# CSB: A Counting and Sampling tool for Bitvectors

Arijit Shaw[1], Kuldeep S. Meel[2]

[1] Chennai Mathematical Institute, India

[2] University of Toronto, Canada

## Problem Statement

### Counting

Bitvector formula F on variables X

Sol(F) set of assignment on X, that satisfy F

Determine the value of |Sol(F)|

## Problem Statement

**Counting**

Bitvector formula F on variables X

Sol(F) set of assignment on X, that satisfy F

Determine the value of |Sol(F)|

Also happy with approximation

$$\Pr\left[\frac{|\mathsf{Sol(F)}|}{1+\varepsilon} \leq c \leq (1+\varepsilon)|\mathsf{Sol(F)}|\right] \geq 1-\delta$$

## Problem Statement

### Counting

Bitvector formula F on variables X

Sol(F) set of assignment on X, that satisfy F

Determine the value of |Sol(F)|

Also happy with approximation

$$\Pr\left[\frac{|\mathsf{Sol}(\mathsf{F})|}{1+\varepsilon} \leq c \leq (1+\varepsilon)|\mathsf{Sol}(\mathsf{F})|\right] \geq 1-\delta$$

### Uniform **Sampling**

Each invocation will return

$$\sigma \in \mathsf{Sol}(\mathsf{F})$$

Such that, $\Pr(\sigma) = \dfrac{1}{|\mathsf{Sol}(\mathsf{F})|}$

## Problem Statement

### Counting

Bitvector formula F on variables X

Sol(F) set of assignment on X, that satisfy F

Determine the value of |Sol(F)|

Also happy with approximation

$$\Pr\left[\frac{|\mathsf{Sol(F)}|}{1+\varepsilon} \leq c \leq (1+\varepsilon)|\mathsf{Sol(F)}|\right] \geq 1-\delta$$

### Uniform **Sampling**

Each invocation will return

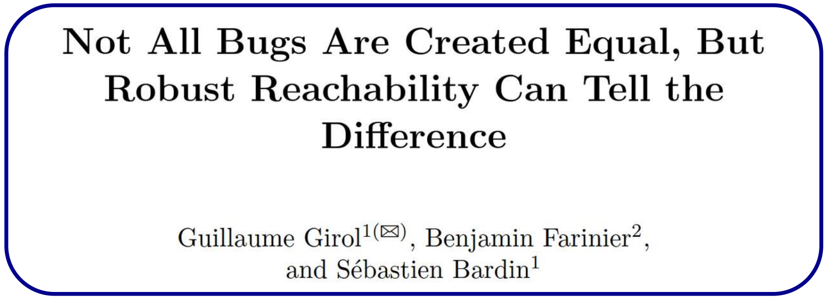
$$\sigma \in \mathsf{Sol(F)}$$

Such that, $\Pr(\sigma) = \dfrac{1}{|\mathsf{Sol(F)}|}$

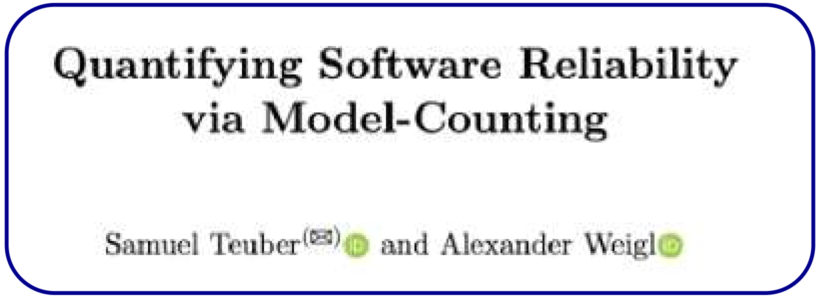
Also, approximation works

$$\frac{1}{(1+\varepsilon)|\mathsf{Sol(F)}|} \leq \Pr(\sigma) \leq \frac{(1+\varepsilon)}{|\mathsf{Sol(F)}|}$$

## Why counting and sampling in bit-vectors?

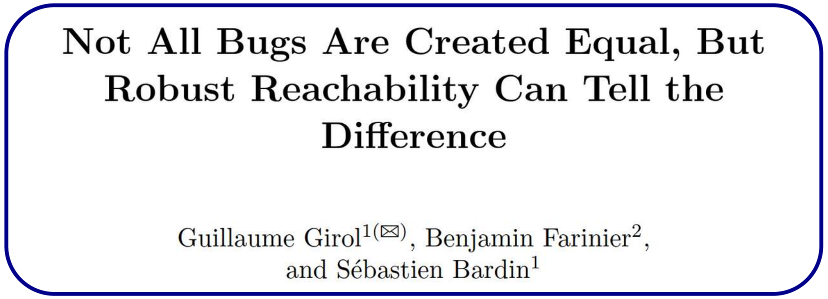

Not All Bugs Are Created Equal, But Robust Reachability Can Tell the Difference

Guillaume Girol[1(✉)], Benjamin Farinier[2], and Sébastien Bardin[1]


Quantifying Software Reliability via Model-Counting

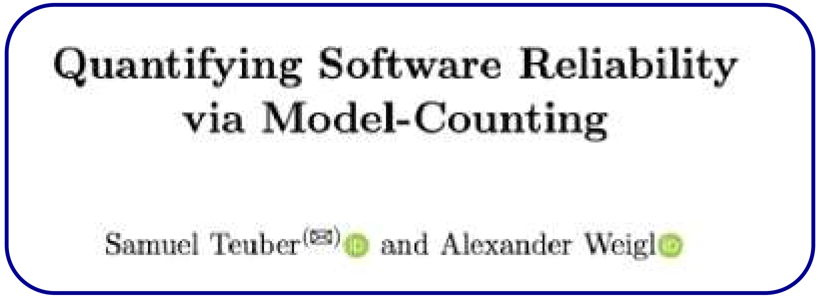
Samuel Teuber[(✉)] and Alexander Weigl

## Why counting and sampling in bit-vectors?



Not All Bugs Are Created Equal, But Robust Reachability Can Tell the Difference

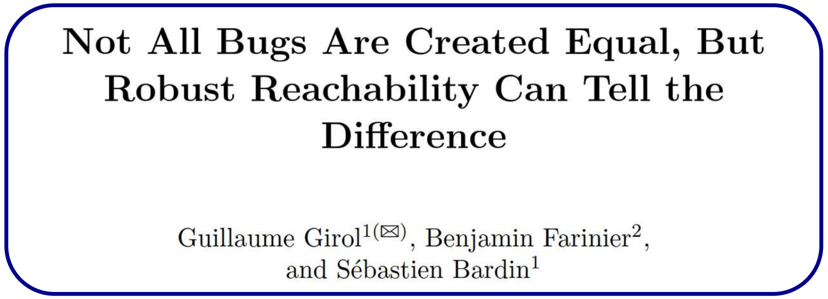
Guillaume Girol[1(✉)], Benjamin Farinier[2], and Sébastien Bardin[1]



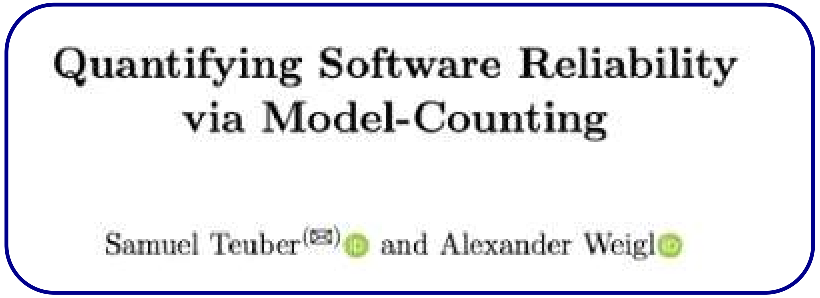
Quantifying Software Reliability via Model-Counting

Samuel Teuber[(✉)] and Alexander Weigl

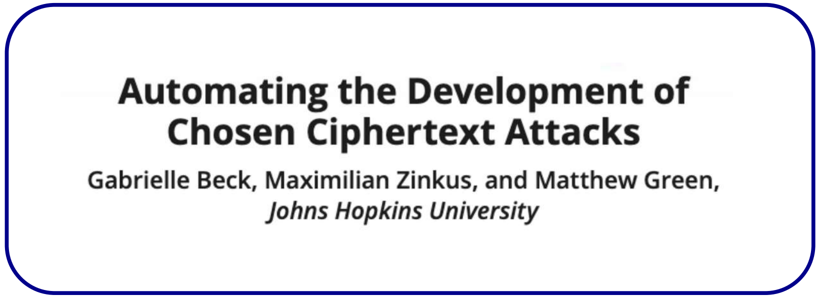
## Why counting and sampling in bit-vectors?


Not All Bugs Are Created Equal, But Robust Reachability Can Tell the Difference

Guillaume Girol[1](✉), Benjamin Farinier[2], and Sébastien Bardin[1]

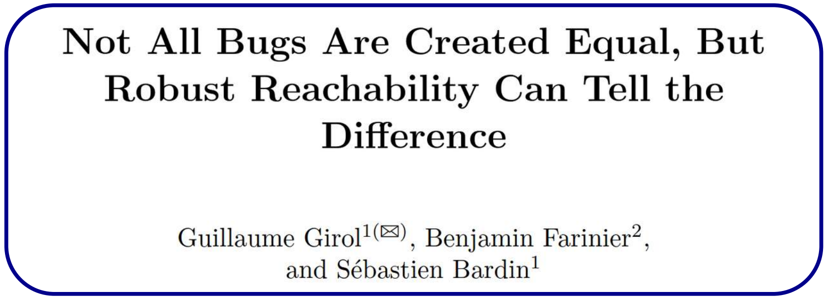

Quantifying Software Reliability via Model-Counting

Samuel Teuber(✉) and Alexander Weigl

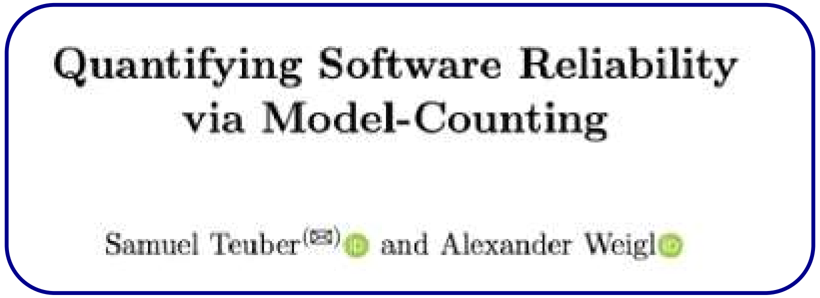

Automating the Development of Chosen Ciphertext Attacks

Gabrielle Beck, Maximilian Zinkus, and Matthew Green, *Johns Hopkins University*

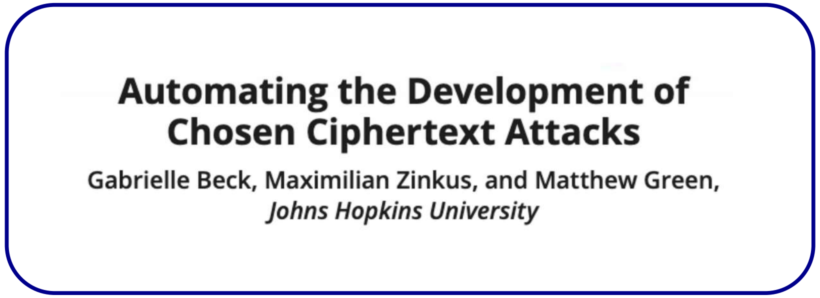
## Why counting and sampling in bit-vectors?


Not All Bugs Are Created Equal, But Robust Reachability Can Tell the Difference

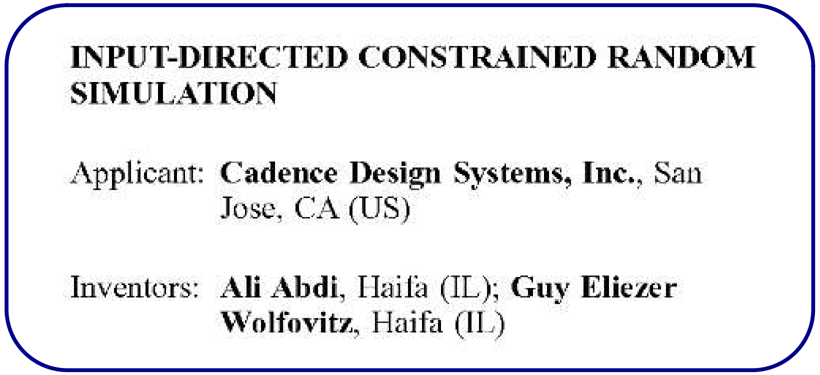
Guillaume Girol[1(✉)], Benjamin Farinier[2], and Sébastien Bardin[1]


Quantifying Software Reliability via Model-Counting

Samuel Teuber[(✉)] and Alexander Weigl


Automating the Development of Chosen Ciphertext Attacks

Gabrielle Beck, Maximilian Zinkus, and Matthew Green, *Johns Hopkins University*


INPUT-DIRECTED CONSTRAINED RANDOM SIMULATION

Applicant: **Cadence Design Systems, Inc.**, San Jose, CA (US)

Inventors: **Ali Abdi**, Haifa (IL); **Guy Eliezer Wolfovitz**, Haifa (IL)

## Current state of the art

### Counting

- SMTApproxMC [CMMV '16]
- SMC [KM '19]
- SearchMC [KM '18]

## Current state of the art

**Counting**

- SMTApproxMC [CMMV '16]
- SMC [KM '19]
- SearchMC [KM '18]

**Sampling**

## Current state of the art

**Counting**

- SMTApproxMC [CMMV '16]
- SMC [KM '19]
- SearchMC [KM '18]

**Sampling**

- SMTSampler [DLBS'18]
- GuidedSampler [DBS'18]
- MegaSampler [PRI'23]

## Current state of the art

**Counting**

- SMTApproxMC [CMMV '16]
- SMC [KM '19]
- SearchMC [KM '18]

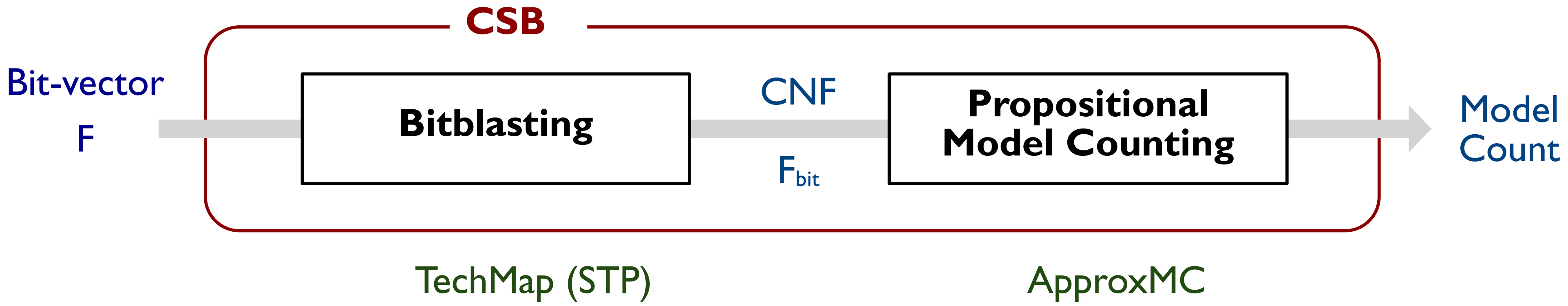
**Sampling**

- SMTSampler [DLBS'18]
- GuidedSampler [DBS'18]
- MegaSampler [PRI'23]

## This work

- Significant recent improvement on CNF counting and sampling
  - Can the improvement be translated to bit-vector counting directly?

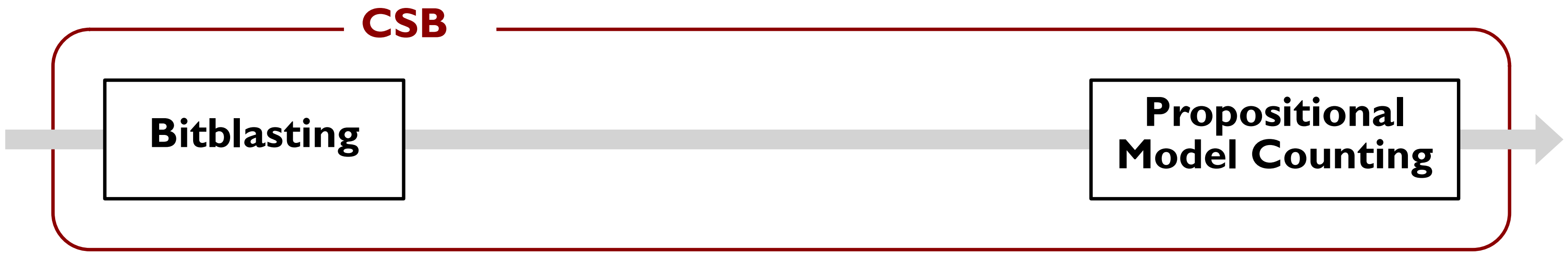
## Framework for Counting



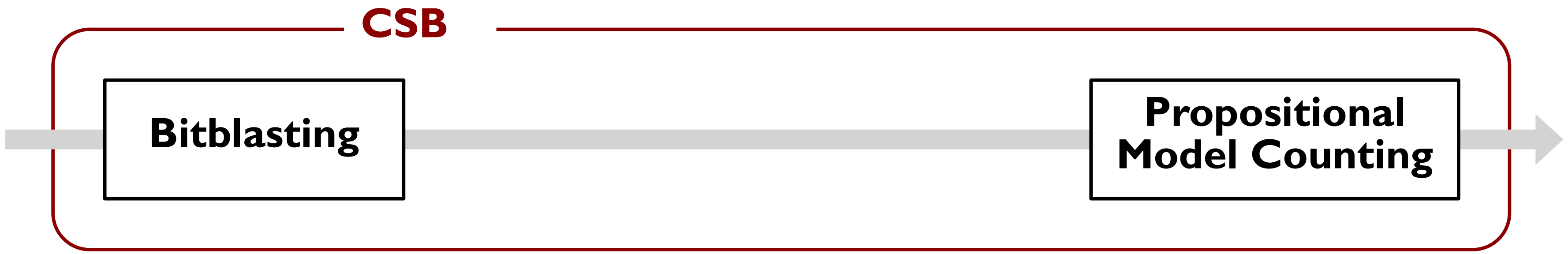
### Implementation

- Built on top of SMT solver **STP**
- Turned off simplifications and rewrites
- Added ApproxMC in place of SAT solver backend

## Trying different possibilites

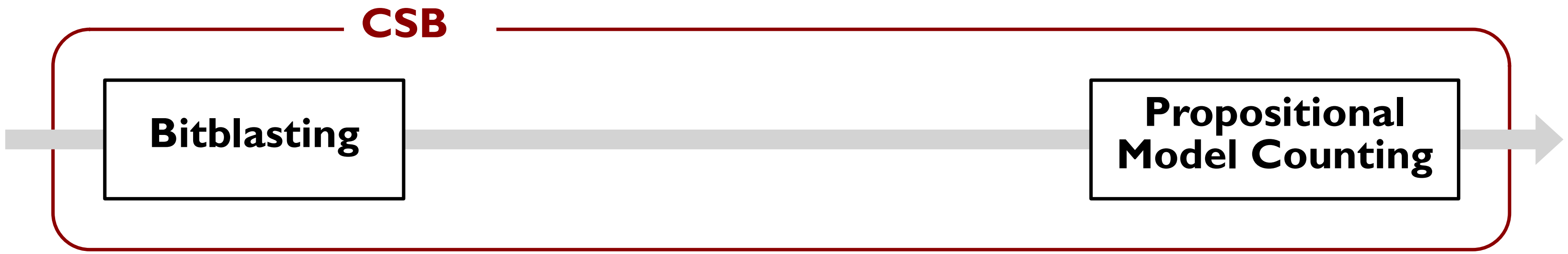


## Trying different possibilites



- Logic synthesis based Technology Mapping
  - [EMS'07] (STP)
- Tseitin encoding
  - [Tse'79] (Boolector)

## Trying different possibilites

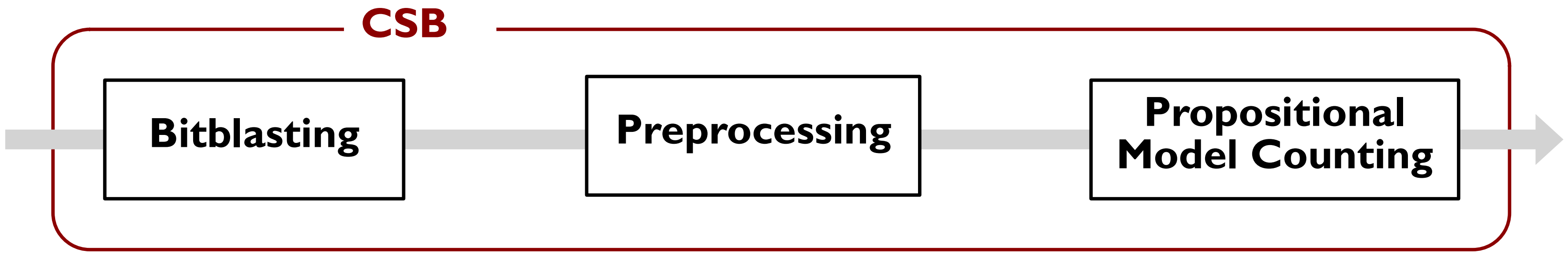


- Logic synthesis based Technology Mapping
  - [EMS'07] (STP)
- Tseitin encoding
  - [Tse'79] (Boolector)

Approximate Counter:
- ApproxMC
  - Hashing-based counter

Exact counter:
- ADDMC
- ExactMC
- SharpSAT-TD
- Ganak

## Trying different possibilites

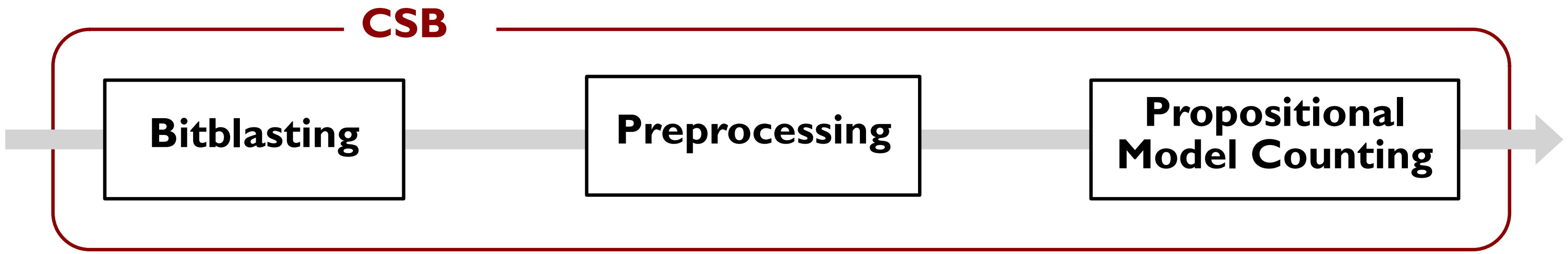


- Logic synthesis based Technology Mapping
  - [EMS'07] (STP)
- Tseitin encoding
  - [Tse'79] (Boolector)

Approximate Counter:
- ApproxMC
  - Hashing-based counter

Exact counter:
- ADDMC
- ExactMC
- SharpSAT-TD
- Ganak

## Trying different possibilites



- Logic synthesis based Technology Mapping

  - [EMS'07] (STP)

- Tseitin encoding

  - [Tse'79] (Boolector)

- Arjun
- B + E

  - Input-output bipartition

Approximate Counter:

- ApproxMC

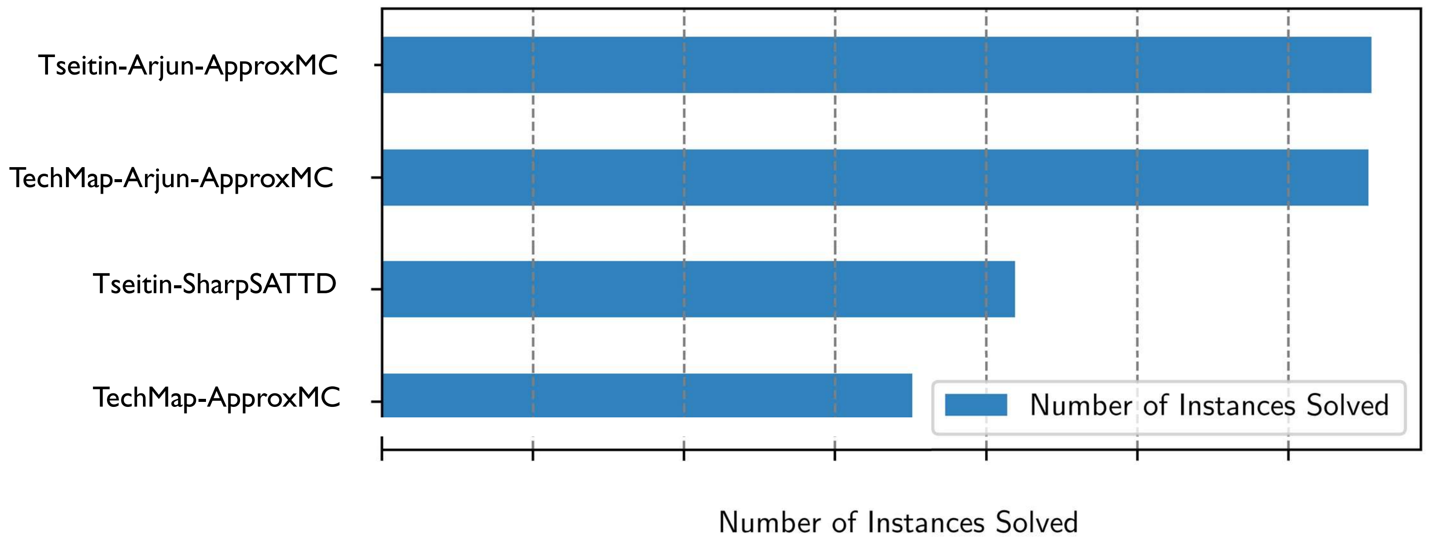
  - Hashing-based counter

Exact counter:

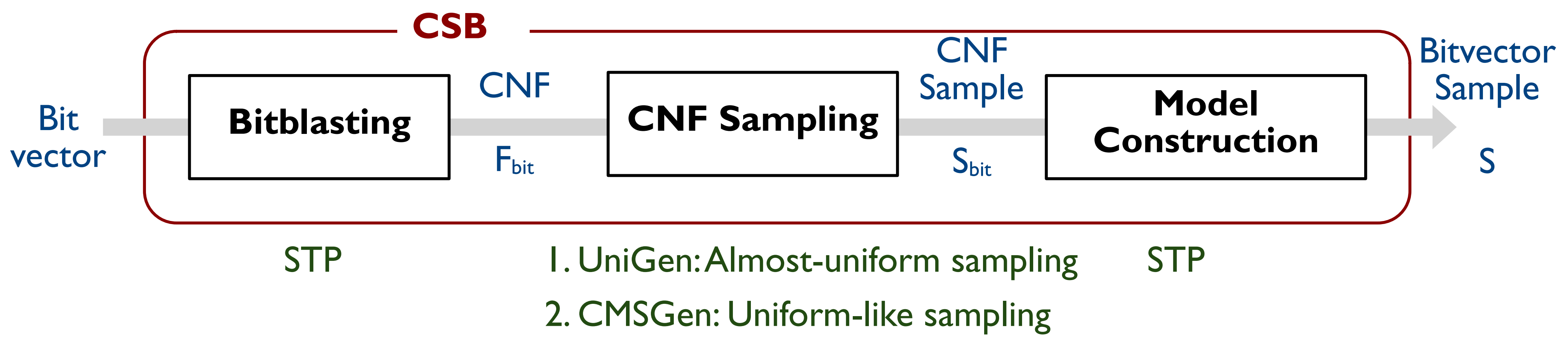
- ADDMC
- ExactMC
- SharpSAT-TD
- Ganak

## Best Performing Settings

### Bitblasting + Preprocessing + Model Counting



## Framework for Sampling



### Implementation

- Built on top of SMT solver **STP**
- Turned off simplifications and rewrites
- Added UniGen and CMSGen in place of SAT solver backend

## Two modes of Sampling

- **Almost-uniform** sampling (UniGen)
  - Sampling with theoretical guarantees. Hashing-based approach.
- **Uniform-like** sampling (CMSGen)
  - No theoretical guarantee, but passes distribution testing based criterions
  - CMSGen is made by using random heursitics in CryptoMiniSat

## Two modes of Sampling

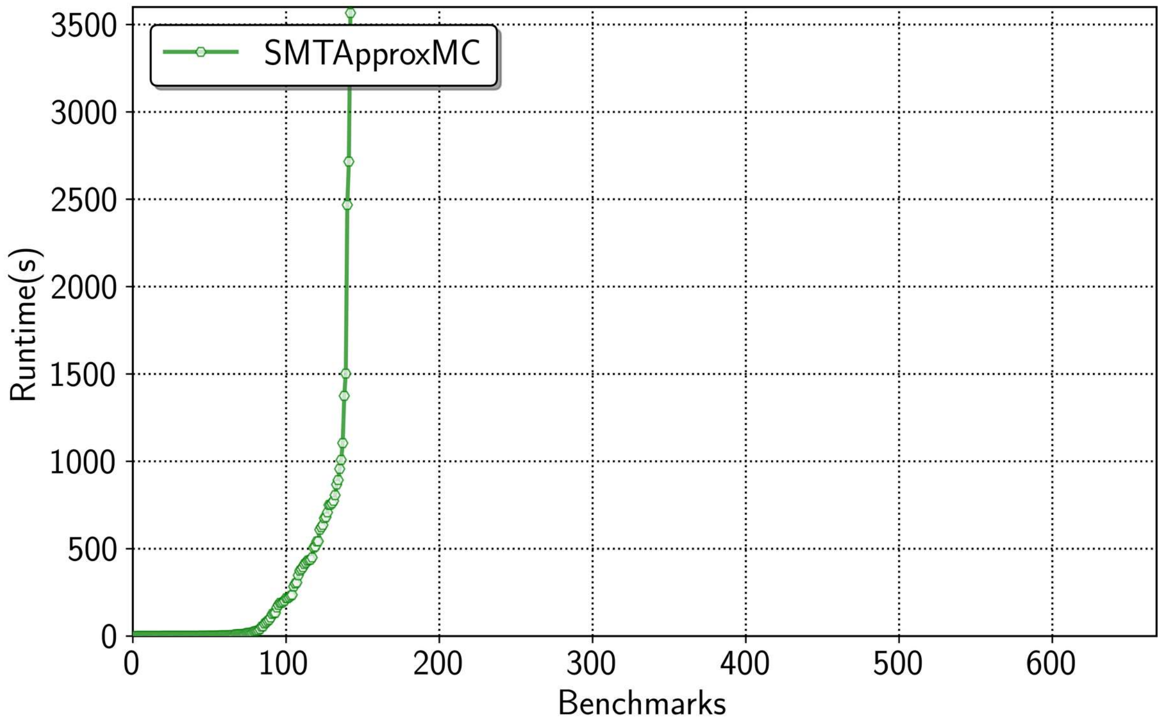
- **Almost-uniform** sampling (UniGen)
  - Sampling with theoretical guarantees. Hashing-based approach.
- **Uniform-like** sampling (CMSGen)
  - No theoretical guarantee, but passes distribution testing based criterions
  - CMSGen is made by using random heursitics in CryptoMiniSat

**Uniform-like sampling is much faster in practice**

## Experiments: Efficiency in Counting

**Compiled a set of 668 benchmarks**
- quantitave model checking
- cryptography
- old literature

| | |
|---|---|
| SMTApproxMC | 143 |

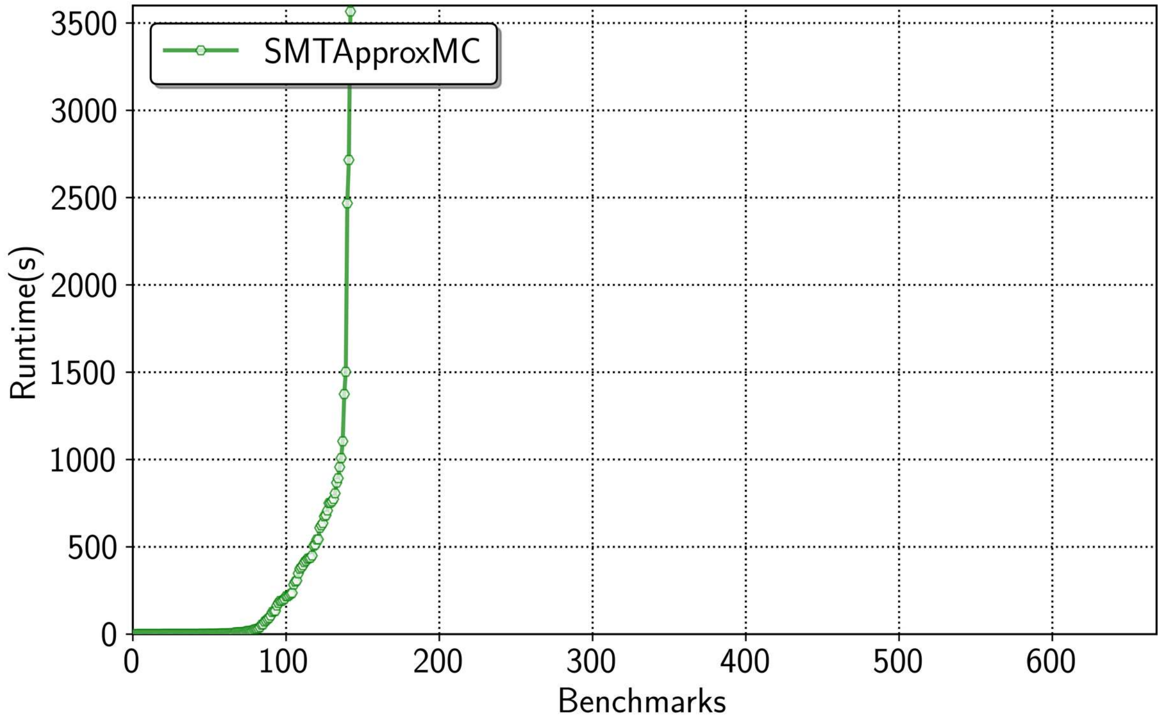


**# Instances counted in 1 hour / 668 instances**

## Experiments: Efficiency in Counting

**Compiled a set of 668 benchmarks**
- quantitave model checking
- cryptography
- old literature

| | |
|---|---|
| SMTApproxMC | 143 |

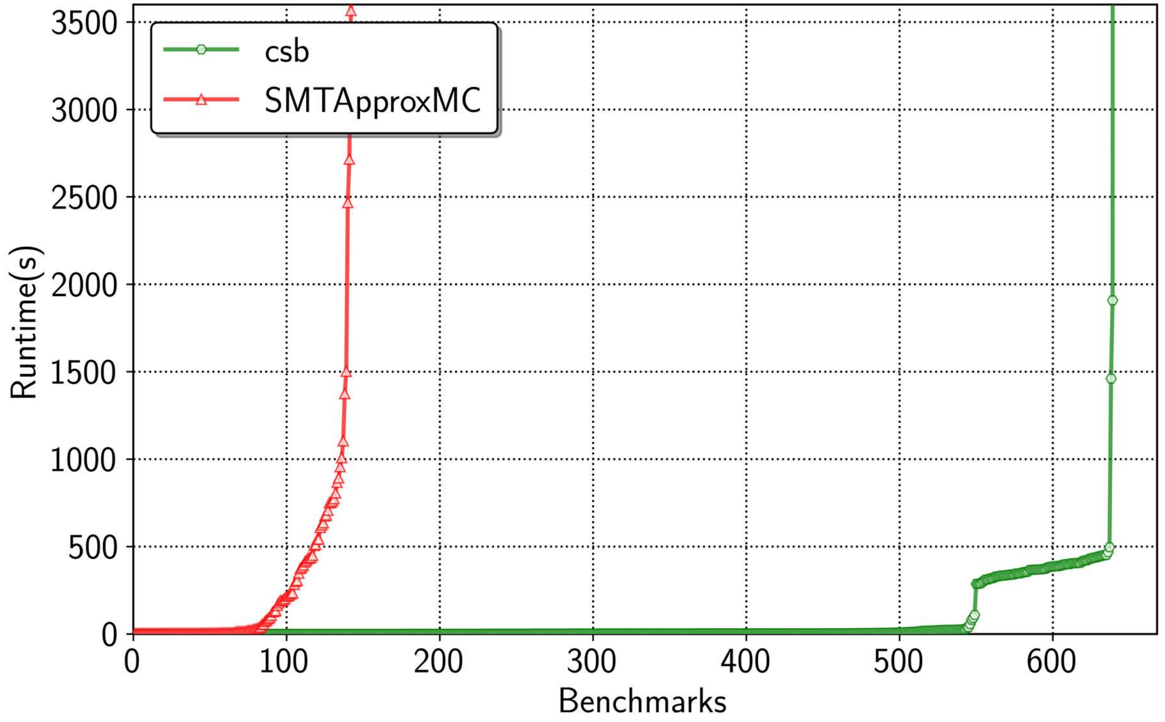


**# Instances counted in 1 hour / 668 instances**

# Experiments: Efficiency in Counting

**Compiled a set of 668 benchmarks**
- quantitave model checking
- cryptography
- old literature

| | |
|---|---|
| csb | 647 |
| SMTApproxMC | 143 |



**# Instances counted in 1 hour / 668 instances**

## Efficiency in Sampling

| Sampling Mode | Median Runtime (s) | Instances solved |
|---|---|---|
| Almost-uniform *(with guarantees)* | 74.6 | 641 |
| Uniform-like *(without theoretical guarantees)* | 1.24 | 662 |

**Sampling 500 samples in 1 hour / 668 instances**

## Conclusion

- A very efficient tool for bitvector counting
- Future work: other theories. (What are the important questions?)
- Searching for **applications**

## Conclusion

- A very efficient tool for bitvector counting
- Future work: other theories. (What are the important questions?)
- Searching for **applications**

github.com/meelgroup/csb/

**Thank You!**