓ Smart heuristics for variable ordering
  - Scalability on real-world instances
- ✓ Fast libraries for serial and parallel computation
  - Eliminates redundant effort

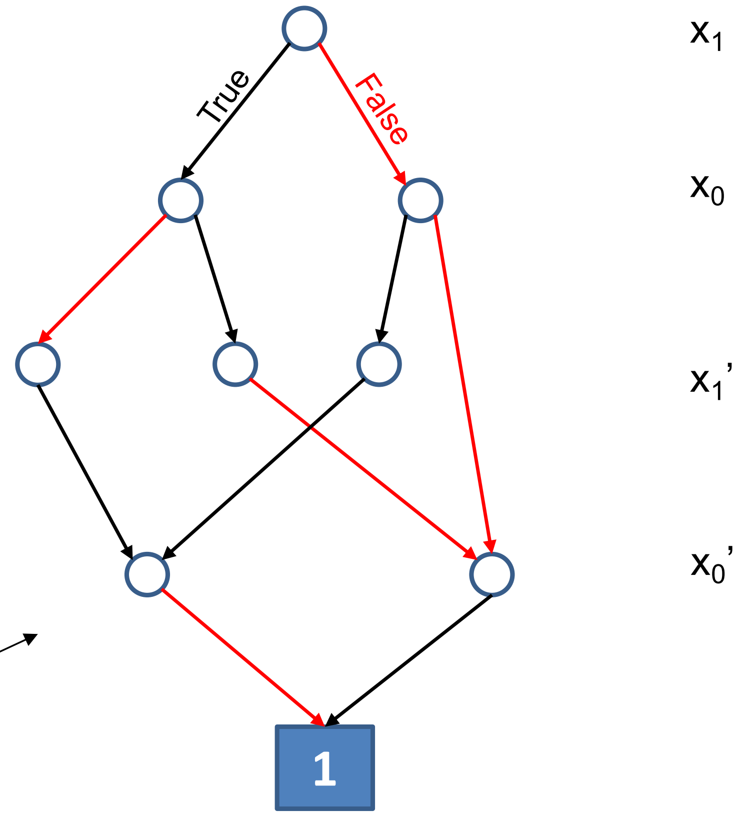
TraceSampler: Two-Phase Algorithm

- Compilation Phase:
  - Compose n-step transition relation t_N sequentially
  - Construct $\log N$ ADDs: $t_1, t_2, t_4, t_8, \dots, t_N$ by iterative-squaring
    - Aggressively prune ADDs to avoid blowup

- Sampling Phase: Divide & Conquer
  - Recursively split trace while ensuring global uniformity
  - Base case: random walk on ADD from root to leaf

Example: 1-step transition function

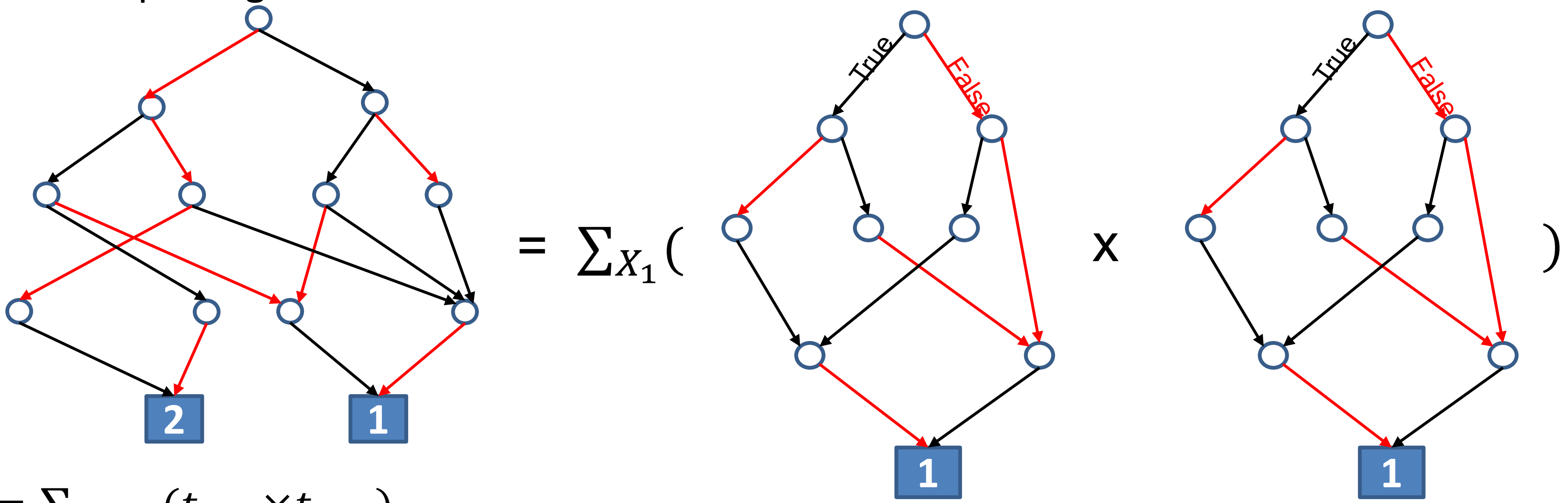




Represents 1-Step Transition Function

TraceSampler: ADD Compilation Phase

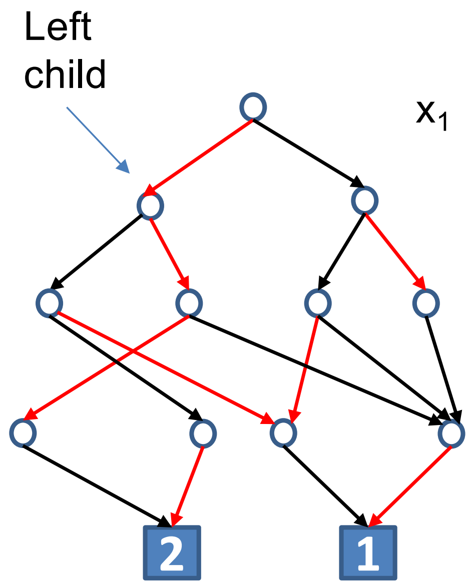
- Iterative-Squaring:



- $t_N = \sum_{X_{N/2}} (t_{N/2} \times t_{N/2})$
- Secret Sauce: Aggressive pruning of ADDs by *novel i-step reachability algorithm*
- Advantages:
  - Only log(N) ADDs necessary: $t_1, t_2, t_4, t_8, \ldots, t_N$
  - Factored forms offer significant speedup & compression [Dudek et al.'20]

TraceSampler: Sampling Phase

- Base case: sample states from ADD
  - Weighted random walk on ADD
  - Root to leaf traversal
    - $\Pr[X_1 = False] = \frac{wt(left\ child)}{wt(left\ child)+wt(right\ child)}$



TraceSampler: Sampling Phase

- Recursive Step
  - Sample state at half-way point then sample two halves independently

Trace Position	0	1	2	…	N/4	…	N/2	…	3N/4	…	N
State											

TraceSampler: Sampling Phase

- Recursive Step
  - Sample state at half-way point then sample two halves independently

Trace Position	0	1	2	…	N/4	…	N/2	…	3N/4	…	N
State											

log Nth ADD: t_N

TraceSampler: Sampling Phase

- Recursive Step
  - Sample state at half-way point then sample two halves independently

Trace Position	0	1	2	…	N/4	…	N/2	…	3N/4	…	N
State	S_0						S_{10}				S_5

log Nth ADD: t_N

TraceSampler: Sampling Phase

- Recursive Step
  - Sample state at half-way point then sample two halves independently

Trace Position	0	1	2	…	N/4	…	N/2	…	3N/4	…	N
State	S_0						S_{10}				S_5

log N -1 ADD: $t_{N/2}$

TraceSampler: Sampling Phase

- Recursive Step
  - Sample state at half-way point then sample two halves independently

Trace Position	0	1	2	…	N/4	…	N/2	…	3N/4	…	N
State	S_0				S_{11}		S_{10}		S_8		S_5

log N -1 ADD: $t_{N/2}$

TraceSampler: Sampling Phase

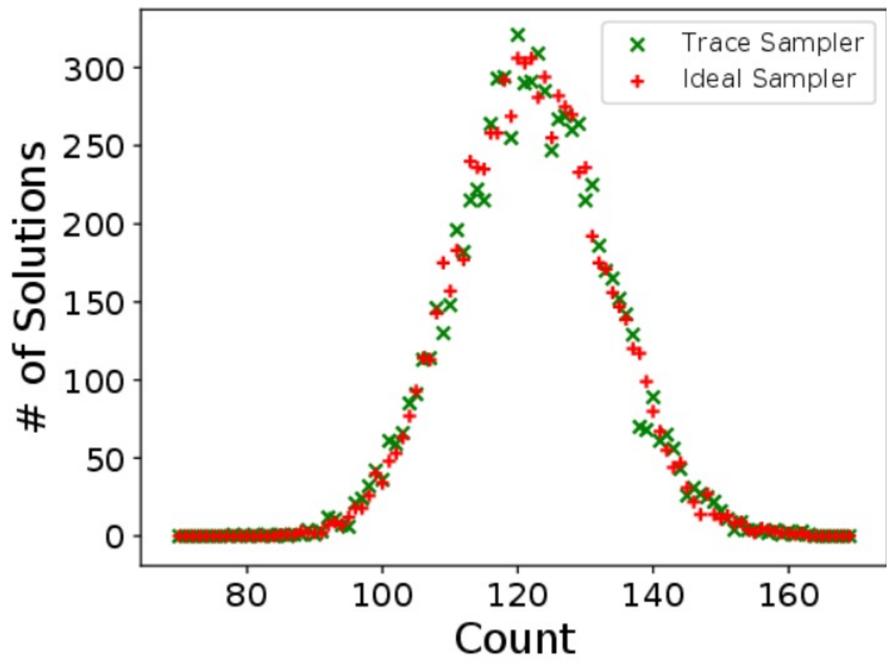
- Recursive Step
  - Sample state at half-way point then sample two halves independently

Trace Position	0	1	2	…	N/4	…	N/2	…	3N/4	…	N
State	S_0				S_{11}		S_{10}		S_8		S_5

log N -2 ADD: $t_{N/4}$

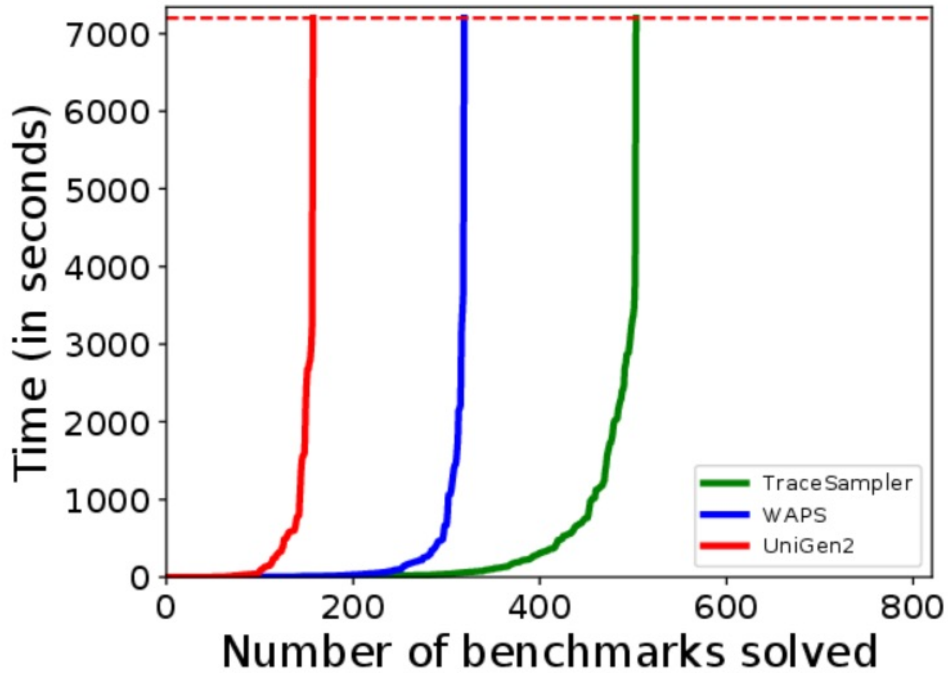
Empirical Evaluation: Uniformity

- Sampled 10^6 traces from small benchmark
  - Using TraceSampler
  - Using Ideal Sampler
- **X-axis**
  - Count of how many times a particular trace was sampled
- **Y-axis**
  - Number of traces with specific count
- Distributions are indistinguishable
  - Jensen-Shannon distance: 0.003



Empirical Evaluation: Scalability

- **Benchmarks**: HWMCC'17, ISCAS89
- **Trace Lengths**: 2,4,8,16,…256
- **Comparison**: Encode circuits as CNF and unroll
  - WAPS: Exact uniform sampler [Gupta et al. '19]
  - Unigen2: Approximately uniform sampler
    - [Chakraborty et al. '15]
- **Results**:
  - **TraceSampler** solves 200+ more instances
  - Fastest on ~90% instances
  - Avg. Speedup: 3x to WAPS, 25x to Unigen2
  - Compilation Speedup: 16x to WAPS



Summary and Takeaways

- **TraceSampler**: Novel ADD based algorithm for uniform / weighted sampling of traces
  - Significantly outperforms competing SAT/CDCL-based approaches
  - First prototype; more engineering effort ➔ more scalability
  - Scope for heuristics and time-space tradeoffs
- Too early to write off Decision Diagrams?
  - ADDs vastly outperform CNF-counters for #Perfect-Matchings
    - [Chakraborty, **Shrotri**, Vardi '19]
  - Many variations not fully explored

https://gitlab.com/Shrotri/tracesampler https://cs.rice.edu/~as128/

References

- [Dudek et al., ‘20] Jeffrey M Dudek, Vu HN Phan, and Moshe Y Vardi. AAAI 2020. ADDMC: Exact weighted model counting with algebraic decision diagrams
- [Gupta et al., 19] Rahul Gupta, Shubham Sharma, Subhajit Roy, and Kuldeep S Meel. 2019. Waps: Weighted and projected sampling. In International Conference on Tools andAlgorithms for the Construction and Analysis of Systems. Springer, 59–76
- [Chakraborty et al., ‘15] Supratik Chakraborty, Daniel J Fremont, Kuldeep S Meel, Sanjit A Seshia, and Moshe Y Vardi. 2015. On parallel scalable uniform SAT witness generation. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 304–319.
- [Chakraborty, Shrotri, Vardi ‘19] "On Symbolic Approaches for Computing the Matrix Permanent." *International Conference on Principles and Practice of Constraint Programming*. Springer, Cham, 2019.

Speaker Bio

- Speaker: **Aditya A. Shrotri**
  - Affiliation: Rice University, Houston TX
  - PhD Student (Dept. of Computer Science)
  - Adviser: Prof. Moshe Y. Vardi
  - Thesis Area: Constrained Sampling and Counting
  - Webpage: https://cs.rice.edu/~as128

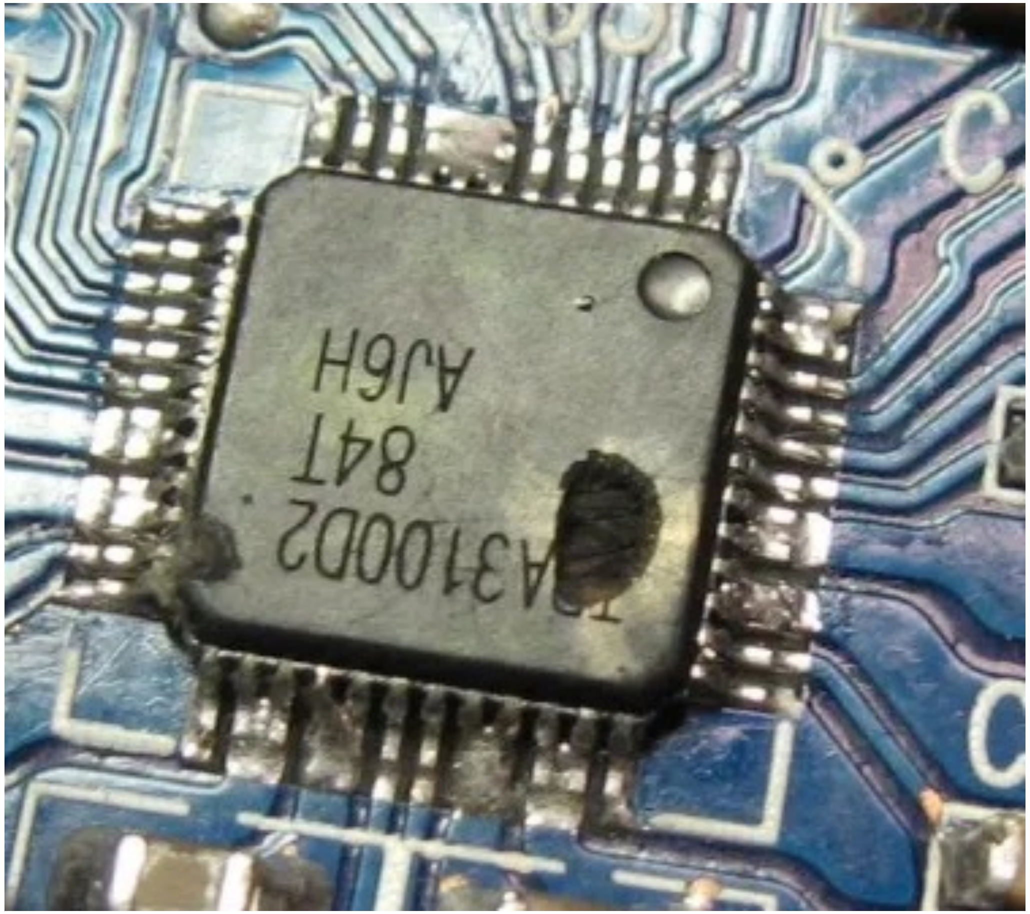
- Co-Authors:
  - **Prof. Supratik Chakraborty** (IIT Bombay, India)
    - https://www.cse.iitb.ac.in/~supratik/

Prof. Moshe Y. Vardi (Rice University, Houston)
https://www.cs.rice.edu/~vardi/

Backup

Correctness of large designs

- Enormous size and complexity of modern digital systems
  - Formal verification fails to scale
- Important to catch bugs early
  - Millions of dollars spent on faulty designs
- Constrained Random Verification balances scalability and coverage



Constrained Random Verification

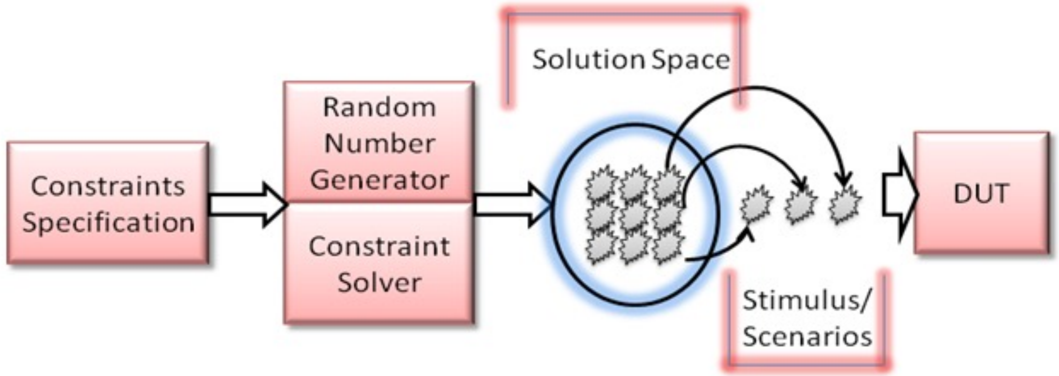


Diagram courtesy www.testbench.in

- Constraints give direction
  - User-defined constraints steer to bug-prone corners
- Randomization enables diversity
  - Inputs sampled at specific simulation steps
- Widely used in industry
  - Ex: SystemVerilog, E, OpenVera etc.

Limitations of Existing CRV Tools

- Provide ‘local’ uniformity over input stimuli

- Insufficient for ‘global’ coverage guarantees

- *Need uniformity of system’s runs or traces*