NP? No Problem! An Invitation to the World of Formal Methods

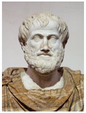
Kuldeep S. Meel

School of Computing

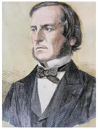
National University of Singapore

Ph.D. (2017), advised by Prof. Supratik Chakraborty (IIT Bombay) and Prof. Moshe Vardi (Rice University)

The Quest for Automated Formal Reasoning



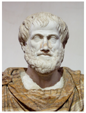
$$\begin{array}{l} \text{All Greeks are humans} \\ \text{All humans are mortal} \\ \hline \text{All Greeks are mortal} \end{array}$$



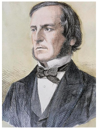
Assign Symbols &
Use Algebra

$$\begin{array}{c} g \rightarrow h \\ h \rightarrow m \\ \hline g \rightarrow m \end{array}$$

The Quest for Automated Formal Reasoning



$$\begin{array}{l} \text{All Greeks are humans} \\ \text{All humans are mortal} \\ \hline \text{All Greeks are mortal} \end{array}$$

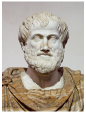


Assign Symbols &
Use Algebra

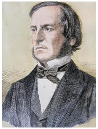
$$\begin{array}{l} \mathrm{g} \rightarrow \mathrm{h} \\ \mathrm{h} \rightarrow \mathrm{m} \\ \hline \mathrm{g} \rightarrow \mathrm{m} \end{array}$$

Central Equation: Is it always the case that $((g \rightarrow h) \wedge (h \rightarrow m)) \rightarrow (g \rightarrow m)$?
Or Equivalently, can it be the case $((g \rightarrow h) \wedge (h \rightarrow m)) \rightarrow !(g \rightarrow m)$?

The Quest for Automated Formal Reasoning



$$\begin{array}{l} \text{All Greeks are humans} \\ \text{All humans are mortal} \\ \hline \text{All Greeks are mortal} \end{array}$$



Assign Symbols &
Use Algebra

$$\begin{array}{l} \mathrm{g} \rightarrow \mathrm{h} \\ \mathrm{h} \rightarrow \mathrm{m} \\ \hline \mathrm{g} \rightarrow \mathrm{m} \end{array}$$

Central Equation: Is it always the case that $((g \rightarrow h) \wedge (h \rightarrow m)) \rightarrow (g \rightarrow m)$?
Or Equivalently, can it be the case $((g \rightarrow h) \wedge (h \rightarrow m)) \rightarrow !(g \rightarrow m)$?

William Stanley Jevons, 1835-1882: "I have given much attention, therefore, to lessening both the manual and mental labour of the process, and I shall describe several devices which may be adopted for saving trouble and risk of mistake."

Boolean Satisfiability

Boolean Satisfiability (SAT); Given a Boolean expression, using "and" ($\wedge$) "or", ($\vee$) and "not" ($\neg$), *is there a satisfying solution* (an assignment of 0's and 1's to the variables that makes the expression equal 1)?
Example:

$$(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (x_3 \vee x_1 \vee x_4)$$

Solution: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$

Boolean Satisfiability

Boolean Satisfiability (SAT); Given a Boolean expression, using "and" ($\wedge$) "or", ($\vee$) and "not" ($\neg$), *is there a satisfying solution* (an assignment of 0's and 1's to the variables that makes the expression equal 1)?
Example:

$$(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (x_3 \vee x_1 \vee x_4)$$

Solution: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$

- Ernst Schröder, 1841-1902: "Getting a handle on the consequences of any premises, or at least the fastest method for obtaining these consequences, seems to me to be one of the noblest, if not the ultimate goal of mathematics and logic."

Algorithmic Boolean Reasoning: Early History

- Davis and Putnam, 1958: "Computational Methods in The Propositional calculus", unpublished report to the NSA
- Davis and Putnam, JACM 1960: "A Computing procedure for quantification theory"
- Davis, Logemman, and Loveland, CACM 1962: "A machine program for theorem proving"

The DPLL algorithm

The DPLL algorithm

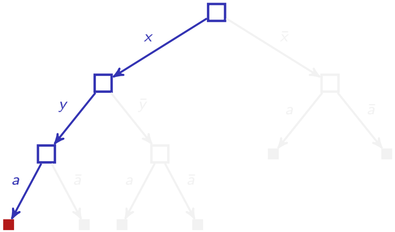
$$\mathcal{F} = (x \vee y) \wedge (a \vee b) \wedge (\bar{a} \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

Level	Dec.	Unit Prop.
0	$\emptyset$	
1	x	
2	y	
3	a	

The DPLL algorithm

$$\mathcal{F} = (x \vee y) \wedge (a \vee b) \wedge (\bar{a} \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

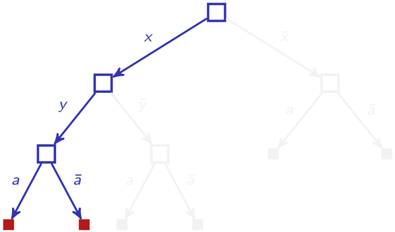
Level	Dec.	Unit Prop.
0	$\emptyset$	
1	x	
2	y	
3	a	$\rightarrow b \rightarrow \perp$



The DPLL algorithm

$$\mathcal{F} = (x \vee y) \wedge (a \vee b) \wedge (\bar{a} \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

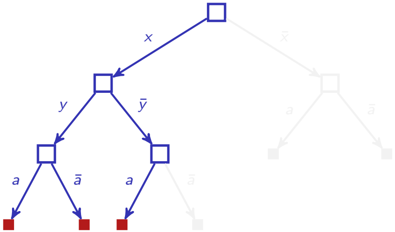
Level	Dec.	Unit Prop.
0	$\emptyset$	
1	x	
2	y	
3	$\bar{a}$	$\bar{b}$ $\perp$



The DPLL algorithm

$$\mathcal{F} = (x \vee y) \wedge (a \vee b) \wedge (\bar{a} \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

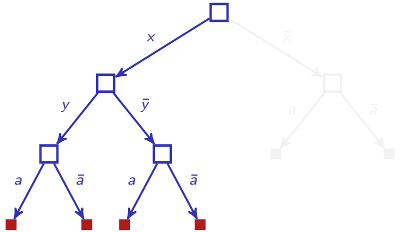
Level	Dec.	Unit Prop.
0	$\emptyset$	
1	x	
2	$\bar{y}$	
3	a	$\longrightarrow b \longrightarrow \perp$



The DPLL algorithm

$$\mathcal{F} = (x \vee y) \wedge (a \vee b) \wedge (\bar{a} \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

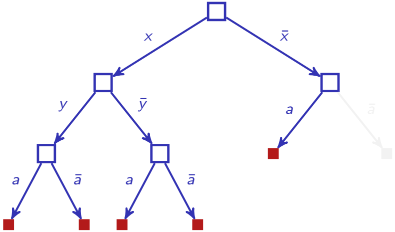
Level	Dec.	Unit Prop.
0	$\emptyset$	
1	x	
2	$\bar{y}$	
3	$\bar{a}$	$\bar{b} \rightarrow \bot$



The DPLL algorithm

$$\mathcal{F} = (x \vee y) \wedge (a \vee b) \wedge (\bar{a} \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

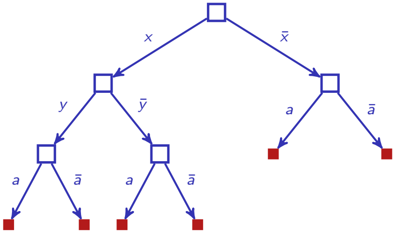
Level	Dec.	Unit Prop.
0	$\emptyset$	
1	$\bar{x}$	$\longrightarrow y$
2	a	$\longrightarrow b \longrightarrow \perp$



The DPLL algorithm

$$\mathcal{F} = (x \vee y) \wedge (a \vee b) \wedge (\bar{a} \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

Level	Dec.	Unit Prop.
0	$\emptyset$	
1	$\bar{x}$	$\longrightarrow y$
2	$\bar{a}$	$\longrightarrow \bar{b} \longrightarrow \perp$



The Dreaded 70s

Hoare, 1969: Proving correctness of the programs can be reduced to theorem proving

- For a large interesting class of programs, proving correctness reduces to SAT

The Dreaded 70s

Hoare, 1969: Proving correctness of the programs can be reduced to theorem proving

- For a large interesting class of programs, proving correctness reduces to SAT

Cook, 1971, Levin, 1972: SAT is NP-complete

The Dreaded 70s

Hoare, 1969: Proving correctness of the programs can be reduced to theorem proving

- For a large interesting class of programs, proving correctness reduces to SAT

Cook, 1971, Levin, 1972: SAT is NP-complete

Vardi: "When I was a graduate student in 1970's, SAT was a scary problem, not to be touched by a 10 foot pole"

The Dreaded 70s

Hoare, 1969: Proving correctness of the programs can be reduced to theorem proving

- For a large interesting class of programs, proving correctness reduces to SAT

Cook, 1971, Levin, 1972: SAT is NP-complete

Vardi: "When I was a graduate student in 1970's, SAT was a scary problem, not to be touched by a 10 foot pole"

"I can't find an efficient algorithm, but neither can all these famous people."

(Cartoon adapted from Gary Johnson)

De Millo, Lipton, Perlis, 1979: "formal verifications of programs, no matter how obtained, will not play the same key role in the development of computer science and software engineering as proofs do in mathematics."

The Cautious 80's followed by the storm

Clark, Emerson, 1981: "We argue that proof construction is unnecessary in the case of finite state concurrent systems and can be replaced by a model-theoretic approach which will mechanically determine if the system meets a specification expressed in propositional temporal logic"

Burch, EM Clarke, KL McMillan, DL Dill, LJ Hwang , 1992: Symbolic model checking, restricted to Binary Decision Diagrams: formulas for which Satisfiability is PTIME

The Cautious 80's followed by the storm

Clark, Emerson, 1981: "We argue that proof construction is unnecessary in the case of finite state concurrent systems and can be replaced by a model-theoretic approach which will mechanically determine if the system meets a specification expressed in propositional temporal logic"

Burch, EM Clarke, KL McMillan, DL Dill, LJ Hwang , 1992: Symbolic model checking, restricted to Binary Decision Diagrams: formulas for which Satisfiability is PTIME

Pentium FDIV Bug, 1994: : The recall due to floating point unit bug cost $500 million

Ariane 5, 1996: The rocket exploded only after 40 seconds due to exception handling

The Cautious 80's followed by the storm

Clark, Emerson, 1981: "We argue that proof construction is unnecessary in the case of finite state concurrent systems and can be replaced by a model-theoretic approach which will mechanically determine if the system meets a specification expressed in propositional temporal logic"

Burch, EM Clarke, KL McMillan, DL Dill, LJ Hwang , 1992: Symbolic model checking, restricted to Binary Decision Diagrams: formulas for which Satisfiability is PTIME

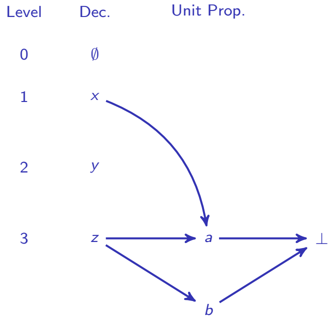
Pentium FDIV Bug, 1994: : The recall due to floating point unit bug cost $500 million

Ariane 5, 1996: The rocket exploded only after 40 seconds due to exception handling

Marques-Silva and Sakallah, 1996: "GRASP is premised on the inevitability of conflicts during the search and its most distinguishing feature is the augmentation of basic backtracking search with a powerful conflict analysis procedure"

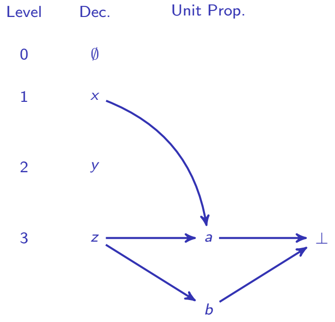
Clause learning

$(\bar{a} \vee \bar{b}) \wedge (\bar{z} \vee b) \wedge (\bar{x} \vee \bar{z} \vee a) \wedge (y \vee b)$



Clause learning

$$(\bar{a} \vee \bar{b}) \wedge (\bar{z} \vee b) \wedge (\bar{x} \vee \bar{z} \vee a) \wedge (y \vee b)$$

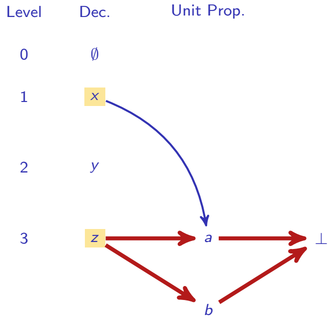


- Analyze conflict

[MSS96a,MSS96b,MSS96c,MSS96d,MSS99]

Clause learning

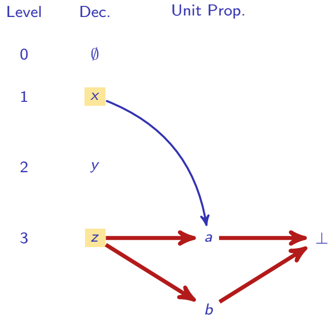
$(\bar{a} \vee \bar{b}) \wedge (\bar{z} \vee b) \wedge (\bar{x} \vee \bar{z} \vee a) \wedge (y \vee b)$



- Analyze conflict [MSS96a,MSS96b,MSS96c,MSS96d,MSS99]
    - Reasons: x and z
        - Decision variable & literals assigned at decision levels less than current

Clause learning

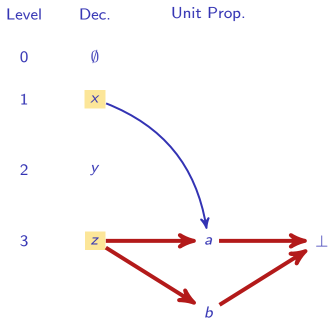
$(\bar{a} \vee \bar{b}) \wedge (\bar{z} \vee b) \wedge (\bar{x} \vee \bar{z} \vee a) \wedge (y \vee b)$



- Analyze conflict [MSS96a,MSS96b,MSS96c,MSS96d,MSS99]
  - Reasons: x and z
    - Decision variable & literals assigned at decision levels less than current
  - Create **new** clause: $(\bar{x} \vee \bar{z})$

Clause learning

$(\bar{a} \vee \bar{b}) \wedge (\bar{z} \vee b) \wedge (\bar{x} \vee \bar{z} \vee a) \wedge (y \vee b)$

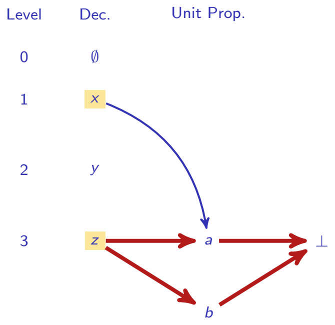


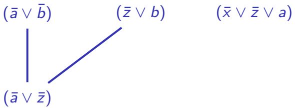
$(\bar{a} \vee \bar{b})$ $(\bar{z} \vee b)$ $(\bar{x} \vee \bar{z} \vee a)$

- Analyze conflict [MSS96a,MSS96b,MSS96c,MSS96d,MSS99]
  - Reasons: x and z
    - Decision variable & literals assigned at decision levels less than current
  - Create **new** clause: $(\bar{x} \vee \bar{z})$
- Can relate clause learning with resolution

Clause learning

$(\bar{a} \vee \bar{b}) \wedge (\bar{z} \vee b) \wedge (\bar{x} \vee \bar{z} \vee a) \wedge (y \vee b)$

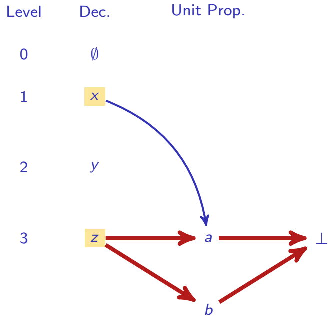


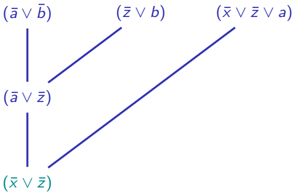


- Analyze conflict [MSS96a,MSS96b,MSS96c,MSS96d,MSS99]
  - Reasons: x and z
    - Decision variable & literals assigned at decision levels less than current
  - Create **new** clause: $(\bar{x} \vee \bar{z})$
- Can relate clause learning with resolution

Clause learning

$(\bar{a} \lor \bar{b}) \land (\bar{z} \lor b) \land (\bar{x} \lor \bar{z} \lor a) \land (y \lor b)$

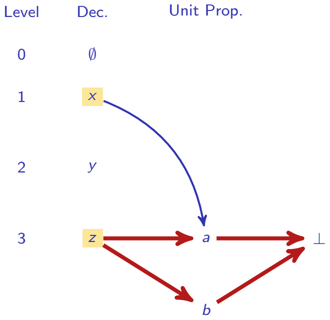


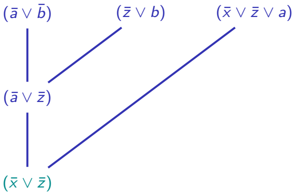


- Analyze conflict [MSS96a,MSS96b,MSS96c,MSS96d,MSS99]
  - Reasons: x and z
    - Decision variable & literals assigned at decision levels less than current
  - Create **new** clause: $(\bar{x} \lor \bar{z})$
- Can relate clause learning with resolution

Clause learning

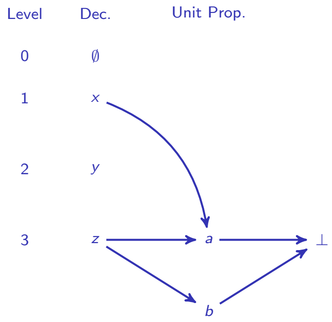
$(\bar{a} \vee \bar{b}) \wedge (\bar{z} \vee b) \wedge (\bar{x} \vee \bar{z} \vee a) \wedge (y \vee b)$



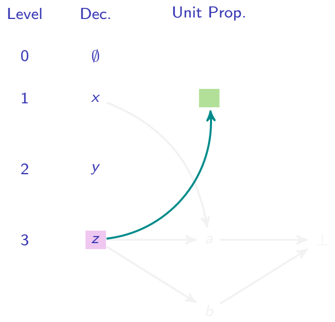


- Analyze conflict [MSS96a,MSS96b,MSS96c,MSS96d,MSS99]
  - Reasons: x and z
    - Decision variable & literals assigned at decision levels less than current
  - Create **new** clause: $(\bar{x} \vee \bar{z})$
- Can relate clause learning with resolution
  - Learned clauses result from (**selected**) resolution operations

Clause learning – after backtracking

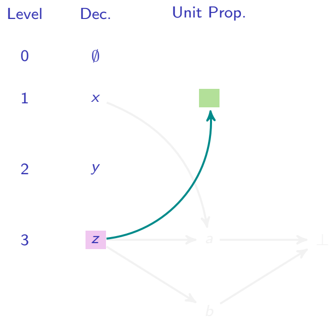


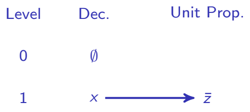
Clause learning – after backtracking



- Clause $(\bar{x} \vee \bar{z})$ is asserting at decision level 1

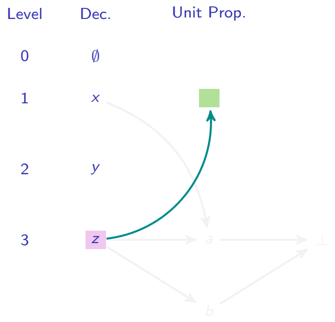
Clause learning – after backtracking

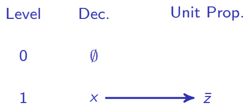




- Clause $(\bar{x} \vee \bar{z})$ is asserting at decision level 1

Clause learning – after backtracking





- Clause $(\bar{x} \vee \bar{z})$ is asserting at decision level 1
- Backtracking differs from plain DPLL:
  - Always bactrack after a conflict

The Roaring 2000s

- Clause learning & non-chronological backtracking [MSS96a,MSS99,BS97,Z97]
  - Exploit UIPs [MSS96a,SSS12]
  - Minimize learned clauses [SB09,VG09]
  - Opportunistically delete clauses [MSS96a,MSS99,GN02]
- Search restarts [GSK98,BMS00,H07,B08]

The Roaring 2000s

- Clause learning & non-chronological backtracking [MSS96a,MSS99,BS97,Z97]
  - Exploit UIPs [MSS96a,SSS12]
  - Minimize learned clauses [SB09,VG09]
  - Opportunistically delete clauses [MSS96a,MSS99,GN02]
- Search restarts [GSK98,BMS00,H07,B08]
- Lazy data structures
  - Watched literals [MMZZM01]
- Conflict-guided branching
  - Lightweight branching heuristics [MMZZM01]
  - Phase saving [S00,PD07]

The Roaring 2000s

- Clause learning & non-chronological backtracking [MSS96a,MSS99,BS97,Z97]
  - Exploit UIPs [MSS96a,SSS12]
  - Minimize learned clauses [SB09,VG09]
  - Opportunistically delete clauses [MSS96a,MSS99,GN02]
- Search restarts [GSK98,BMS00,H07,B08]
- Lazy data structures
  - Watched literals [MMZZM01]
- Conflict-guided branching
  - Lightweight branching heuristics [MMZZM01]
  - Phase saving [S00,PD07]

Biere, Cimatti, Clarke, Zhu, 1999: "We show how boolean decision procedures can replace BDDs. This new technique avoids the space blow up of BDDs, generates counterexamples much faster."

Clark, Emerson, Sifakis, 2007: Turing Award

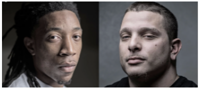
Knuth, 2010s: SAT is far from an abstract exercise in understanding formal systems. These so-called "SAT solvers" can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.

Beyond NP

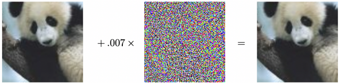
[Circa 2012 @IIT Bombay]: Now that NP is "No Problem", it is time to look beyond satisfiability

```
PC2 (char[] SP, char[] UI) {
        match = true;
        for (int i=0; i<UI.length(); i++) {
                if (SP[i] != UI[i]) match=false;
                else match = match;
        }
        if match return Yes;
        else return No;
}
```

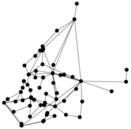
Information Leakage



Fairness



Robustness



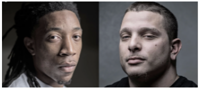
Critical Infrastructure

Beyond NP

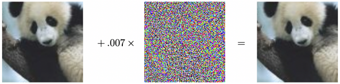
[Circa 2012 @IIT Bombay]: Now that NP is "No Problem", it is time to look beyond satisfiability

```
PC2 (char[] SP, char[] UI) {
        match = true;
        for (int i=0; i<UI.length(); i++) {
                if (SP[i] != UI[i]) match=false;
                else match = match;
        }
        if match return Yes;
        else return No;
}
```

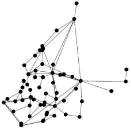
Information Leakage



Fairness



Robustness



Critical Infrastructure

Quantification: How often does $\mathcal{M}$ satisfy $\mathcal{P}$?

Counting

Resilience of Critical Infrastructure Networks

[DMPV17,PDMV19]

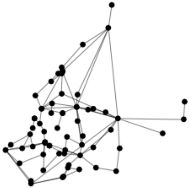


Can we predict the likelihood of a blackout due to natural disaster?

Resilience of Critical Infrastructure Networks

[DMPV17,PDMV19]

Can we predict the likelihood of a blackout due to natural disaster?



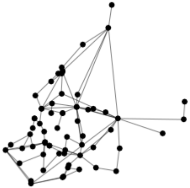
- $G = (V, E)$; set of source nodes S and terminal node t
- failure probability $g : E \to [0, 1]$
- Compute Pr[t is disconnected from S]?

Resilience of Critical Infrastructure Networks

[DMPV17,PDMV19]



Can we predict the likelihood of a blackout due to natural disaster?



- $G = (V, E)$; set of source nodes S and terminal node t
- failure probability $g : E \to [0, 1]$
- Compute Pr[t is disconnected from S]?

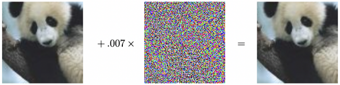
  Counting

Key Idea: Encode disconnectedness using constraints

Impact: The first theoretically sound estimates of resilience in power transmission networks of ten medium sized cities in US

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22]

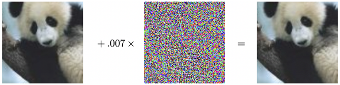


Robustness Quantification

$$\Big|\{x : \mathcal{N}(x+\varepsilon) \neq \mathcal{N}(x)\}\Big|$$

Quantitative Analysis of AI Systems

[BSS**M**S19,NS**M**IS19,NS**M**ISV22]



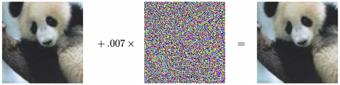
Robustness Quantification

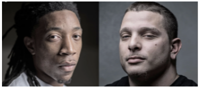
$$\underbrace{\left|\{x : \mathcal{N}(x+\varepsilon) \neq \mathcal{N}(x)\}\right|}_{\text{Encode Symbolically}}$$

$\boxed{\text{Counting}}$

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22]





Robustness Quantification

Fairness Quantification

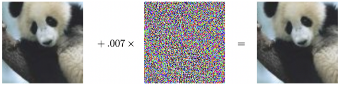
$$\underbrace{\left|\{x : \mathcal{N}(x+\varepsilon) \neq \mathcal{N}(x)\}\right|}_{\text{Encode Symbolically}}$$

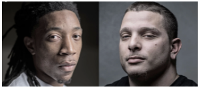
$$\underbrace{\left|\{x : \ \mathcal{N}(x \wedge \textsc{Black}) \neq \mathcal{N}(x \wedge \textsc{White})\}\right|}_{\text{Encode Symbolically}}$$

$\boxed{\text{Counting}}$

Quantitative Analysis of AI Systems

[BSS**M**S19,NS**M**IS19,NS**M**ISV22]





Robustness Quantification

Fairness Quantification

$$\underbrace{\left|\{x : \mathcal{N}(x+\varepsilon) \neq \mathcal{N}(x)\}\right|}_{\text{Encode Symbolically}}$$

$$\underbrace{\left|\{x :\ \mathcal{N}(x \wedge \textsc{Black}) \neq \mathcal{N}(x \wedge \textsc{White})\}\right|}_{\text{Encode Symbolically}}$$

$\boxed{\text{Counting}}$

Impact: The first scalable technique for rigorous quantification of robustness and fairness of Binarized Neural Networks

Counting

- Given: A Boolean formula F over $X_1, X_2, \cdots X_n$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$
- SAT: Determine if $\mathsf{Sol}(F)$ is non-empty
- Counting: Determine $|\mathsf{Sol}(F)|$

Counting

- Given: A Boolean formula F over $X_1, X_2, \cdots X_n$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$
- SAT: Determine if $\mathsf{Sol}(F)$ is non-empty
- Counting: Determine $|\mathsf{Sol}(F)|$
- Example: $F := (X_1 \vee X_2)$
  - $\mathsf{Sol}(F) = \{(0,1),(1,0),(1,1)\}$
  - $|\mathsf{Sol}(F)| = 3$

Valiant, 1979: Counting exactly is **#P-hard**

Counting

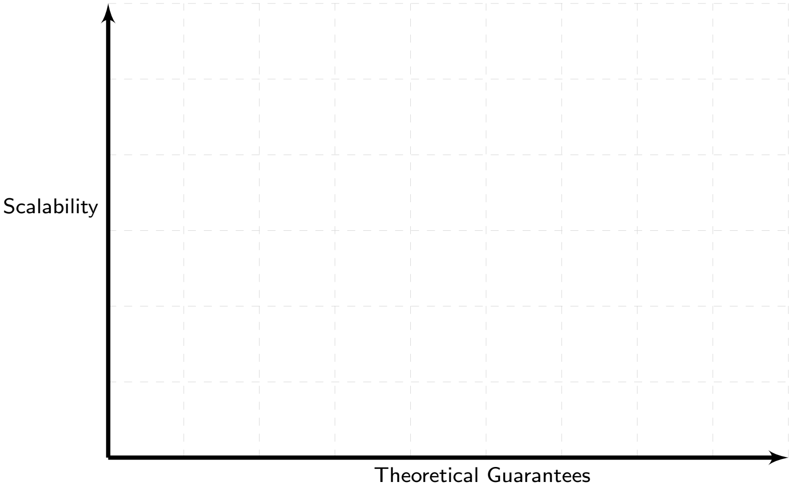
- Given: A Boolean formula F over $X_1, X_2, \cdots X_n$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$
- SAT: Determine if $\mathsf{Sol}(F)$ is non-empty
- Counting: Determine $|\mathsf{Sol}(F)|$
- Example: $F := (X_1 \vee X_2)$
  - $\mathsf{Sol}(F) = \{(0,1),(1,0),(1,1)\}$
  - $|\mathsf{Sol}(F)| = 3$

Valiant, 1979: Counting exactly is **#P-hard**

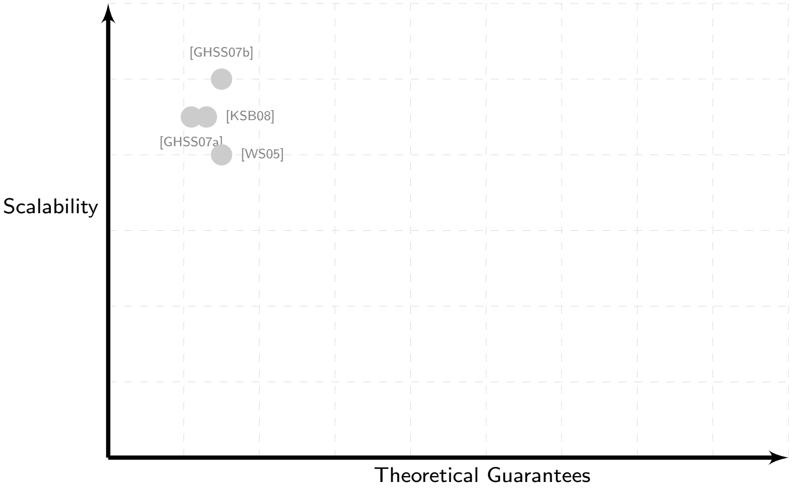
Stockmeyer, 1983: Probably Approximately Correct (PAC) aka (ε, δ)-guarantees

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

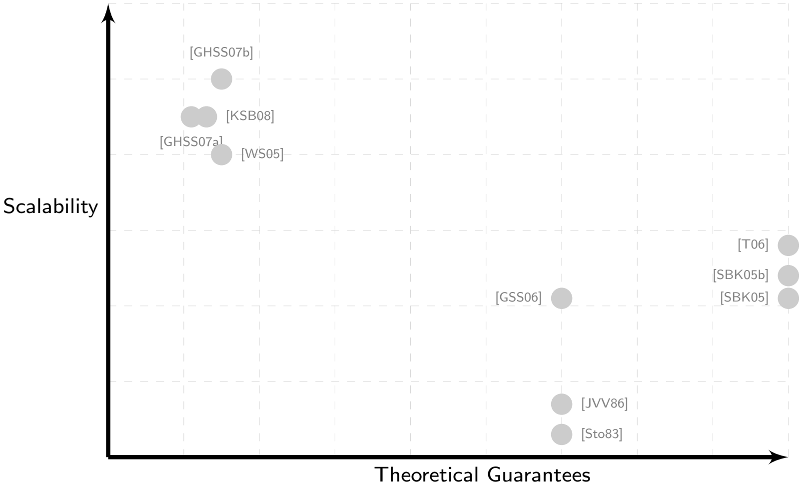
Snapshot from 2012



Snapshot from 2012

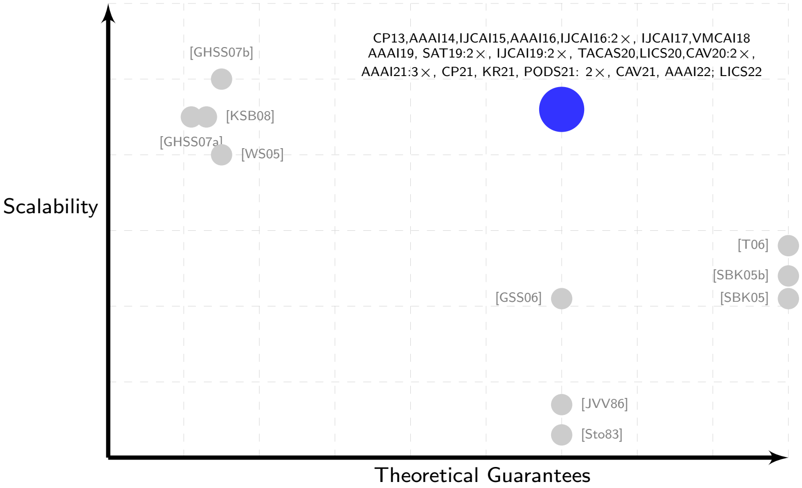


Snapshot from 2012



State of the art tool in 2012 could handle one out of 1076 robustness instances

Can we bridge the gap between theory and practice?



State of the art tool in 2012 could handle one out of 1076 robustness instances

Can we bridge the gap between theory and practice?

Counting in Mumbai

How many people in Mumbai like coffee?

- Population of Mumbai = 12.5M
- Assign every person a unique ($n =$) 24 bit identifier ($2^n \approx$ 12.5M)

Counting in Mumbai

How many people in Mumbai like coffee?

- Population of Mumbai $= 12.5$M
- Assign every person a unique ($n =$) 24 bit identifier ($2^n \approx 12.5$M)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 12.5M/50

Counting in Mumbai

How many people in Mumbai like coffee?

- Population of Mumbai = 12.5M
- Assign every person a unique ($n =$) 24 bit identifier ($2^n \approx 12.5$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 12.5M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50

Counting in Mumbai

How many people in Mumbai like coffee?

- Population of Mumbai = 12.5M
- Assign every person a unique ($n =$) 24 bit identifier ($2^n \approx$ 12.5M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 12.5M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50
- SAT Query: Find a person who likes coffee

Counting in Mumbai

How many people in Mumbai like coffee?

- Population of Mumbai = 12.5M
- Assign every person a unique ($n =$) 24 bit identifier ($2^n \approx 12.5$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 12.5M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50
- SAT Query: Find a person who likes coffee
- A SAT solver can answer queries like:
  - Q1: Find a person who likes coffee
  - Q2: Find a person who likes coffee and is not person y

Counting in Mumbai

How many people in Mumbai like coffee?

- Population of Mumbai = 12.5M
- Assign every person a unique ($n =$) 24 bit identifier ($2^n \approx 12.5$M)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 12.5M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50

- SAT Query: Find a person who likes coffee
- A SAT solver can answer queries like:
  - Q1: Find a person who likes coffee
  - Q2: Find a person who likes coffee and is not person y

- Attempt #2: Enumerate every person who likes coffee

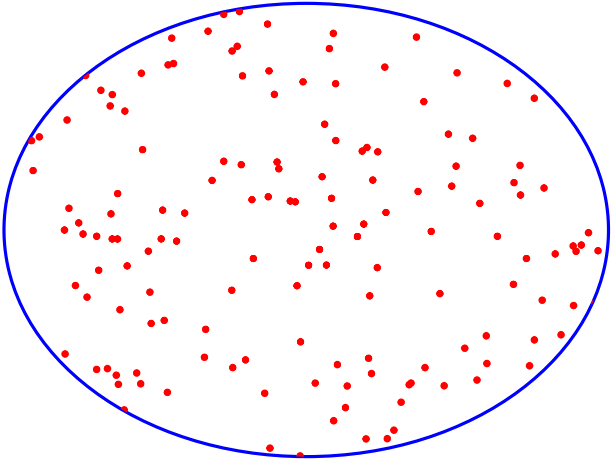
Counting in Mumbai

How many people in Mumbai like coffee?

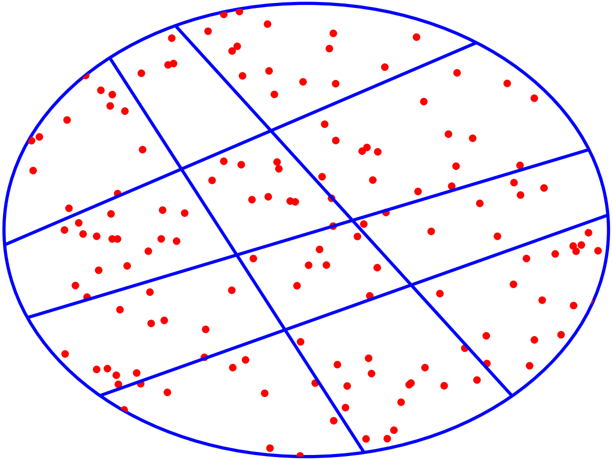
- Population of Mumbai = 12.5M
- Assign every person a unique ($n =$) 24 bit identifier ($2^n \approx 12.5$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 12.5M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50
- SAT Query: Find a person who likes coffee
- A SAT solver can answer queries like:
  - Q1: Find a person who likes coffee
  - Q2: Find a person who likes coffee and is not person y
- Attempt #2: Enumerate every person who likes coffee
  - Potentially 2^n queries

Can we do with lesser # of SAT queries – $\mathcal{O}(n)$ or $\mathcal{O}(\log n)$?

As Simple as Counting Dots

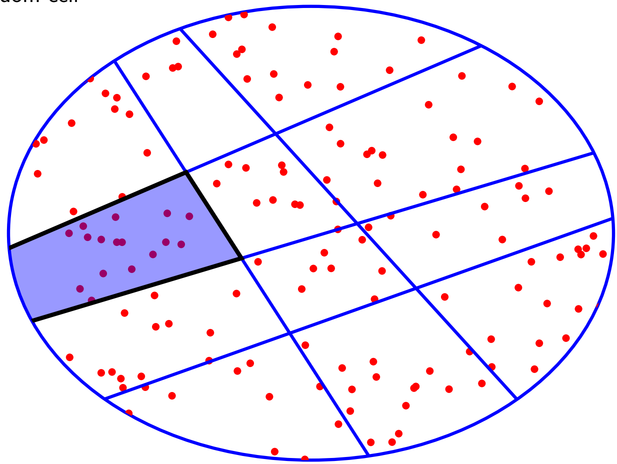


As Simple as Counting Dots



As Simple as Counting Dots

Pick a random cell



Estimate $=$ Number of solutions in a cell $\times$ Number of cells

Challenges

Challenge 1 How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

Challenge 2 How many cells?

Challenges

Challenge 1 How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

- Designing function h : assignments $\rightarrow$ cells (hashing)

Challenges

Challenge 1 How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

- Designing function h : assignments $\rightarrow$ cells (hashing)
- Deterministic h unlikely to work

Challenges

Challenge 1 How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

- Designing function h : assignments $\rightarrow$ cells (hashing)
- Deterministic h unlikely to work
- Choose h randomly from a large family H of hash functions

$\boxed{\text{2-wise Independent Hashing}}$ [CW77]

2-wise Independent Hash Functions

- To construct $h : \{0, 1\}^n \to \{0, 1\}^m$, choose m random XORs
- Pick every X_i with prob. $\frac{1}{2}$ and XOR them
  - $X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2}$

2-wise Independent Hash Functions

- To construct $h : \{0,1\}^n \rightarrow \{0,1\}^m$, choose m random XORs
- Pick every X_i with prob. $\frac{1}{2}$ and XOR them
  - $X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2}$
- To choose $\alpha \in \{0,1\}^m$, set every XOR equation to 0 or 1 randomly

$$X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2} = 0 \quad (Q_1)$$
$$X_2 \oplus X_5 \oplus X_6 \cdots \oplus X_{n-1} = 1 \quad (Q_2)$$
$$\cdots \quad (\cdots)$$
$$X_1 \oplus X_2 \oplus X_5 \cdots \oplus X_{n-2} = 1 \quad (Q_m)$$

- Solutions in a cell: $F \wedge Q_1 \cdots \wedge Q_m$

Challenges

Challenge 1 How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

$\boxed{\text{Random XOR-based Hash Functions}}$ [CW77]

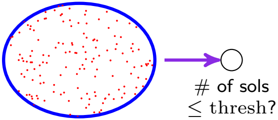
Challenge 2 How many cells?

Challenge 2: How many cells? [CMV13,CMV16]

- A cell is small if it has $\approx \mathrm{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells

Challenge 2: How many cells? [CMV13,CMV16]

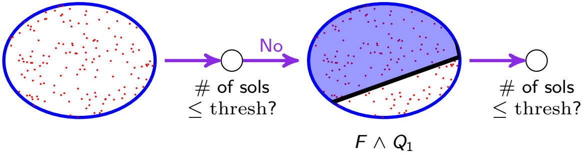
- A cell is small if it has $\approx \mathrm{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

[CMV13,CMV16]

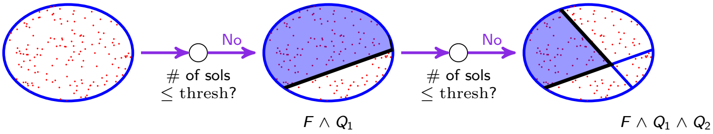
- A cell is small if it has $\approx \mathrm{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

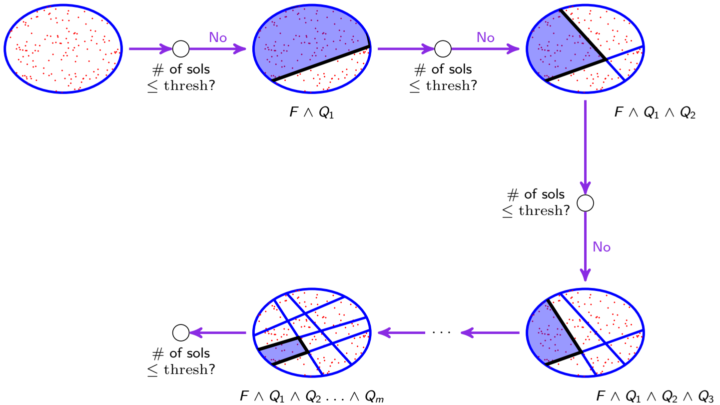
[CMV13,CMV16]

- A cell is small if it has $\approx \text{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells? [CMV13,CMV16]

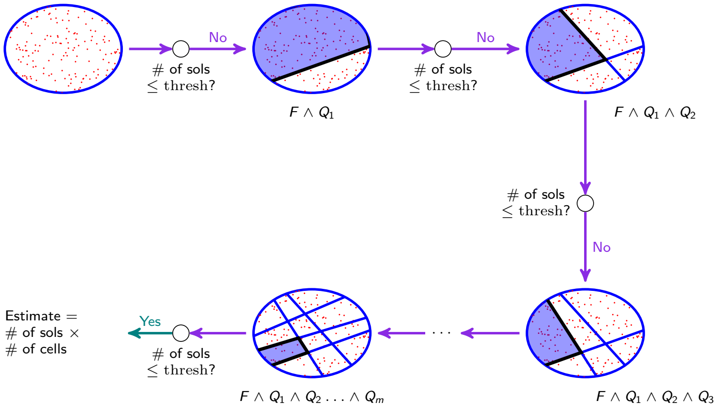
- A cell is small if it has $\approx \mathrm{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

[CMV13,CMV16]

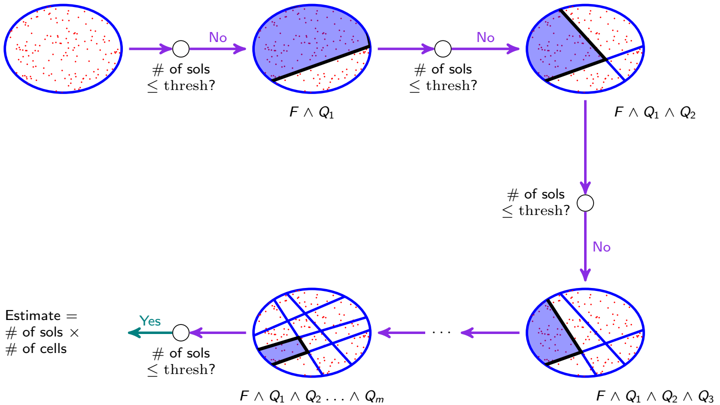
- A cell is small if it has $\approx \text{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

[CMV13,CMV16]

- A cell is small if it has $\approx \text{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



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

ApproxMC makes $O(\frac{1}{\varepsilon^2} \cdot \log\frac{1}{\delta} \cdot \log n)$ SAT queries.

ApproxMC: Early Years (2013-17)

Handle **reasonable** formulas: **reasonable** grids, **reasonable** programs

ApproxMC: Early Years (2013-17)

Handle **reasonable** formulas: **reasonable** grids, **reasonable** programs

B. Cook: Virtuous cycle: application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. Around and around.

In Pursuit of Scalability (2017-now)

Theoretical Advances	Sparse hashing SAT-20 LICS-20	Duality CP-19	DNF CP-18,IJCAI-19 PODS-21,22	Phase Transition IJCAI-16,17,19 CP-20
Algorithmic Engineering	Indep Supp Constraints-16 ICCAD-22	Chain Formulas IJCAI-15 NeurIPS-20	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23
Software Development	CNF-XOR AAAI-19, CAV-20	Pseudo-Boolean CP-21	MaxSAT-XOR KR-21	Hardware Accelerator SAT-21

Reliability of Critical Infrastructure Networks

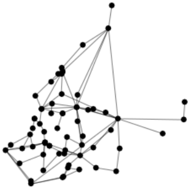
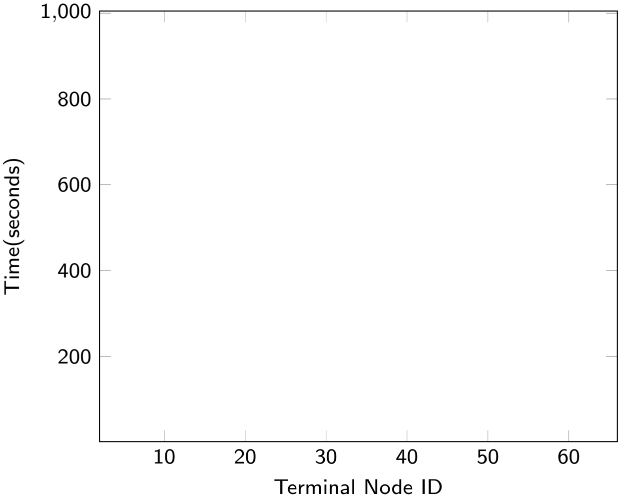


Figure: Plantersville, SC



Timeout = 1000 seconds

Reliability of Critical Infrastructure Networks

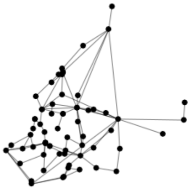
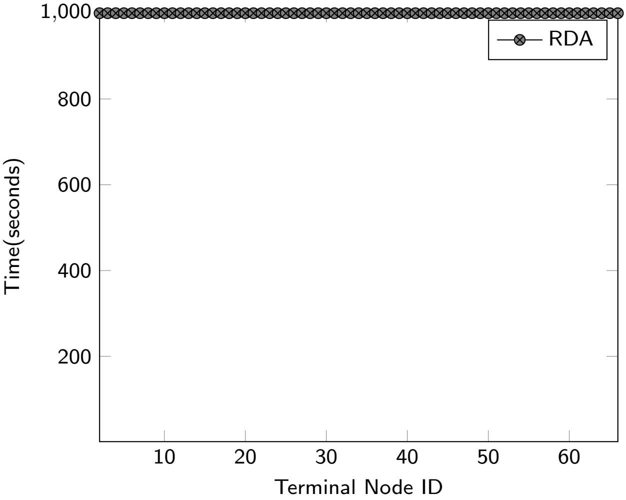


Figure: Plantersville, SC



Timeout = 1000 seconds

Reliability of Critical Infrastructure Networks

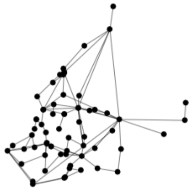
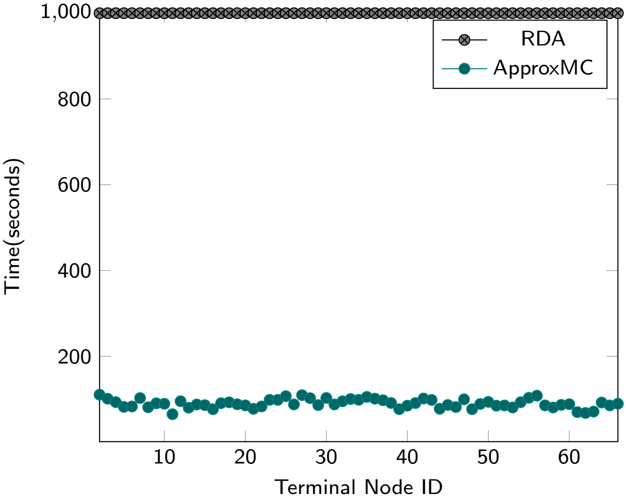


Figure: Plantersville, SC



Timeout = 1000 seconds

Impact: The first theoretically sound estimates of resilience in power transmission networks of ten medium sized cities in US

Applications

approxmc Public

Approximate Model Counter

C++ ☆ 51 18

SharpTNI: Counting and Sampling Parsimonious Transmission Networks under a Weak Bottleneck

Palash Sashittal[1] and Mohammed El-Kebir[2*]

Static Evaluation of Noninterference using Approximate Model Counting

Ziqiao Zhou Zhiyun Qian Michael K. Reiter Yinqian Zhang

Check before You Change: Preventing Correlated Failures in Service Updates

Ennan Zhai[†], Ang Chen[‡], Ruzica Piskac[○], Mahesh Balakrishnan[§,*]
Bingchuan Tian[♮], Bo Song[•], Haoliang Zhang[•]

A Study of the Learnability of Relational Properties

Model Counting Meets Machine Learning (MCML)

Muhammad Usman, University of Texas at Austin, USA, muhammadusman@utexas.edu
Wenxi Wang, University of Texas at Austin, USA, wenxiw@utexas.edu
Marko Vasic, University of Texas at Austin, USA, vasic@utexas.edu
Kaiyuan Wang[*] Haris Vikalo Sarfraz Khurshid

Automating the Development of Chosen Ciphertext Attacks

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

Quantifying Software Reliability via Model-Counting

Samuel Teuber[(✉)] and Alexander Weigl

Quantifying the Efficacy of Logic Locking Methods

Joseph Sweeney, Deepali Garg, Lawrence Pileggi

IN SEARCH FOR A SAT-FRIENDLY BINARIZED NEURAL NETWORK ARCHITECTURE

Nina Narodytska **Hongce Zhang[*]**

Where do we go from here?

Where do we go from here?

B. Cook, 2022: Virtuous cycle: ...application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. **Around and around**.

Where do we go from here?

B. Cook, 2022: Virtuous cycle: ...application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. **Around and around**.

Today's Counters $\approx$ SAT Solvers in early 2000s

Where do we go from here?

B. Cook, 2022: Virtuous cycle: ...application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. **Around and around**.

Today's Counters $\approx$ SAT Solvers in early 2000s

The storm is coming: Statistical systems are being integrated into our lives, the Pentium FDIV and Ariane 5 Rocket moments are inevitable if we do not act

Mission 2028: $100\times$ Speedup for Counting to enable Quantitative Reasoning at Scale

Where do we go from here?

B. Cook, 2022: Virtuous cycle: ...application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. **Around and around**.

Today's Counters $\approx$ SAT Solvers in early 2000s

The storm is coming: Statistical systems are being integrated into our lives, the Pentium FDIV and Ariane 5 Rocket moments are inevitable if we do not act

Mission 2028: 100$\times$ Speedup for Counting to enable Quantitative Reasoning at Scale

Challenge Problems

Civil Engineering Rigorous resilience estimation for power grid of Los Angeles

Neural Network Verification Neural networks with 1M neurons

Software Engineering Information Flow analysis of programs with 10K lines of code

Where do we go from here?

B. Cook, 2022: Virtuous cycle: ...application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. **Around and around**.

Today's Counters $\approx$ SAT Solvers in early 2000s

The storm is coming: Statistical systems are being integrated into our lives, the Pentium FDIV and Ariane 5 Rocket moments are inevitable if we do not act

Mission 2028: $100\times$ Speedup for Counting to enable Quantitative Reasoning at Scale

Challenge Problems

Civil Engineering Rigorous resilience estimation for power grid of Los Angeles

Neural Network Verification Neural networks with 1M neurons

Software Engineering Information Flow analysis of programs with 10K lines of code

The road to promised land: Theory + Algorithms + Software Development

Where do we go from here?

B. Cook, 2022: Virtuous cycle: ...application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. **Around and around**.

Today's Counters $\approx$ SAT Solvers in early 2000s

The storm is coming: Statistical systems are being integrated into our lives, the Pentium FDIV and Ariane 5 Rocket moments are inevitable if we do not act

Mission 2028: 100$\times$ Speedup for Counting to enable Quantitative Reasoning at Scale

Challenge Problems

Civil Engineering Rigorous resilience estimation for power grid of Los Angeles

Neural Network Verification Neural networks with 1M neurons

Software Engineering Information Flow analysis of programs with 10K lines of code

The road to promised land: Theory + Algorithms + Software Development

Where to start?: Here! At IIT Bombay. **IIT Bombay Formal Methods group is hiring!**

These slides are available at `tinyurl.com/meel-talk`