Stochastic Boolean Satisfiability

Decision Procedures, Generalization, and Applications

Nian-Ze Lee

Advisor: Prof. Jie-Hong Roland Jiang

Graduate Institute of Electronics Engineering, National Taiwan University

Doctoral Dissertation Oral Defense, 2nd June 2021

Outline

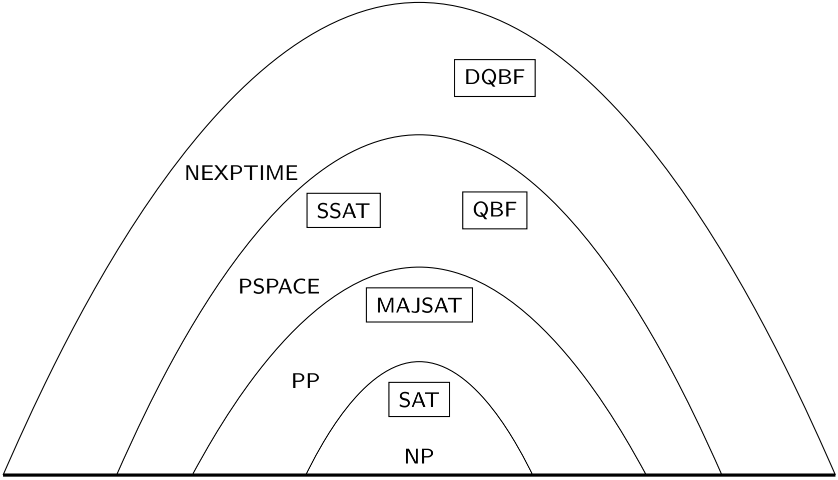
Outline

Outline

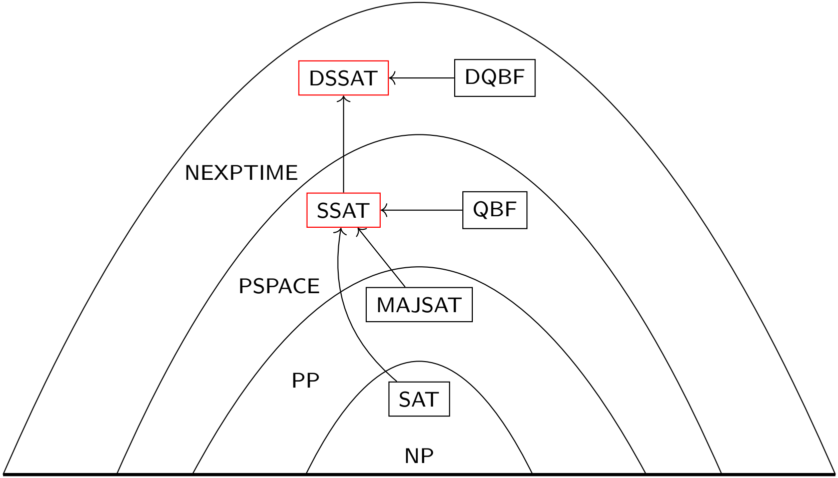
Satisfiability Solving: A Success Story

- Satisfiability solvers [6] have succeeded in various fields
  - Artificial intelligence [50, 55]
  - Electronic design automation [43, 65]
  - Formal verification [4, 27]

Satisfiability beyond Propositional Logic



This Dissertation in a Nutshell



Decision Making under Uncertainty

- Stochastic Boolean satisfiability (SSAT)
  - *Games against nature* [51]
  - Randomized quantifier $\reflectbox{R}^{p} x$: $\Pr[x = \textsc{true}] = p$
  - Logical formalism for problems with uncertainty
    - Probabilistic planning [40–42] and POMDP [56]

Decision Making under Uncertainty

- Stochastic Boolean satisfiability (SSAT)
  - *Games against nature* [51]
  - Randomized quantifier $\reflectbox{R}^{p}x$: $\Pr[x = \textsc{true}] = p$
  - Logical formalism for problems with uncertainty
    - Probabilistic planning [40–42] and POMDP [56]
- Application to VLSI systems?
  - Conventionally: error detection [15] or correction [46]
  - Post-Moore: probabilistic behavior of devices [12]

Accepting Device Imperfection

- New computational paradigms
  - Approximate design: deterministic deviation
    - E.g., neural-network deployment to edge devices
    - Circuit architectures [28, 29, 66]
    - Performance analysis [38, 64]
    - Automatic synthesis [44, 45, 48, 53, 63]
  - Probabilistic design: nondeterministic deviation
    - E.g., low-power video decoding
    - Energy consumption vs. correct switching of probabilistic CMOS [12]

Analyzing Probabilistic Design

- Circuit reliability analysis
  - Permanent defects or transient faults
  - Error probability at primary outputs
  - Monte Carlo simulation [47] or statistical methods [3, 31, 54]
  - Inadequate to analyze probabilistic design
    - Single-gate failure
    - Average error rate

Analyzing Probabilistic Design

- Circuit reliability analysis
  - Permanent defects or transient faults
  - Error probability at primary outputs
  - Monte Carlo simulation [47] or statistical methods [3, 31, 54]
  - Inadequate to analyze probabilistic design
    - Single-gate failure
    - Average error rate
- Research need: a framework to analyze probabilistic design
  - Design space exploration
  - Fault-tolerant applications
  - Intrinsically probabilistic systems
  - SSAT is a suitable logical formalism

State-of-the-Art SSAT Solving

- DPLL search [19]
  - MAXPLAN [41]: pure variables and unit propagation
  - ZANDER [42]: threshold-pruning heuristics and memorization
  - DC-SSAT [40]: divide-and-conquer (structure of planning problems)

State-of-the-Art SSAT Solving

- DPLL search [19]
  - `MAXPLAN` [41]: pure variables and unit propagation
  - `ZANDER` [42]: threshold-pruning heuristics and memorization
  - `DC-SSAT` [40]: divide-and-conquer (structure of planning problems)
- Knowledge compilation [18]
  - `ComPlan` [26]: deterministic, decomposable NNF (d-DNNF) [16, 17]

State-of-the-Art SSAT Solving

- DPLL search [19]
  - `MAXPLAN` [41]: pure variables and unit propagation
  - `ZANDER` [42]: threshold-pruning heuristics and memorization
  - `DC-SSAT` [40]: divide-and-conquer (structure of planning problems)
- Knowledge compilation [18]
  - `ComPlan` [26]: deterministic, decomposable NNF (d-DNNF) [16, 17]
- Closely related to model counting (MAJSAT) and QBF
  - Randomized quantifier: weighted summation of satisfying assignments
  - PSPACE-complete [61]: the same as QBF

State-of-the-Art SSAT Solving

- DPLL search [19]
  - `MAXPLAN` [41]: pure variables and unit propagation
  - `ZANDER` [42]: threshold-pruning heuristics and memorization
  - `DC-SSAT` [40]: divide-and-conquer (structure of planning problems)
- Knowledge compilation [18]
  - `ComPlan` [26]: deterministic, decomposable NNF (d-DNNF) [16, 17]
- Closely related to model counting (MAJSAT) and QBF
  - Randomized quantifier: weighted summation of satisfying assignments
  - PSPACE-complete [61]: the same as QBF
- Research need: novel algorithms for SSAT solving
  - Leverage advancements of other formalisms

Problems beyond PSPACE-Completeness

- SSAT is limited within PSPACE-completeness

Problems beyond PSPACE-Completeness

- SSAT is limited within PSPACE-completeness
- NEXPTIME-complete [52] problems with randomness
    - E.g., decentralized POMDP (Dec-POMDP) [5]
    - Difficult to obtain succinct encodings using SSAT

Problems beyond PSPACE-Completeness

- SSAT is limited within PSPACE-completeness
- NEXPTIME-complete [52] problems with randomness
  - E.g., decentralized POMDP (Dec-POMDP) [5]
  - Difficult to obtain succinct encodings using SSAT
- Research need: modeling NEXPTIME problems with uncertainty
  - Extend DQBF to stochastic domain

Difficulty in Algorithm Comparison

- Most SSAT work was done before 2005 [39–42]
  - Open-source solvers and formula instances are barely available

Difficulty in Algorithm Comparison

- Most SSAT work was done before 2005 [39–42]
  - Open-source solvers and formula instances are barely available
- Research need: public SSAT solvers and instances
  - Convenient comparison of different algorithms

Outline

Contributing to the Research Needs

- Probabilistic-design analysis: probabilistic property evaluation
  - Random-exist/Exist-random SSAT for average-/worst-case analysis

[1] https://github.com/NTU-ALComLab/ssatABC

[2] https://github.com/NTU-ALComLab/ssat-benchmarks

Contributing to the Research Needs

- Probabilistic-design analysis: probabilistic property evaluation
  - Random-exist/Exist-random SSAT for average-/worst-case analysis
- Novel algorithms: random-exist (RE) and exist-random (ER) SSAT
  - Modern techniques of SAT, model counting, and QBF
  - Approximate SSAT solving

[1] https://github.com/NTU-ALComLab/ssatABC

[2] https://github.com/NTU-ALComLab/ssat-benchmarks

Contributing to the Research Needs

- Probabilistic-design analysis: probabilistic property evaluation
  - Random-exist/Exist-random SSAT for average-/worst-case analysis
- Novel algorithms: random-exist (RE) and exist-random (ER) SSAT
  - Modern techniques of SAT, model counting, and QBF
  - Approximate SSAT solving
- NEXPTIME problems with uncertainty: dependency SSAT
  - The same NEXPTIME-complete complexity as DQBF
  - Applications to probabilistic/approximate design and Dec-POMDP

[1] https://github.com/NTU-ALComLab/ssatABC

[2] https://github.com/NTU-ALComLab/ssat-benchmarks

Contributing to the Research Needs

- Probabilistic-design analysis: probabilistic property evaluation
  - Random-exist/Exist-random SSAT for average-/worst-case analysis
- Novel algorithms: random-exist (RE) and exist-random (ER) SSAT
  - Modern techniques of SAT, model counting, and QBF
  - Approximate SSAT solving
- NEXPTIME problems with uncertainty: dependency SSAT
  - The same NEXPTIME-complete complexity as DQBF
  - Applications to probabilistic/approximate design and Dec-POMDP
- Algorithm evaluation: open-source solver[1] and benchmark set[2]
  - Benchmark set will become public after necessary licenses are added

[1] https://github.com/NTU-ALComLab/ssatABC
[2] https://github.com/NTU-ALComLab/ssat-benchmarks

Outline

Dissertation Structure

- The dissertation is based on the following publications
  - Chapter 4: probabilistic property evaluation
    - Published at ICCAD '14 [33] and in Trans. Computers '18 [34]
  - Chapter 5: random-exist quantified SSAT solving
    - Published at IJCAI '17 [36]
  - Chapter 6: exist-random quantified SSAT solving
    - Published at IJCAI '18 [37]
  - Chapter 7: dependency SSAT
    - Published at AAAI '21 [35]

[3] `https://github.com/sosy-lab/benchexec`

Dissertation Structure

- The dissertation is based on the following publications
  - Chapter 4: probabilistic property evaluation
    - Published at ICCAD'14 [33] and in Trans. Computers'18 [34]
  - Chapter 5: random-exist quantified SSAT solving
    - Published at IJCAI'17 [36]
  - Chapter 6: exist-random quantified SSAT solving
    - Published at IJCAI'18 [37]
  - Chapter 7: dependency SSAT
    - Published at AAAI'21 [35]
- Repeat the experiments with `BenchExec`[3]
  - Precise measurement: reproducibility
  - Data visualization

[3] `https://github.com/sosy-lab/benchexec`

Outline

Outline

Propositional Logic

Symbol	Description
τ	An assignment (a mapping from a variable set to $\mathbb{B}$)
ϕ	A quantifier-free formula
$\tau \models \phi$	τ satisfies ϕ
$\phi\|_x, \phi\|_{\neg x}$	Positive and negative cofactors of ϕ w.r.t. x
$\phi\|_\tau$	The resultant formula after cofactoring ϕ with τ

Outline

Stochastic Boolean Satisfiability

Definition: SSAT

Given a quantified formula $\Phi = Q_1 x_1, \ldots, Q_n x_n.\phi$:

- $Q_1 x_1, \ldots, Q_n x_n$: quantification structure, $Q_i \in \{\text{Я}^p, \exists\}$ (*prefix*)
- ϕ: quantifier-free formula over $\{x_1, \ldots, x_n\}$ (*matrix*)

Stochastic Boolean Satisfiability

Definition: SSAT

Given a quantified formula $\Phi = Q_1 x_1, \ldots, Q_n x_n . \phi$:

- $Q_1 x_1, \ldots, Q_n x_n$: quantification structure, $Q_i \in \{\text{Ɐ}^p, \exists\}$ (*prefix*)
- ϕ: quantifier-free formula over $\{x_1, \ldots, x_n\}$ (*matrix*)

Definition: Satisfying Probability of SSAT

Given an SSAT formula Φ, $\Pr[\Phi]$ is computed by:

1. $\Pr[\top] = 1$
2. $\Pr[\bot] = 0$
3. $\Pr[\Phi] = \max\{\Pr[\Phi|_{\neg x}], \Pr[\Phi|_x]\}$, if x is quantified by $\exists$
4. $\Pr[\Phi] = (1-p)\Pr[\Phi|_{\neg x}] + p\Pr[\Phi|_x]$, if x is quantified by Ɐ^p

Example: Satisfying Probability of SSAT

$$\Phi = \reflectbox{R}^{0.5} x_1, \exists y_1, \reflectbox{R}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

Stochastic Boolean Satisfiability

Example: Satisfying Probability of SSAT

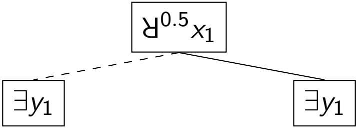
$$\Phi = \text{Я}^{0.5} x_1, \exists y_1, \text{Я}^{0.5} x_2, \exists y_2 . \phi$$
$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

$$\boxed{\text{Я}^{0.5} x_1}$$

Example: Satisfying Probability of SSAT

$$\Phi = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \exists y_1, \rotatebox[origin=c]{180}{R}^{0.5} x_2, \exists y_2 . \phi$$

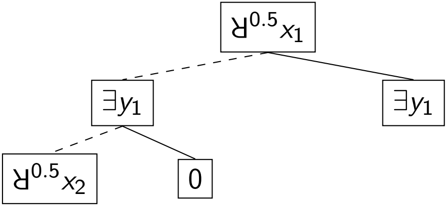
$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$



Example: Satisfying Probability of SSAT

$$\Phi = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \exists y_1, \rotatebox[origin=c]{180}{R}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

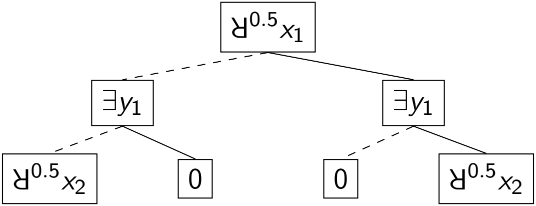


Stochastic Boolean Satisfiability

Example: Satisfying Probability of SSAT

$$\Phi = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \exists y_1, \rotatebox[origin=c]{180}{R}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

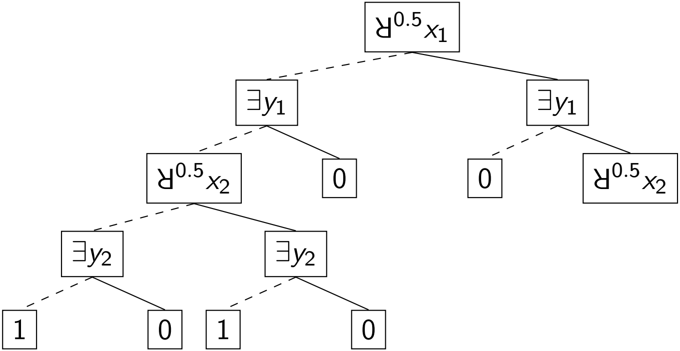


Stochastic Boolean Satisfiability

Example: Satisfying Probability of SSAT

$$\Phi = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \exists y_1, \rotatebox[origin=c]{180}{R}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

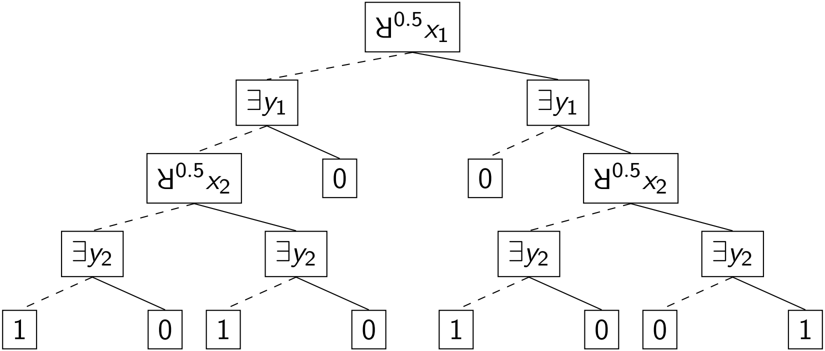


Stochastic Boolean Satisfiability

Example: Satisfying Probability of SSAT

$$\Phi = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \exists y_1, \rotatebox[origin=c]{180}{R}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

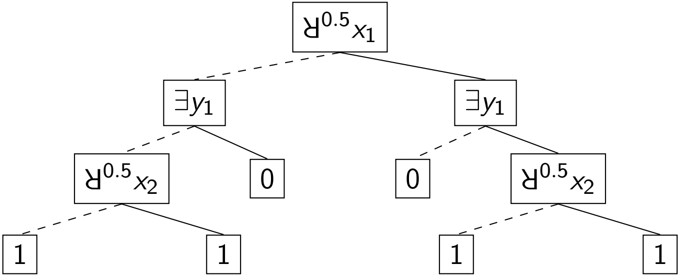


Stochastic Boolean Satisfiability

Example: Satisfying Probability of SSAT

$$\Phi = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \exists y_1, \rotatebox[origin=c]{180}{R}^{0.5} x_2, \exists y_2 . \phi$$

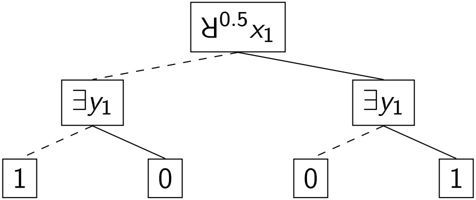
$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$



Example: Satisfying Probability of SSAT

$$\Phi = \reflectbox{R}^{0.5} x_1, \exists y_1, \reflectbox{R}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

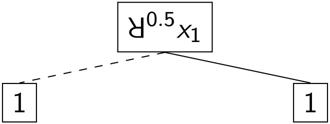


Stochastic Boolean Satisfiability

Example: Satisfying Probability of SSAT

$$\Phi = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \exists y_1, \rotatebox[origin=c]{180}{R}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$



Example: Satisfying Probability of SSAT

$$\Phi = \text{Я}^{0.5} x_1, \exists y_1, \text{Я}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

$$\boxed{\Pr[\Phi] = 1}$$

Game-Theoretical Interpretations of SSAT

- $\Phi = Q_1 x_1, \ldots, Q_n x_n.\phi, Q_i \in \{\reflectbox{R}^p, \exists\}$
  - $\reflectbox{R}^p$: nondeterministic factors
  - $\exists$: an agent who plays under uncertainty
  - ϕ: game matrix
  - $\Pr[\Phi]$: the maximum winning probability of the agent
  - Skolem functions: a strategy of the agent
  - Optimal Skolem functions: maximize the winning probability

Game-Theoretical Interpretations of SSAT

Example: Optimal Skolem Functions

$$\Phi = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \exists y_1, \rotatebox[origin=c]{180}{R}^{0.5} x_2, \exists y_2 . \phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

- Variable y_1: $f_1(x_1) = x_1$; variable y_2: $f_2(x_1, x_2) = x_1 \wedge x_2$

Outline

Model Counting

Definition: Unweighted Model Counting

Given a Boolean formula ϕ:

- Exact: find $\#\phi$
- Approximate: $\Pr[(1+\epsilon)^{-1}\#\phi \leq A \leq (1+\epsilon)\#\phi] \geq 1-\delta$

Model Counting

Definition: Unweighted Model Counting

Given a Boolean formula ϕ:

- Exact: find $\#\phi$
- Approximate: $\Pr[(1+\epsilon)^{-1}\#\phi \leq A \leq (1+\epsilon)\#\phi] \geq 1-\delta$

Definition: Weighted Model Counting

Given ϕ and $\omega : \mathtt{vars}(\phi) \mapsto [0,1]$:

- Weight of ϕ: sum of the weights of the satisfying assignments
  - Weight of x: $\omega(x)$
  - Weight of $\neg x$: $1-\omega(x)$
  - Weight of τ: product of the weights of the individual literals

Model Counting

- Exact
  - `Cachet` [57, 58]: DPLL search plus subformula caching
  - `c2d` [16, 17]: CNF-to-d-DNNF compilation
  - `DPMC` [20]: project-join tree and arithmetic decision diagrams

Model Counting

- Exact
  - `Cachet` [57, 58]: DPLL search plus subformula caching
  - `c2d` [16, 17]: CNF-to-d-DNNF compilation
  - `DPMC` [20]: project-join tree and arithmetic decision diagrams
- Approximate
  - `ApproxMC` [10, 11]: sampling with XOR constraints

Model Counting

- Exact
  - `Cachet` [57, 58]: DPLL search plus subformula caching
  - `c2d` [16, 17]: CNF-to-d-DNNF compilation
  - `DPMC` [20]: project-join tree and arithmetic decision diagrams
- Approximate
  - `ApproxMC` [10, 11]: sampling with XOR constraints
- Variants: expressible by SSAT
  - Weighted model counting [13, 59]
  - Projected model counting [2]
  - Maximum model counting [23]
  - Weighted projected model counting
    - `ProCount` [21]: ordering of projected and non-projected variables

Express Model-Counting Variants with SSAT

Variant	SSAT encoding
Unweighted	$\reflectbox{R}^{0.5} x_1, \ldots, \reflectbox{R}^{0.5} x_n . \phi$
Weighted	$\reflectbox{R}^{p_1} x_1, \ldots, \reflectbox{R}^{p_n} x_n . \phi$
Projected	$\reflectbox{R}^{0.5} x_1, \ldots, \reflectbox{R}^{0.5} x_n, \exists y_1, \ldots, \exists y_m . \phi$
Maximum	$\exists x_1, \ldots, \exists x_n, \reflectbox{R}^{0.5} y_1, \ldots, \reflectbox{R}^{0.5} y_m . \phi$
Weighted projected	$\reflectbox{R}^{p_1} x_1, \ldots, \reflectbox{R}^{p_n} x_n, \exists y_1, \ldots, \exists y_m . \phi$
Maximum weighted	$\exists x_1, \ldots, \exists x_n, \reflectbox{R}^{p_1} y_1, \ldots, \reflectbox{R}^{p_m} y_m . \phi$

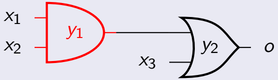
Outline

Outline

Modeling Probabilistic Design

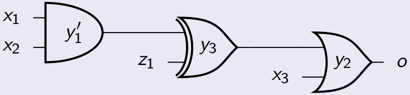
Example: Probabilistic Boolean Network and Standardization

A PBN with $V_I = \{x_1, x_2, x_3\}$ and $V_O = \{o\}$:



- $p_{x_1} = p_{x_2} = p_{x_3} = 0.5; p_{y_1} = 0.25; p_{y_2} = p_o = 0$

After standardization:

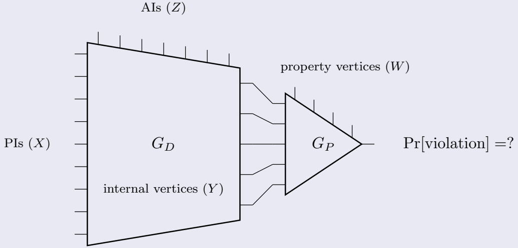


- $p_{z_1} = 0.25; p_{y_1'} = p_{y_3} = 0$

Outline

Probabilistic Property Evaluation

Definition: Property Violation Probability



- PPE: the average violation probability under $\pi : X \mapsto [0,1]$

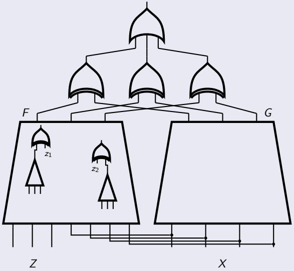
$$\reflectbox{R}^{\pi(x_1)} x_1, \ldots, \reflectbox{R}^{\pi(x_n)} x_n, \reflectbox{R}^{p_{z_1}} z_1, \ldots, \reflectbox{R}^{p_{z_l}} z_l, \reflectbox{R}^{p_{w_1}} w_1, \ldots, \reflectbox{R}^{p_{w_q}} w_q, \exists y_1, \ldots, \exists y_m . \phi_M$$

- MPPE: the maximum violation probability

$$\exists x_1, \ldots, \exists x_n, \reflectbox{R}^{p_{z_1}} z_1, \ldots, \reflectbox{R}^{p_{z_l}} z_l, \reflectbox{R}^{p_{w_1}} w_1, \ldots, \reflectbox{R}^{p_{w_q}} w_q, \exists y_1, \ldots, \exists y_m . \phi_M$$

Probabilistic Property Evaluation

Example: Probabilistic Equivalence Checking



- PEC: the average difference probability under $\pi : X \mapsto [0,1]$
$$\forall X, \forall Z, \exists Y.(F(X,Z) \neq G(X))$$
- MPEC: the maximum difference probability
$$\exists X, \forall Z, \exists Y.(F(X,Z) \neq G(X))$$

Solving MPPE and PPE

- MPPE: SSAT
  - CNF-based
  - BDD-based: graph traversal

Solving MPPE and PPE

- MPPE: SSAT
  - CNF-based
  - BDD-based: graph traversal
- PPE: model counting
  - Weighted model counting
  - Unweighted model counting with formula rewriting
    - Approximate model counting mostly focuses on unweighted instances
    - Express a weight of the form $\frac{k}{2^n}$ with additional variables and clauses

Outline

Evaluation

- Compared approaches
  - SSAT formulation (MPPE and PPE)
    - `BDDsp`: `C` language using `CUDD` [60] inside `ABC` [7]
    - `BDDsp-nr`: `BDDsp` without variable reordering
    - `DC-SSAT`: state-of-the-art CNF-based solver
  - Model-counting formulation (PPE)
    - `Cachet`: exact weighted model counter
    - `ApproxMC-4.0.1`: epsilon=0.99 and delta=0.01

Evaluation

- Compared approaches
  - SSAT formulation (MPPE and PPE)
    - `BDDsp`: `C` language using `CUDD` [60] inside `ABC` [7]
    - `BDDsp-nr`: `BDDsp` without variable reordering
    - `DC-SSAT`: state-of-the-art CNF-based solver
  - Model-counting formulation (PPE)
    - `Cachet`: exact weighted model counter
    - `ApproxMC`-4.0.1: epsilon=0.99 and delta=0.01
- Experimental setup
  - A machine with one 2.2 GHz CPU (Intel Xeon Silver 4210) with 40 processing units and 134 616 MB of RAM
  - Ubuntu 20.04 (64 bit), running Linux 5.4
  - CPU time: 15 min; memory: 15 GB

Benchmark Set

- Probabilistic equivalence checking
  - Average case: PEC
  - Worst case: MPEC

Benchmark Set

- Probabilistic equivalence checking
  - Average case: PEC
  - Worst case: MPEC
- ISCAS '85 [8] and EPFL [1] benchmark suites
  - And-inverter graphs (AIGs)
  - 30 circuits with sizes from 100 to 100K gates
  - Error rate: $\epsilon = 0.125$
  - Defect rate: $\delta = 0.01$ and 0.1

- BDDsp performs the best for small- and medium-sized circuits

Implications from the Results

- `BDDsp` performs the best for small- and medium-sized circuits
- `ApproxMC` uniquely solves large instances

Implications from the Results

- `BDDsp` performs the best for small- and medium-sized circuits
- `ApproxMC` uniquely solves large instances
- `Cachet` and `DC-SSAT` do not scale well

Results for PEC ($\delta = 0.01$)

Circuit	BDDsp T (s)	BDDsp Pr	DC-SSAT T (s)	DC-SSAT Pr	Cachet T (s)	Cachet Pr	ApproxMC T (s)	ApproxMC Pr
adder	3.71e−1	7.28e−1	–	–	–	–	6.70e+2	7.19e−1
bar	7.04e+0	9.85e−1	–	–	–	–	5.84e+2	1.00e+0
c1355	5.20e+0	4.32e−1	–	–	–	–	2.95e+1	4.30e−1
c1908	4.93e−1	6.25e−2	–	–	–	–	1.35e+1	6.05e−2
c2670	2.67e−1	3.10e−1	–	–	–	–	4.75e+2	3.13e−1
c3540	8.26e+0	2.28e−1	–	–	–	–	4.39e+1	2.30e−1
c432	6.19e−2	3.15e−2	–	–	2.87e−1	3.15e−2	1.17e+1	3.13e−2
c499	2.14e+0	2.62e−1	–	–	–	–	1.98e+1	2.66e−1
c5315	6.57e+1	6.53e−1	–	–	–	–	4.52e+2	6.56e−1
c6288	–	–	–	–	–	–	6.93e+1	9.06e−1
c7552	–	–	–	–	–	–	6.54e+2	7.03e−1
c880	6.31e−1	1.23e−1	–	–	1.10e+1	1.23e−1	3.17e+1	1.25e−1
cavlc	4.86e−2	4.96e−2	1.51e−1	4.96e−2	1.06e−1	4.96e−2	1.76e+1	4.98e−2
ctrl	4.44e−2	1.87e−1	9.67e−3	1.87e−1	3.52e−2	1.87e−1	1.05e+0	1.88e−1
dec	4.32e−2	6.56e−1	6.66e−3	6.56e−1	3.74e−2	6.56e−1	6.03e+0	6.56e−1
i2c	8.01e−2	4.33e−1	–	–	7.42e+2	4.33e−1	3.12e+2	4.22e−1
int2float	4.20e−2	6.39e−3	1.12e−2	6.39e−3	3.95e−2	6.39e−3	1.04e+0	6.47e−3
priority	1.16e−1	3.93e−1	–	–	–	–	1.70e+2	3.91e−1
router	5.06e−2	9.40e−4	–	–	6.84e+0	9.40e−4	3.69e+1	9.16e−4
sin	–	–	–	–	–	–	4.90e+2	1.00e+0

Results for MPEC ($\delta = 0.01$)

Circuit	BDDsp T (s)	BDDsp Pr	BDDsp-nr T (s)	BDDsp-nr Pr	DC-SSAT T (s)	DC-SSAT Pr
adder	$3.65e+0$	$7.99e-1$	–	–	–	–
c1355	$2.23e+1$	$6.56e-1$	$1.58e+0$	$6.56e-1$	–	–
c1908	$9.23e-1$	$4.14e-1$	$2.32e-1$	$4.14e-1$	$4.78e+1$	$4.14e-1$
c2670	$3.15e-1$	$5.51e-1$	–	–	–	–
c3540	$2.82e+1$	$6.07e-1$	$1.92e+1$	$6.07e-1$	–	–
c432	$6.40e-2$	$2.34e-1$	$4.99e-2$	$2.34e-1$	–	–
c499	$3.32e+0$	$4.14e-1$	$3.91e-1$	$4.14e-1$	–	–
c880	$6.71e-1$	$3.30e-1$	$9.85e-1$	$3.30e-1$	–	–
cavlc	$1.51e+0$	$5.42e-1$	$4.58e-2$	$5.42e-1$	$1.46e-1$	$5.42e-1$
ctrl	$4.53e-2$	$2.34e-1$	$4.45e-2$	$2.34e-1$	$7.27e-3$	$2.34e-1$
dec	$4.55e-2$	$6.56e-1$	$4.55e-2$	$6.56e-1$	$6.06e-3$	$6.56e-1$
i2c	–	–	$3.65e-1$	$8.57e-1$	–	–
int2float	$4.63e-2$	$2.34e-1$	$4.14e-2$	$2.34e-1$	$1.31e-2$	$2.34e-1$
priority	$1.78e+0$	$6.34e-1$	$6.86e-2$	$6.34e-1$	–	–
router	$5.39e-2$	$5.42e-1$	$4.84e-2$	$5.42e-1$	–	–

Outline

Outline

Definition: RE-SSAT

$\Phi = \text{Я} X, \exists Y. \phi(X, Y)$:

- X and Y: two disjoint sets of Boolean variables
- $\phi(X, Y)$: a CNF formula

Preliminaries

Definition: RE-SSAT

$\Phi = \reflectbox{R}X, \exists Y.\phi(X, Y)$:

- X and Y: two disjoint sets of Boolean variables
- $\phi(X, Y)$: a CNF formula

Definition: SAT/UNSAT Minterms and Cubes

Given $\Phi = \reflectbox{R}X, \exists Y.\phi(X, Y)$ and an assignment τ over X:

- $\phi(X, Y)|_\tau$ is satisfiable: τ is a SAT minterm of ϕ over X
- $\phi(X, Y)|_\tau$ is unsatisfiable: τ is an UNSAT minterm of ϕ over X
- τ is a partial assignment: τ is a SAT or an UNSAT cube

Generalization of SAT and UNSAT Minterms

- Given $\Phi = \reflectbox{R} X, \exists Y.\phi(X, Y)$
  - SAT minterm τ over X
    - Find a minimum subset of literals from τ that satisfy all clauses
    - Also known as *minimum hitting set*
  - UNSAT minterm τ over X
    - Modern SAT solvers: conflict analysis
    - $\phi(X, Y)|_\tau$ is UNSAT: a subset of literals from τ
    - Also known as *minimum UNSAT core*

Outline

Decision Procedure

- Given $\Phi = \reflectbox{R}X, \exists Y.\phi(X, Y)$, $\Pr[\Phi]$ equals
  - Sum of weights of all SAT minterms over X
  - One minus sum of weights of all UNSAT minterms over X

Decision Procedure

- Given $\Phi = \text{Ɐ}X, \exists Y.\phi(X, Y)$, $\Pr[\Phi]$ equals
  - Sum of weights of all SAT minterms over X
  - One minus sum of weights of all UNSAT minterms over X
- Avoid exhaustive enumeration by minterm generalization
  - Overlapping cubes: weights cannot be summed up directly
  - Handle overlap by weighted model counting

Decision Procedure

- Given $\Phi = \reflectbox{R} X, \exists Y . \phi(X, Y)$, $\Pr[\Phi]$ equals
  - Sum of weights of all SAT minterms over X
  - One minus sum of weights of all UNSAT minterms over X
- Avoid exhaustive enumeration by minterm generalization
  - Overlapping cubes: weights cannot be summed up directly
  - Handle overlap by weighted model counting
- The collected cubes reflect bounds of $\Pr[\Phi]$
  - SAT cubes: lower bound
  - UNSAT cubes: upper bound

Decision Procedure

Example: RE-SSAT Solving

Consider $\text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3, \exists e_1, \exists e_2, \exists e_3 . \phi$ with

$C_1 : (r_1 \vee r_2 \vee e_1);\ C_2 : (r_1 \vee \neg r_3 \vee e_2);\ C_3 : (r_2 \vee \neg r_3 \vee \neg e_1 \vee \neg e_2)$

$C_4 : (r_3 \vee e_3);\ C_5 : (r_3 \vee \neg e_3)$

Decision Procedure

Example: RE-SSAT Solving

Consider $\text{Ɐ}^{0.5} r_1, \text{Ɐ}^{0.5} r_2, \text{Ɐ}^{0.5} r_3, \exists e_1, \exists e_2, \exists e_3 . \phi$ with

$C_1 : (r_1 \vee r_2 \vee e_1);\ C_2 : (r_1 \vee \neg r_3 \vee e_2);\ C_3 : (r_2 \vee \neg r_3 \vee \neg e_1 \vee \neg e_2)$

$C_4 : (r_3 \vee e_3);\ C_5 : (r_3 \vee \neg e_3)$

Assignment	Minterm Type	Generalization	UB	LB
$\tau_1 = \neg r_1 \neg r_2 \neg r_3$	UNSAT	$\tau_1^+ = \neg r_3$	0.5	0

Decision Procedure

Example: RE-SSAT Solving

Consider $\text{Ɐ}^{0.5} r_1, \text{Ɐ}^{0.5} r_2, \text{Ɐ}^{0.5} r_3, \exists e_1, \exists e_2, \exists e_3 . \phi$ with

$$C_1 : (r_1 \vee r_2 \vee e_1);\ C_2 : (r_1 \vee \neg r_3 \vee e_2);\ C_3 : (r_2 \vee \neg r_3 \vee \neg e_1 \vee \neg e_2)$$

$$C_4 : (r_3 \vee e_3);\ C_5 : (r_3 \vee \neg e_3)$$

Assignment	Minterm Type	Generalization	UB	LB
$\tau_1 = \neg r_1 \neg r_2 \neg r_3$	UNSAT	$\tau_1^+ = \neg r_3$	0.5	0
$\tau_2 = \neg r_1 \neg r_2 r_3$	UNSAT	$\tau_2^+ = \neg r_1 \neg r_2$	0.375	0

Decision Procedure

Example: RE-SSAT Solving

Consider $\text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3, \exists e_1, \exists e_2, \exists e_3 . \phi$ with

$$C_1 : (r_1 \vee r_2 \vee e_1);\ C_2 : (r_1 \vee \neg r_3 \vee e_2);\ C_3 : (r_2 \vee \neg r_3 \vee \neg e_1 \vee \neg e_2)$$

$$C_4 : (r_3 \vee e_3);\ C_5 : (r_3 \vee \neg e_3)$$

Assignment	Minterm Type	Generalization	UB	LB
$\tau_1 = \neg r_1 \neg r_2 \neg r_3$	UNSAT	$\tau_1^+ = \neg r_3$	0.5	0
$\tau_2 = \neg r_1 \neg r_2 r_3$	UNSAT	$\tau_2^+ = \neg r_1 \neg r_2$	0.375	0
$\tau_3 = \neg r_1 r_2 r_3$	SAT	$\tau_3^+ = r_2 r_3$	0.375	0.25

Decision Procedure

Example: RE-SSAT Solving

Consider $\text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3, \exists e_1, \exists e_2, \exists e_3 . \phi$ with

$$C_1 : (r_1 \vee r_2 \vee e_1);\ C_2 : (r_1 \vee \neg r_3 \vee e_2);\ C_3 : (r_2 \vee \neg r_3 \vee \neg e_1 \vee \neg e_2)$$

$$C_4 : (r_3 \vee e_3);\ C_5 : (r_3 \vee \neg e_3)$$

Assignment	Minterm Type	Generalization	UB	LB
$\tau_1 = \neg r_1 \neg r_2 \neg r_3$	UNSAT	$\tau_1^+ = \neg r_3$	0.5	0
$\tau_2 = \neg r_1 \neg r_2 r_3$	UNSAT	$\tau_2^+ = \neg r_1 \neg r_2$	0.375	0
$\tau_3 = \neg r_1 r_2 r_3$	SAT	$\tau_3^+ = r_2 r_3$	0.375	0.25
$\tau_4 = r_1 \neg r_2 r_3$	SAT	$\tau_4^+ = r_1 r_3$	0.375	0.375

Example: RE-SSAT Solving

Consider $\text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3, \exists e_1, \exists e_2, \exists e_3.\phi$ with

$C_1 : (r_1 \vee r_2 \vee e_1);\ C_2 : (r_1 \vee \neg r_3 \vee e_2);\ C_3 : (r_2 \vee \neg r_3 \vee \neg e_1 \vee \neg e_2)$

$C_4 : (r_3 \vee e_3);\ C_5 : (r_3 \vee \neg e_3)$

Assignment	Minterm Type	Generalization	UB	LB
$\tau_1 = \neg r_1 \neg r_2 \neg r_3$	UNSAT	$\tau_1^+ = \neg r_3$	0.5	0
$\tau_2 = \neg r_1 \neg r_2 r_3$	UNSAT	$\tau_2^+ = \neg r_1 \neg r_2$	0.375	0
$\tau_3 = \neg r_1 r_2 r_3$	SAT	$\tau_3^+ = r_2 r_3$	0.375	0.25
$\tau_4 = r_1 \neg r_2 r_3$	SAT	$\tau_4^+ = r_1 r_3$	0.375	0.375

- $C_\perp = \{\tau_1^+, \tau_2^+\} = \{\neg r_3, \neg r_1 \neg r_2\}$
- $C_\top = \{\tau_3^+, \tau_4^+\} = \{r_2 r_3, r_1 r_3\}$
- $\Pr[\Phi] = 0.375$

Outline

Evaluation

- Compared solvers
  - `reSSAT`: `C++` language inside `ABC` environment [7]
    - SAT solver: `MiniSat-2.2` [22]
    - Weighted model counter: `Cachet` [57]
  - `reSSAT-b`: `reSSAT` without minterm generalization
  - `DC-SSAT`: state-of-the-art DPLL-based SSAT solver

Evaluation

- Compared solvers
  - `reSSAT`: `C++` language inside `ABC` environment [7]
    - SAT solver: `MiniSat-2.2` [22]
    - Weighted model counter: `Cachet` [57]
  - `reSSAT-b`: `reSSAT` without minterm generalization
  - `DC-SSAT`: state-of-the-art DPLL-based SSAT solver
- Experimental setup
  - A machine with one 2.2 GHz CPU (Intel Xeon Silver 4210) with 40 processing units and 134 616 MB of RAM
  - Ubuntu 20.04 (64 bit), running Linux 5.4
  - CPU time: 15 min; memory: 15 GB

Benchmark Set

- Random k-CNF formulas (by `CNFgen` [32])
  - $k \in \{3, \ldots, 9\}, n \in \{10, 20, \ldots, 50\}, \frac{m}{n} = \{k-1, k, k+1, k+2\}$
  - Quantify half variables randomly and others existentially
  - 5 samples per configuration: 700 formulas

Benchmark Set

- Random k-CNF formulas (by `CNFgen` [32])
  - $k \in \{3, \ldots, 9\}, n \in \{10, 20, \ldots, 50\}, \frac{m}{n} = \{k-1, k, k+1, k+2\}$
  - Quantify half variables randomly and others existentially
  - 5 samples per configuration: 700 formulas
- Strategic-company [9] formulas
  - Forall-exist QBF: decide whether a company is *strategic*
  - Replace universal quantifiers with randomized ones
  - From `QBFLIB` [49]: 60 formulas

Benchmark Set

- Random k-CNF formulas (by `CNFgen` [32])
  - $k \in \{3, \ldots, 9\}, n \in \{10, 20, \ldots, 50\}, \frac{m}{n} = \{k-1, k, k+1, k+2\}$
  - Quantify half variables randomly and others existentially
  - 5 samples per configuration: 700 formulas
- Strategic-company [9] formulas
  - Forall-exist QBF: decide whether a company is *strategic*
  - Replace universal quantifiers with randomized ones
  - From `QBFLIB` [49]: 60 formulas
- PEC formulas: 60 formulas

Implications from the Results

- `reSSAT` outperforms `DC-SSAT` on random and strategic formulas

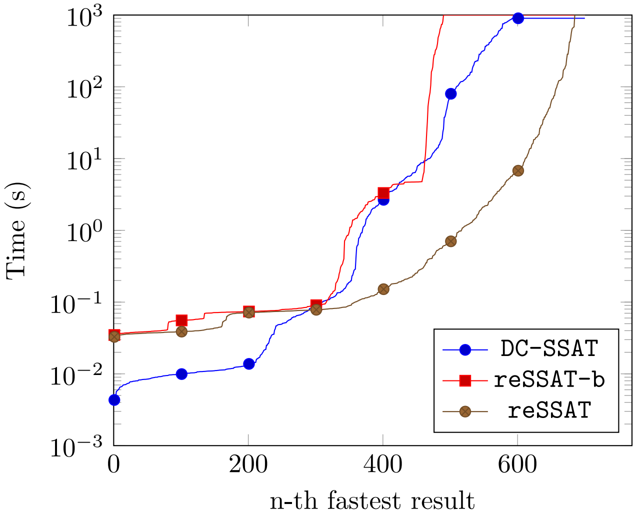
Implications from the Results

- `reSSAT` outperforms `DC-SSAT` on random and strategic formulas
- `reSSAT` is able to derive non-trivial bounds when `DC-SSAT` timeouts

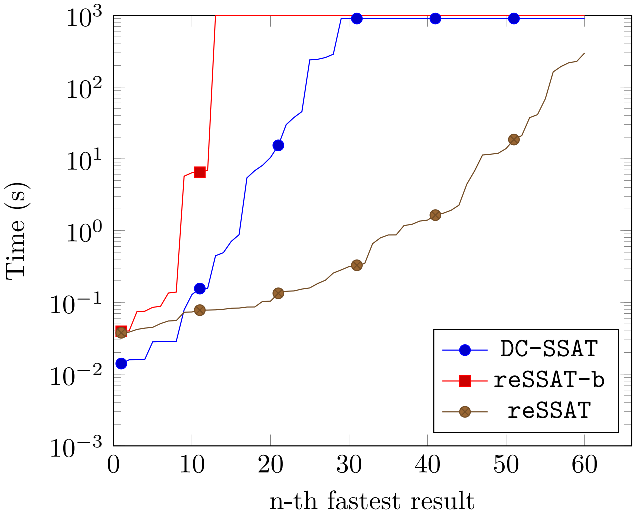
Implications from the Results

- `reSSAT` outperforms `DC-SSAT` on random and strategic formulas
- `reSSAT` is able to derive non-trivial bounds when `DC-SSAT` timeouts
- Minterm generalization is crucial to the performance of `reSSAT`

Quantile Plot for Random Formulas



Quantile Plot for Strategic-Company Formulas



Results for PEC Formulas ($\delta = 0.01$)

Formula	DC-SSAT T (s)	DC-SSAT Pr	reSSAT T (s)	reSSAT Pr	reSSAT UB	reSSAT-b T (s)	reSSAT-b Pr	reSSAT-b UB
adder	–	–	–	–	$7.98e{-1}$	–	–	–
bar	–	–	–	–	$9.98e{-1}$	–	–	–
c1908	–	–	–	–	–	–	–	$2.65e{-1}$
c2670	–	–	–	–	$5.03e{-1}$	–	–	$1.00e{+0}$
c3540	–	–	–	–	$2.98e{-1}$	–	–	$9.96e{-1}$
c432	–	–	–	–	$3.15e{-2}$	–	–	$5.92e{-1}$
c5315	–	–	–	–	$9.66e{-1}$	–	–	–
c6288	–	–	–	–	$1.00e{+0}$	–	–	$1.00e{+0}$
c880	–	–	–	–	$1.74e{-1}$	–	–	$8.83e{-1}$
cavlc	$1.60e{-1}$	$4.96e{-2}$	–	–	$4.96e{-2}$	–	–	$4.96e{-2}$
ctrl	$1.20e{-2}$	$1.87e{-1}$	$7.65e{-2}$	$1.87e{-1}$	–	$7.94e{-2}$	$1.87e{-1}$	–
dec	$9.19e{-3}$	$6.56e{-1}$	$7.46e{-2}$	$6.56e{-1}$	–	$1.00e{+1}$	$6.56e{-1}$	–
i2c	–	–	–	–	$8.18e{-1}$	–	–	$8.07e{-1}$
int2float	$1.84e{-2}$	$6.39e{-3}$	$8.49e{-2}$	$6.39e{-3}$	–	$1.08e{-1}$	$6.39e{-3}$	–
max	–	–	–	–	$9.74e{-1}$	–	–	–
priority	–	–	–	–	$7.61e{-1}$	–	–	–
router	–	–	–	–	$1.41e{-3}$	–	–	$2.41e{-1}$

Outline

Outline

Preliminaries

Definition: ER-SSAT (E-MAJSAT)

$\Phi = \exists X, \rotatebox[origin=c]{180}{R} Y.\phi(X, Y)$:

- X and Y: two disjoint sets of Boolean variables
- $\phi(X, Y)$: a CNF formula

Definition: ER-SSAT (E-MAJSAT)

$\Phi = \exists X, \text{Ⅎ} Y.\phi(X, Y)$:

- X and Y: two disjoint sets of Boolean variables
- $\phi(X, Y)$: a CNF formula

Definition: Clause Selection

Given a CNF formula $\phi(X, Y)$:

- $C = C^X \vee C^Y$
- An assignment τ over X *selects* C if τ falsifies every literal in C^X
- Selection variable: $s_C \equiv \neg C^X$
- Selection relation: $\psi(X, S) = \bigwedge_{C \in \phi} (s_C \equiv \neg C^X)$

Outline

Decision Procedure

- Given $\Phi = \exists X, \text{Ɐ} Y.\phi(X, Y)$, $\Pr[\Phi]$ equals
  - The maximum conditional satisfying probability $\Pr[\Phi|_{\tau^*}]$

Decision Procedure

- Given $\Phi = \exists X, \reflectbox{R} Y.\phi(X, Y)$, Pr[$\Phi$] equals
  - The maximum conditional satisfying probability $\Pr[\Phi|_{\tau^*}]$
- Clause-containment learning
  - Prune assignments that select a superset of the selected clauses
    - An assignments τ_1 selects a set of clauses $\phi|_{\tau_1}$
    - $\phi|_{\tau_1} \subseteq \phi|_{\tau_2} \implies (\phi|_{\tau_2} \rightarrow \phi|_{\tau_1}) \implies \Pr[\Phi|_{\tau_2}] \leq \Pr[\Phi|_{\tau_1}]$
    - Learnt clause: $\bigvee_{C \in \phi|_{\tau_1}} \neg s_C$

Decision Procedure

Example: E-MAJSAT Solving

Consider $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3.\phi$ with

$C_1 : (e_1 \vee r_1 \vee r_2)$ $C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$

$C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3)$ $C_4 : (\neg e_1 \vee e_3 \vee r_3)$

Decision Procedure

Example: E-MAJSAT Solving

Consider $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3.\phi$ with

$$C_1 : (e_1 \vee r_1 \vee r_2) \; C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$$

$$C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3) \; C_4 : (\neg e_1 \vee e_3 \vee r_3)$$

The selection relation $\psi(X, S)$ of $\phi(X, Y)$ equals

$$(s_1 \equiv \neg e_1) \wedge (s_2 \equiv \neg e_1 \wedge \neg e_2) \wedge (s_3 \equiv e_2 \wedge e_3) \wedge (s_4 \equiv e_1 \wedge \neg e_3)$$

Decision Procedure

Example: E-MAJSAT Solving

Consider $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3 . \phi$ with

$$C_1 : (e_1 \vee r_1 \vee r_2) \;\; C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$$

$$C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3) \;\; C_4 : (\neg e_1 \vee e_3 \vee r_3)$$

The selection relation $\psi(X, S)$ of $\phi(X, Y)$ equals

$$(s_1 \equiv \neg e_1) \wedge (s_2 \equiv \neg e_1 \wedge \neg e_2) \wedge (s_3 \equiv e_2 \wedge e_3) \wedge (s_4 \equiv e_1 \wedge \neg e_3)$$

Assignment	Selected Clauses	$\Pr[\Phi\|_\tau]$	Learnt Clause	LB
$\tau_1 = \neg e_1 \neg e_2 \neg e_3$	$\{C_1, C_2\}$	0.75	$(\neg s_1 \vee \neg s_2)$	0.75

Example: E-MAJSAT Solving

Consider $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3 . \phi$ with

$$C_1 : (e_1 \vee r_1 \vee r_2) \; C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$$

$$C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3) \; C_4 : (\neg e_1 \vee e_3 \vee r_3)$$

The selection relation $\psi(X, S)$ of $\phi(X, Y)$ equals

$$(s_1 \equiv \neg e_1) \wedge (s_2 \equiv \neg e_1 \wedge \neg e_2) \wedge (s_3 \equiv e_2 \wedge e_3) \wedge (s_4 \equiv e_1 \wedge \neg e_3)$$

Assignment	Selected Clauses	$\Pr[\Phi\|_\tau]$	Learnt Clause	LB
$\tau_1 = \neg e_1 \neg e_2 \neg e_3$	$\{C_1, C_2\}$	0.75	$(\neg s_1 \vee \neg s_2)$	0.75
$\tau_2 = \neg e_1 e_2 \neg e_3$	$\{C_1\}$	0.75	$(\neg s_1)$	0.75

Example: E-MAJSAT Solving

Consider $\Phi = \exists e_1, \exists e_2, \exists e_3, \rotatebox[origin=c]{180}{R}^{0.5} r_1, \rotatebox[origin=c]{180}{R}^{0.5} r_2, \rotatebox[origin=c]{180}{R}^{0.5} r_3 . \phi$ with

$$C_1 : (e_1 \vee r_1 \vee r_2) \;\; C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$$

$$C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3) \;\; C_4 : (\neg e_1 \vee e_3 \vee r_3)$$

The selection relation $\psi(X, S)$ of $\phi(X, Y)$ equals

$$(s_1 \equiv \neg e_1) \wedge (s_2 \equiv \neg e_1 \wedge \neg e_2) \wedge (s_3 \equiv e_2 \wedge e_3) \wedge (s_4 \equiv e_1 \wedge \neg e_3)$$

Assignment	Selected Clauses	$\Pr[\Phi\|_\tau]$	Learnt Clause	LB
$\tau_1 = \neg e_1 \neg e_2 \neg e_3$	$\{C_1, C_2\}$	0.75	$(\neg s_1 \vee \neg s_2)$	0.75
$\tau_2 = \neg e_1 e_2 \neg e_3$	$\{C_1\}$	0.75	$(\neg s_1)$	0.75
$\tau_3 = e_1 e_2 \neg e_3$	$\{C_4\}$	0.5	$(\neg s_4)$	0.75

Decision Procedure

Example: E-MAJSAT Solving

Consider $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3.\phi$ with

$$C_1 : (e_1 \vee r_1 \vee r_2)\ C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$$

$$C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3)\ C_4 : (\neg e_1 \vee e_3 \vee r_3)$$

The selection relation $\psi(X, S)$ of $\phi(X, Y)$ equals

$$(s_1 \equiv \neg e_1) \wedge (s_2 \equiv \neg e_1 \wedge \neg e_2) \wedge (s_3 \equiv e_2 \wedge e_3) \wedge (s_4 \equiv e_1 \wedge \neg e_3)$$

Assignment	Selected Clauses	$\Pr[\Phi\vert_\tau]$	Learnt Clause	LB
$\tau_1 = \neg e_1 \neg e_2 \neg e_3$	$\{C_1, C_2\}$	0.75	$(\neg s_1 \vee \neg s_2)$	0.75
$\tau_2 = \neg e_1 e_2 \neg e_3$	$\{C_1\}$	0.75	$(\neg s_1)$	0.75
$\tau_3 = e_1 e_2 \neg e_3$	$\{C_4\}$	0.5	$(\neg s_4)$	0.75
$\tau_4 = e_1 e_2 e_3$	$\{C_3\}$	0.75	$(\neg s_3)$	0.75

Decision Procedure

Example: E-MAJSAT Solving

Consider $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Ⅎ}^{0.5} r_1, \text{Ⅎ}^{0.5} r_2, \text{Ⅎ}^{0.5} r_3 . \phi$ with

$$C_1 : (e_1 \vee r_1 \vee r_2) \; C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$$

$$C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3) \; C_4 : (\neg e_1 \vee e_3 \vee r_3)$$

The selection relation $\psi(X, S)$ of $\phi(X, Y)$ equals

$$(s_1 \equiv \neg e_1) \wedge (s_2 \equiv \neg e_1 \wedge \neg e_2) \wedge (s_3 \equiv e_2 \wedge e_3) \wedge (s_4 \equiv e_1 \wedge \neg e_3)$$

Assignment	Selected Clauses	$\Pr[\Phi\|_\tau]$	Learnt Clause	LB
$\tau_1 = \neg e_1 \neg e_2 \neg e_3$	$\{C_1, C_2\}$	0.75	$(\neg s_1 \vee \neg s_2)$	0.75
$\tau_2 = \neg e_1 e_2 \neg e_3$	$\{C_1\}$	0.75	$(\neg s_1)$	0.75
$\tau_3 = e_1 e_2 \neg e_3$	$\{C_4\}$	0.5	$(\neg s_4)$	0.75
$\tau_4 = e_1 e_2 e_3$	$\{C_3\}$	0.75	$(\neg s_3)$	0.75
$\tau_5 = e_1 \neg e_2 e_3$	$\{\}$	1	$()$	1

Decision Procedure

Example: E-MAJSAT Solving

Consider $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3.\phi$ with

$C_1 : (e_1 \vee r_1 \vee r_2)$ $C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$

$C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3)$ $C_4 : (\neg e_1 \vee e_3 \vee r_3)$

The selection relation $\psi(X, S)$ of $\phi(X, Y)$ equals

$(s_1 \equiv \neg e_1) \wedge (s_2 \equiv \neg e_1 \wedge \neg e_2) \wedge (s_3 \equiv e_2 \wedge e_3) \wedge (s_4 \equiv e_1 \wedge \neg e_3)$

Assignment	Selected Clauses	$\Pr[\Phi\|_\tau]$	Learnt Clause	LB
$\tau_1 = \neg e_1 \neg e_2 \neg e_3$	$\{C_1, C_2\}$	0.75	$(\neg s_1 \vee \neg s_2)$	0.75
$\tau_2 = \neg e_1 e_2 \neg e_3$	$\{C_1\}$	0.75	$(\neg s_1)$	0.75
$\tau_3 = e_1 e_2 \neg e_3$	$\{C_4\}$	0.5	$(\neg s_4)$	0.75
$\tau_4 = e_1 e_2 e_3$	$\{C_3\}$	0.75	$(\neg s_3)$	0.75
$\tau_5 = e_1 \neg e_2 e_3$	$\{\}$	1	$()$	1

Heuristics to Strengthen Learnt Clauses

- A shorter learnt clause prunes more search space
  - Minimal clause selection: SAT solving
  - Induced clause subsumption: syntax checking
  - Partial assignment pruning: model counting

Heuristics to Strengthen Learnt Clauses

- A shorter learnt clause prunes more search space
  - Minimal clause selection: SAT solving
  - Induced clause subsumption: syntax checking
  - Partial assignment pruning: model counting
- Extra reasoning effort vs. Effectiveness of strengthened learnt clauses

Outline

Evaluation

- Compared solvers
  - erSSAT: C++ language inside ABC [7]
    - SAT solver: MiniSat-2.2 [22]
    - Weighted model counter: Cachet [57] and CUDD [18]
  - erSSAT-b: erSSAT without heuristics to strengthen learnt clauses
  - DC-SSAT: state-of-the-art DPLL-based SSAT solver

Evaluation

- Compared solvers
  - erSSAT: C++ language inside ABC [7]
    - SAT solver: MiniSat-2.2 [22]
    - Weighted model counter: Cachet [57] and CUDD [18]
  - erSSAT-b: erSSAT without heuristics to strengthen learnt clauses
  - DC-SSAT: state-of-the-art DPLL-based SSAT solver
- Experimental setup
  - A machine with one 2.2 GHz CPU (Intel Xeon Silver 4210) with 40 processing units and 134 616 MB of RAM
  - Ubuntu 20.04 (64 bit), running Linux 5.4
  - CPU time: 15 min; memory: 15 GB

Benchmark Set

- Random k-CNF formulas (by `CNFgen` [32])
  - $k \in \{3, \ldots, 9\}, n \in \{10, 20, \ldots, 50\}, \frac{m}{n} = \{k-1, k, k+1, k+2\}$
  - Quantify half variables existentially and others randomly
  - 5 samples per configuration: 700 formulas

Benchmark Set

- Random k-CNF formulas (by `CNFgen` [32])
  - $k \in \{3, \ldots, 9\}, n \in \{10, 20, \ldots, 50\}, \frac{m}{n} = \{k-1, k, k+1, k+2\}$
  - Quantify half variables existentially and others randomly
  - 5 samples per configuration: 700 formulas
- Application formulas: 212 formulas

Family	Description	Number
Toilet-A	Adapted from exist-forall QBFs [49]	77
Conformant	Adapted from exist-forall QBFs [49]	24
Sand-Castle	A probabilistic planning problem [41]	25
Max-Count	Adapted from maximum model counting [23]	26
MPEC	Maximum probabilistic equivalence checking	60

Implications from the Results

- erSSAT performs similarly as DC-SSAT on random formulas

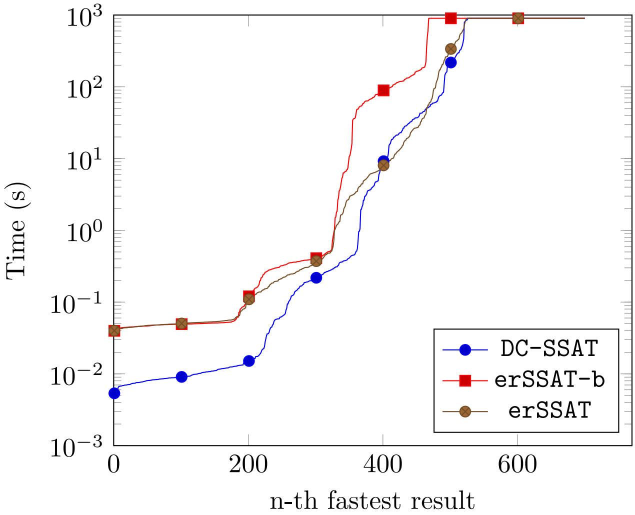
Implications from the Results

- erSSAT performs similarly as DC-SSAT on random formulas
- erSSAT is not suitable for certain application formulas
    - Clause-containment learning might degenerate to brute-force search
    - Overhead incurred by the heuristics to strengthen learnt clauses

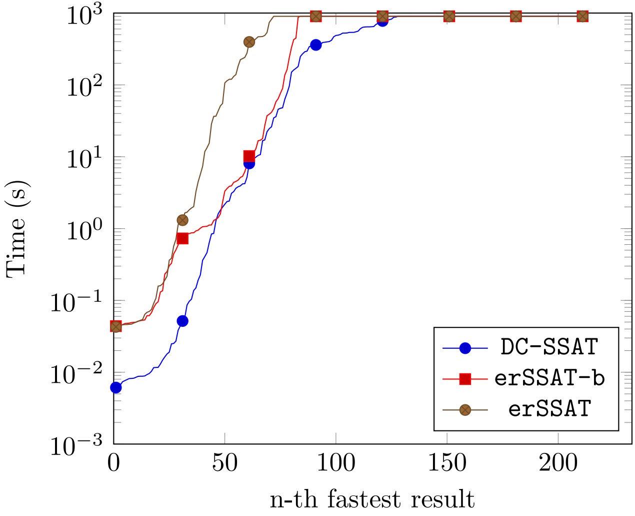
Implications from the Results

- erSSAT performs similarly as DC-SSAT on random formulas
- erSSAT is not suitable for certain application formulas
  - Clause-containment learning might degenerate to brute-force search
  - Overhead incurred by the heuristics to strengthen learnt clauses
- erSSAT is good at deriving tight lower bounds for large formulas

Quantile Plot for Random Formulas



Quantile Plot for Application Formulas



Summary of Results for Application Formulas

Algorithm	`DC-SSAT`	`erSSAT`	`erSSAT-b`
Solved formulas	78	59	65
Toilet-A	44	38	46
Conformant	1	2	1
Sand-Castle	22	13	14
Max-Count	3	3	1
MPEC	8	3	3
Timeouts	85	141	129
Out of memory	38	0	0
Other inconclusive	11	12	18

Summary of Results for Application Formulas

Algorithm	DC-SSAT	erSSAT	erSSAT-b
Solved formulas	78	59	65
Toilet-A	44	38	46
Conformant	1	2	1
Sand-Castle	22	13	14
Max-Count	3	3	1
MPEC	8	3	3
Timeouts	85	141	129
Out of memory	38	0	0
Other inconclusive	11	12	18

- An agent wants to build a sand castle within finite stages
  - At each stage: either dig a moat or erect a castle

Case Study: *Sand-Castle*

- An agent wants to build a sand castle within finite stages
  - At each stage: either dig a moat or erect a castle
- Conformant planning: E-MAJSAT
  - Existentially quantified variables: policy selection
  - Randomly quantified variables: probabilistic mechanism

Case Study: *Sand-Castle*

- An agent wants to build a sand castle within finite stages
  - At each stage: either dig a moat or erect a castle
- Conformant planning: E-MAJSAT
  - Existentially quantified variables: policy selection
  - Randomly quantified variables: probabilistic mechanism
- Two-stage formula: $(\neg d_1 \vee \phi_d^{(1)})(\neg e_1 \vee \phi_e^{(1)})(\neg d_2 \vee \phi_d^{(2)})(\neg e_2 \vee \phi_e^{(2)})$
  - Dig at stage 1 and erect at stage 2: $\phi_d^{(1)} \wedge \phi_e^{(2)}$
  - Each policy selects a distinct set of clauses
  - Clause-containment learning degenerates to brute-force search

Case Study: *Sand-Castle*

- An agent wants to build a sand castle within finite stages
  - At each stage: either dig a moat or erect a castle
- Conformant planning: E-MAJSAT
  - Existentially quantified variables: policy selection
  - Randomly quantified variables: probabilistic mechanism
- Two-stage formula: $(\neg d_1 \vee \phi_d^{(1)})(\neg e_1 \vee \phi_e^{(1)})(\neg d_2 \vee \phi_d^{(2)})(\neg e_2 \vee \phi_e^{(2)})$
  - Dig at stage 1 and erect at stage 2: $\phi_d^{(1)} \wedge \phi_e^{(2)}$
  - Each policy selects a distinct set of clauses
  - Clause-containment learning degenerates to brute-force search
- *Sand-Castle* favors `DC-SSAT`: same subformulas across the stages

Approximate E-MAJSAT Solving: erSSAT on *MPEC*

	DC-SSAT		erSSAT			
Formula	T (s)	Pr	T (s)	Pr	LB	T-LB (s)
c1355-0.01	–	–	–	–	4.14e−1	6.76e+2
c1908-0.01	4.80e+1	4.14e−1	–	–	3.18e−1	5.87e+2
c2670-0.01	–	–	–	–	4.87e−1	1.46e+2
c3540-0.01	–	–	–	–	–	–
c432-0.01	–	–	–	–	2.34e−1	1.41e+2
c499-0.01	–	–	–	–	4.14e−1	3.33e+1
c880-0.01	–	–	–	–	3.30e−1	4.14e+0
cavlc-0.01	1.49e−1	5.42e−1	–	–	–	–
ctrl-0.01	8.87e−3	2.34e−1	7.01e−2	2.34e−1	–	–
ctrl-0.10	5.77e−2	8.65e−1	–	–	–	–
dec-0.01	8.50e−3	6.56e−1	4.28e−2	6.56e−1	–	–
dec-0.10	1.81e+2	9.88e−1	–	–	–	–
i2c-0.01	–	–	–	–	–	–
int2float-0.01	1.17e−2	2.34e−1	1.31e+0	2.34e−1	–	–
int2float-0.10	4.22e+0	9.01e−1	–	–	–	–
priority-0.01	–	–	–	–	4.45e−1	1.40e+2
router-0.01	–	–	–	–	1.25e−1	3.60e−1

Outline

Outline

Preliminaries

Definition: Dependency QBF (DQBF)

$\Phi = \forall x_1, \ldots, \forall x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi$

- $D_{y_j} \subseteq \{x_1, \ldots, x_n\}$: the dependency set of y_j
- Φ is satisfiable if
  1. A Boolean function f_j over variables in D_{y_j} exists for each y_j
  2. ϕ becomes a tautology over $\{x_1, \ldots, x_n\}$ after substituting y_j with f_j

Preliminaries

Definition: Dependency QBF (DQBF)

$\Phi = \forall x_1, \ldots, \forall x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi$

- $D_{y_j} \subseteq \{x_1, \ldots, x_n\}$: the dependency set of y_j
- Φ is satisfiable if
  1. A Boolean function f_j over variables in D_{y_j} exists for each y_j
  2. ϕ becomes a tautology over $\{x_1, \ldots, x_n\}$ after substituting y_j with f_j

Example: DQBF

$$\Phi = \forall x_1, \forall x_2, \exists y_1(\{x_1\}), \exists y_2(\{x_1, x_2\}).\phi$$
$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

- Φ is satisfiable: $f_1(x_1) = x_1; f_2(x_1, x_2) = x_1 \wedge x_2$

Outline

Formulation

Definition: Dependency SSAT (DSSAT)

$\Phi = \reflectbox{R}^{p_1} x_1, \ldots, \reflectbox{R}^{p_n} x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi$

- Satisfying probability of Φ with respect to $\mathcal{F} = \{f_1, \ldots, f_m\}$

$$\Pr[\Phi|_{\mathcal{F}}] = \Pr[\reflectbox{R}^{p_1} x_1, \ldots, \reflectbox{R}^{p_n} x_n.\phi|_{\mathcal{F}}]$$

- Decision version: given Φ and θ decide if $\Pr[\Phi|_{\mathcal{F}}] \geq \theta$ for some $\mathcal{F}$
- Optimization version: find $\mathcal{F}$ to maximize $\Pr[\Phi|_{\mathcal{F}}]$

Formulation

Example: DSSAT

$$\Phi = \reflectbox{R}^{0.5} x_1, \reflectbox{R}^{0.5} x_2, \exists y_1(\{x_1\}), \exists y_2(\{x_2\}).\phi$$

$$\phi = (x_1 \vee \neg y_1)(\neg x_1 \vee y_1)(\neg x_1 \vee \neg x_2 \vee y_2)(x_1 \vee \neg y_2)(x_2 \vee \neg y_2)$$

$$\mathcal{F} = \{f_1(x_1) = x_1, f_2(x_2) = x_2\}$$

- $\Pr[\Phi|_{\mathcal{F}}] = \Pr[\reflectbox{R}^{0.5} x_1, \reflectbox{R}^{0.5} x_2.(x_1 \vee \neg x_2)] = 0.75$

Complexity

Theorem: NEXPTIME-Completeness of DSSAT

The decision version of DSSAT is NEXPTIME-complete

- NEXPTIME: guess $\mathcal{F}$ in exponential time
- NEXPTIME-hard: DQBF $\leq_P$ DSSAT

Benefits of DSSAT over DQBF

- Optimization vs. Decision
  - DSSAT: maximum satisfying probability
  - DQBF: satisfiability
  - Similar scenario: SSAT vs. QBF

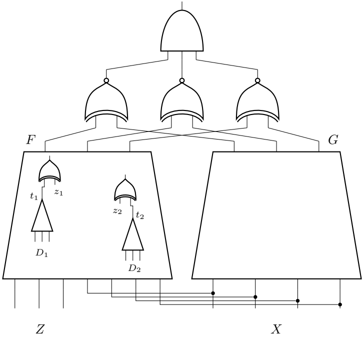
Benefits of DSSAT over DQBF

- Optimization vs. Decision
  - DSSAT: maximum satisfying probability
  - DQBF: satisfiability
  - Similar scenario: SSAT vs. QBF
- Stochastic vs. Deterministic
  - DSSAT: natural encoding for NEXPTIME problems with uncertainty
  - DQBF: less straightforward for problems with probabilities

Outline

Analyzing Probabilistic/Approximate Partial Design

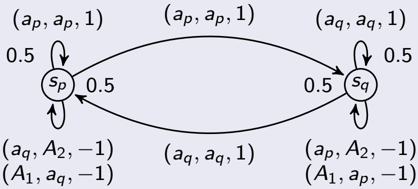
- Partial design problem [25] of probabilistic design
  - Synthesize black-box outputs T to realize specification
  - Constraints on inputs to black boxes: $D_i \subseteq X \cup Y$



$$\rotatebox[origin=c]{180}{R} X, \rotatebox[origin=c]{180}{R} Z, \forall Y, \exists T(D).(Y \equiv E(X)) \rightarrow (F(X, Z, T) \equiv G(X))$$

Modeling Decentralized POMDP

Example: Dec-POMDP



$$\mathcal{M} = (\{1, 2\}, \{s_p, s_q\}, \{a_p, a_q\}, T, \rho, \{o_p, o_q\}, \Omega, \Delta_0, h)$$

- $T(s_p, a_p, a_p, s_p) = T(s_p, a_p, a_p, s_q) = 0.5$
- $\rho(s_p, a_p, a_p) = 1; \rho(s_p, a_q, A_2) = \rho(s_p, A_1, a_q) = -1$
- $\Omega(s_p, o_p) = \Pr[o_p | s_p]$

Modeling Decentralized POMDP

- Optimal *joint policy* to maximize the expected total reward
  - An agent can only base its actions on its own observations
  - NEXPTIME-completeness [5]

Modeling Decentralized POMDP

- Optimal *joint policy* to maximize the expected total reward
  - An agent can only base its actions on its own observations
  - NEXPTIME-completeness [5]
- Policy selection for an individual agent [56]
  - Agent 1: $\exists x_a^{1,0}, \rotatebox[origin=c]{180}{R} x_o^{1,0}, \exists x_a^{1,1}, \rotatebox[origin=c]{180}{R} x_o^{1,1}, \exists x_a^{1,2}, \ldots$
  - Agent 2: $\exists x_a^{2,0}, \rotatebox[origin=c]{180}{R} x_o^{2,0}, \exists x_a^{2,1}, \rotatebox[origin=c]{180}{R} x_o^{2,1}, \exists x_a^{2,2}, \ldots$
  - A linearly ordered prefix for all agents?

Modeling Decentralized POMDP

- Optimal *joint policy* to maximize the expected total reward
  - An agent can only base its actions on its own observations
  - NEXPTIME-completeness [5]
- Policy selection for an individual agent [56]
  - Agent 1: $\exists x_a^{1,0}, \reflectbox{R} x_o^{1,0}, \exists x_a^{1,1}, \reflectbox{R} x_o^{1,1}, \exists x_a^{1,2}, \ldots$
  - Agent 2: $\exists x_a^{2,0}, \reflectbox{R} x_o^{2,0}, \exists x_a^{2,1}, \reflectbox{R} x_o^{2,1}, \exists x_a^{2,2}, \ldots$
  - A linearly ordered prefix for all agents?
- Explicitly specify dependency sets in DSSAT
  - $\reflectbox{R} x_o^{1,0}, \reflectbox{R} x_o^{2,0}, \reflectbox{R} x_o^{1,1}, \reflectbox{R} x_o^{2,1}, \exists x_a^{1,0}, \exists x_a^{1,1}(\{x_o^{1,0}\}), \exists x_a^{1,2}(\{x_o^{1,0}, x_o^{1,1}\}), \ldots$

Outline

Conclusion

- Probabilistic property evaluation
  - Average case: RE-SSAT
  - Worst case: ER-SSAT

Conclusion

- Probabilistic property evaluation
  - Average case: RE-SSAT
  - Worst case: ER-SSAT
- Novel algorithms for RE/ER-SSAT
  - Model counting
  - Minterm generalization from SAT
  - Clause selection from QBF

Conclusion

- Probabilistic property evaluation
    - Average case: RE-SSAT
    - Worst case: ER-SSAT
- Novel algorithms for RE/ER-SSAT
    - Model counting
    - Minterm generalization from SAT
    - Clause selection from QBF
- Dependency SSAT
    - NEXPTIME-completeness
    - Applications to probabilistic partial design and Dec-POMDP

Conclusion

- Probabilistic property evaluation
  - Average case: RE-SSAT
  - Worst case: ER-SSAT
- Novel algorithms for RE/ER-SSAT
  - Model counting
  - Minterm generalization from SAT
  - Clause selection from QBF
- Dependency SSAT
  - NEXPTIME-completeness
  - Applications to probabilistic partial design and Dec-POMDP
- Open-source solvers and instances

Future Work

- Approximate analysis for probabilistic design based on simulation
  - Monte Carlo method
  - Symbolic sampling [30]

Future Work

- Approximate analysis for probabilistic design based on simulation
  - Monte Carlo method
  - Symbolic sampling [30]
- General SSAT and DSSAT formulas
  - Clause selection for SSAT [14] and DQBF [62]

Future Work

- Approximate analysis for probabilistic design based on simulation
  - Monte Carlo method
  - Symbolic sampling [30]
- General SSAT and DSSAT formulas
  - Clause selection for SSAT [14] and DQBF [62]
- Tighter integration of model-counting component
  - More advanced data structure, e.g., d-DNNF [17]
  - Incremental model counting

Future Work

- Approximate analysis for probabilistic design based on simulation
  - Monte Carlo method
  - Symbolic sampling [30]
- General SSAT and DSSAT formulas
  - Clause selection for SSAT [14] and DQBF [62]
- Tighter integration of model-counting component
  - More advanced data structure, e.g., d-DNNF [17]
  - Incremental model counting
- Approximate SSAT solving
  - Trade inexactness for scalability

Future Work

- Approximate analysis for probabilistic design based on simulation
  - Monte Carlo method
  - Symbolic sampling [30]
- General SSAT and DSSAT formulas
  - Clause selection for SSAT [14] and DQBF [62]
- Tighter integration of model-counting component
  - More advanced data structure, e.g., d-DNNF [17]
  - Incremental model counting
- Approximate SSAT solving
  - Trade inexactness for scalability
- Machine-learning applications
  - Verify fairness of supervised learning [24]

References I

[1] L. Amarú, P.-E. Gaillardon, and G. De Micheli. 2015. The EPFL combinational benchmark suite. In *Proc. IWLS*. `https://www.epfl.ch/labs/lsi/page-102566-en-html/benchmarks/`

[2] R. A. Aziz, G. Chu, C. J. Muise, and P. J. Stuckey. 2015. $\#\exists$SAT: Projected model counting. In *Proc. SAT (LNCS 9340)*. Springer, 121–137. `https://doi.org/10.1007/978-3-319-24318-4_10`

[3] R. I. Bahar, J. L. Mundy, and J. Chen. 2003. A probabilistic-based design methodology for nanoscale computation. In *Proc. ICCAD*. IEEE Computer Society / ACM, 480–486. `https://doi.org/10.1109/ICCAD.2003.1257854`

[4] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. 2013. *Systems and Software Verification: Model-Checking Techniques and Tools* (1st ed.). Springer. `https://doi.org/10.1007/978-3-662-04558-9`

References II

[5] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research* 27, 4 (2002), 819–840. `https://doi.org/10.1287/moor.27.4.819.297`

[6] A. Biere, M. Heule, H. van Maaren, and T. Walsh (Eds.). 2009. *Handbook of Satisfiability*. Frontiers in Artificial Intelligence and Applications, Vol. 185. IOS Press.

[7] R. K. Brayton and A. Mishchenko. 2010. ABC: An academic industrial-strength verification tool. In *Proc. CAV (LNCS 6174)*. Springer, 24–40. `https://doi.org/10.1007/978-3-642-14295-6_5` Latest version available at: https://github.com/berkeley-abc/abc.

[8] F. Brglez and H. Fujiwara. 1985. A neutral netlist of 10 combinational benchmark circuits. In *Proc. ISCAS*. IEEE, 695–698. `https://people.engr.ncsu.edu/brglez/CBL/benchmarks/index.html`

References III

[9] M. Cadoli, T. Eiter, and G. Gottlob. 1997. Default logic as a query language. *IEEE Transactions on Knowledge and Data Engineering* 9, 3 (1997), 448–463. https://doi.org/10.1109/69.599933

[10] S. Chakraborty, K. S. Meel, and M. Y. Vardi. 2013. A scalable approximate model counter. In *Proc. CP (LNCS 8124)*. Springer, 200–216. https://doi.org/10.1007/978-3-642-40627-0_18

[11] S. Chakraborty, K. S. Meel, and M. Y. Vardi. 2016. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In *Proc. IJCAI*. IJCAI/AAAI Press, 3569–3576. http://www.ijcai.org/Abstract/16/503

[12] L. N. B. Chakrapani, J. George, B. Marr, B. E. S. Akgul, and K. V. Palem. 2006. Probabilistic design: A survey of probabilistic CMOS technology and future directions for terascale IC design. In *Proc. VLSI-SoC (IFIP 249)*. Springer, 101–118. https://doi.org/10.1007/978-0-387-74909-9_7

References IV

[13] M. Chavira and A. Darwiche. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence* 172, 6-7 (2008), 772–799. https://doi.org/10.1016/j.artint.2007.11.002

[14] P.-W. Chen, Y.-C. Huang, and J.-H. R. Jiang. 2021. A sharp leap from quantified Boolean formula to stochastic Boolean satisfiability solving. In *Proc. AAAI*. AAAI Press.

[15] C. Constantinescu. 2003. Trends and challenges in VLSI circuit reliability. *IEEE Micro* 23, 4 (2003), 14–19. https://doi.org/10.1109/MM.2003.1225959

[16] A. Darwiche. 2001. Decomposable negation normal form. *J. ACM* 48, 4 (2001), 608–647. https://doi.org/10.1145/502090.502091

[17] A. Darwiche. 2002. A compiler for deterministic, decomposable negation normal form. In *Proc. AAAI*. AAAI Press / The MIT Press, 627–634. http://www.aaai.org/Library/AAAI/2002/aaai02-094.php

References V

[18] A. Darwiche and P. Marquis. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17 (2002), 229–264. https://doi.org/10.1613/jair.989

[19] M. Davis, G. Logemann, and D. W. Loveland. 1962. A machine program for theorem-proving. *Commun. ACM* 5, 7 (1962), 394–397. https://doi.org/10.1145/368273.368557

[20] J. M. Dudek, V. H. N. Phan, and M. Y. Vardi. 2020. DPMC: Weighted Model Counting by Dynamic Programming on Project-Join Trees. In *Proc. CP (LNCS 12333)*. Springer, 211–230. https://doi.org/10.1007/978-3-030-58475-7_13

[21] J. M. Dudek, V. H. N. Phan, and M. Y. Vardi. 2021. ProCount: Weighted Projected Model Counting with Graded Project-Join Trees. In *Proc. SAT*.

References VI

[22] N. Eén and N. Sörensson. 2003. An extensible SAT-solver. In *Proc. SAT (LNCS 2919)*. Springer, 502–518. https://doi.org/10.1007/978-3-540-24605-3_37

[23] D. J. Fremont, M. N. Rabe, and S. A. Seshia. 2017. Maximum model counting. In *Proc. AAAI*. AAAI Press, 3885–3892. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14968

[24] B. Ghosh, D. Basu, and K. S. Meel. 2020. Justicia: A Stochastic SAT Approach to Formally Verify Fairness. *CoRR* abs/2009.06516 (2020). arXiv:2009.06516 https://arxiv.org/abs/2009.06516

[25] K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, and B. Becker. 2013. Equivalence checking of partial designs using dependency quantified Boolean formulae. In *Proc. ICCD*. 396–403. https://doi.org/10.1109/ICCD.2013.6657071

References VII

[26] J. Huang. 2006. Combining knowledge compilation and search for conformant probabilistic planning. In *Proc. ICAPS*. AAAI, 253–262. http://www.aaai.org/Library/ICAPS/2006/icaps06-026.php

[27] R. Jhala and R. Majumdar. 2009. Software model checking. *Comput. Surveys* 41, 4 (2009), 21:1–21:54. https://doi.org/10.1145/1592434.1592438

[28] A. B. Kahng and S. Kang. 2012. Accuracy-configurable adder for approximate arithmetic designs. In *Proc. DAC*. ACM, 820–825. https://doi.org/10.1145/2228360.2228509

[29] Y. Kim, Y. Zhang, and P. Li. 2013. An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems. In *Proc. ICCAD*. IEEE, 130–137. https://doi.org/10.1109/ICCAD.2013.6691108

[30] V. N. Kravets, N.-Z. Lee, and J.-H. R. Jiang. 2019. Comprehensive search for ECO rectification using symbolic sampling. In *Proc. DAC*. ACM, 71:1–71:6. `https://doi.org/10.1145/3316781.3317790`

[31] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes. 2005. Accurate reliability evaluation and enhancement via probabilistic transfer matrices. In *Proc. DATE*. IEEE Computer Society, 282–287. `https://doi.org/10.1109/DATE.2005.47`

[32] M. Lauria, J. Elffers, J. Nordström, and M. Vinyals. 2017. CNFgen: A generator of crafted benchmarks. In *Proc. SAT (LNCS 10491)*. Springer, 464–473. `https://doi.org/10.1007/978-3-319-66263-3_30`

[33] N.-Z. Lee and J.-H. R. Jiang. 2014. Towards formal evaluation and verification of probabilistic design. In *Proc. ICCAD*. IEEE, 340–347. `https://doi.org/10.1109/ICCAD.2014.7001372`

References IX

[34] N.-Z. Lee and J.-H. R. Jiang. 2018. Towards formal evaluation and verification of probabilistic design. *IEEE Trans. Comput.* 67, 8 (2018), 1202–1216. `https://doi.org/10.1109/TC.2018.2807431`

[35] N.-Z. Lee and J.-H. R. Jiang. 2021. Dependency stochastic Boolean satisfiability: A logical formalism for NEXPTIME decision problems with uncertainty. In *Proc. AAAI*. AAAI Press.

[36] N.-Z. Lee, Y.-S. Wang, and J.-H. R. Jiang. 2017. Solving stochastic Boolean satisfiability under random-exist quantification. In *Proc. IJCAI*. IJCAI Organization, 688–694. `https://doi.org/10.24963/ijcai.2017/96`

[37] N.-Z. Lee, Y.-S. Wang, and J.-H. R. Jiang. 2018. Solving exist-random quantified stochastic Boolean satisfiability via clause selection. In *Proc. IJCAI*. IJCAI Organization, 1339–1345. `https://doi.org/10.24963/ijcai.2018/186`

References X

[38] L. Li and H. Zhou. 2014. On error modeling and analysis of approximate adders. In *Proc. ICCAD*. IEEE, 511–518. https://doi.org/10.1109/ICCAD.2014.7001399

[39] S. M. Majercik. 2004. Nonchronological backtracking in stochastic Boolean satisfiability. In *Proc. ICTAI*. IEEE Computer Society, 498–507. https://doi.org/10.1109/ICTAI.2004.94

[40] S. M. Majercik and B. Boots. 2005. DC-SSAT: A divide-and-conquer approach to solving stochastic satisfiability problems efficiently. In *Proc. AAAI*. AAAI Press / The MIT Press, 416–422. http://www.aaai.org/Library/AAAI/2005/aaai05-066.php

[41] S. M. Majercik and M. L. Littman. 1998. MAXPLAN: A new approach to probabilistic planning. In *Proc. AIPS*. AAAI, 86–93. http://www.aaai.org/Library/AIPS/1998/aips98-011.php

[42] S. M. Majercik and M. L. Littman. 2003. Contingent planning under uncertainty via stochastic satisfiability. *Artificial Intelligence* 147, 1-2 (2003), 119–162.
https://doi.org/10.1016/S0004-3702(02)00379-X

[43] J. P. Marques-Silva and K. A. Sakallah. 2000. Boolean satisfiability in electronic design automation. In *Proc. DAC*. ACM, 675–680.
https://doi.org/10.1145/337292.337611

[44] J. Miao, A. Gerstlauer, and M. Orshansky. 2013. Approximate logic synthesis under general error magnitude and frequency constraints. In *Proc. ICCAD*. IEEE, 779–786.
https://doi.org/10.1109/ICCAD.2013.6691202

[45] J. Miao, A. Gerstlauer, and M. Orshansky. 2014. Multi-level approximate logic synthesis under general error constraints. In *Proc. ICCAD*. IEEE, 504–510.
https://doi.org/10.1109/ICCAD.2014.7001398

[46] S. Mitra, M. Zhang, S. Waqas, N. Seifert, B. S. Gill, and K. S. Kim. 2006. Combinational logic soft error correction. In *Proc. ITC*. IEEE Computer Society, 1–9.
https://doi.org/10.1109/TEST.2006.297681

[47] K. Mohanram and N. A. Touba. 2003. Cost-effective approach for reducing soft error failure rate in logic circuits. In *Proc. ITC*. IEEE Computer Society, 893–901.
https://doi.org/10.1109/TEST.2003.1271075

[48] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasícek, and K. Roy. 2016. Design of power-efficient approximate multipliers for approximate artificial neural networks. In *Proc. ICCAD*. ACM, 81:1–81:7.
https://doi.org/10.1145/2966986.2967021

[49] M. Narizzano, L. Pulina, and A. Tacchella. 2006. The QBFEVAL web portal. In *Proc. JELIA (LNCS 4160)*. Springer, 494–497.
https://doi.org/10.1007/11853886_45

References XIII

[50] N. J. Nilsson. 2014. *Principles of Artificial Intelligence*. Morgan Kaufmann.

[51] C. H. Papadimitriou. 1985. Games against nature. *J. Comput. System Sci.* 31, 2 (1985), 288–301. https://doi.org/10.1016/0022-0000(85)90045-5

[52] G. L. Peterson and J. H. Reif. 1979. Multiple-person alternation. In *Proc. FOCS*. IEEE Computer Society, 348–363. https://doi.org/10.1109/SFCS.1979.25

[53] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel. 2016. Architectural-space exploration of approximate multipliers. In *Proc. ICCAD*. ACM, 80:1–80:8. https://doi.org/10.1145/2966986.2967005

[54] T. Rejimon and S. Bhanja. 2005. Scalable probabilistic computing models using Bayesian networks. In *Proc. MWSCAS*. IEEE, 712–715. https://doi.org/10.1109/MWSCAS.2005.1594200

References XIV

[55] S. J. Russell and P. Norvig. 2020. *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

[56] R. Salmon and P. Poupart. 2020. On the relationship between stochastic satisfiability and Markov decision processes. In *Proc. UAI*. PMLR, 1105–1115.
http://proceedings.mlr.press/v115/salmon20a.html

[57] T. Sang, F. Bacchus, P. Beame, H. A. Kautz, and T. Pitassi. 2004. Combining component caching and clause learning for effective model counting. In *Proc. SAT*. 20–28.
http://www.satisfiability.org/SAT04/programme/21.pdf

[58] T. Sang, P. Beame, and H. A. Kautz. 2005. Heuristics for fast exact model counting. In *Proc. SAT (LNCS 3569)*. Springer, 226–240.
https://doi.org/10.1007/11499107_17

References XV

[59] T. Sang, P. Beame, and H. A. Kautz. 2005. Performing Bayesian inference by weighted model counting. In *Proc. AAAI*. AAAI Press / The MIT Press, 475–482.
`http://www.aaai.org/Library/AAAI/2005/aaai05-075.php`

[60] F. Somenzi. [n.d.]. CUDD: CU decision diagram package. http://vlsi.colorado.edu/ fabio/.

[61] L. J. Stockmeyer and A. R. Meyer. 1973. Word problems requiring exponential time: Preliminary report. In *Proc. STOC*. ACM, 1–9.
`https://doi.org/10.1145/800125.804029`

[62] L. Tentrup and M. N. Rabe. 2019. Clausal abstraction for DQBF. In *Proc. SAT (LNCS 11628)*. Springer, 388–405.
`https://doi.org/10.1007/978-3-030-24258-9_27`

References XVI

[63] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan. 2012. SALSA: Systematic logic synthesis of approximate circuits. In *Proc. DAC*. ACM, 796–801. https://doi.org/10.1145/2228360.2228504

[64] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan. 2011. MACACO: Modeling and analysis of circuits for approximate computing. In *Proc. ICCAD*. IEEE Computer Society, 667–673. https://doi.org/10.1109/ICCAD.2011.6105401

[65] L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng. 2009. *Electronic Design Automation: Synthesis, Verification, and Test* (1st ed.). Morgan Kaufmann.

[66] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu. 2013. On reconfiguration-oriented approximate adder design and its application. In *Proc. ICCAD*. IEEE, 48–54. https://doi.org/10.1109/ICCAD.2013.6691096

Thanks for your attention!
Questions?

BDD-Based SSAT Solving

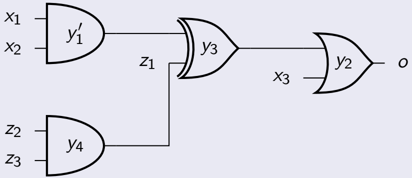
Input: An ROBDD node N and a prefix Q
Output: $\Pr[N = \top]$ under Q

```
if (N is a terminal node) then
    return N.sp
end if
if (N.visited = FALSE) then
    if (Q(N.var) = Ⴘ^p) then
        N.sp := (1 − p) · BddSsatRecur(N.else, Q) + p · BddSsatRecur(N.then, Q)
    else
        N.sp := max{BddSsatRecur(N.else, Q), BddSsatRecur(N.then, Q)}
    end if
    N.visited := TRUE
end if
return N.sp
```

Rewriting WMC into Unweighted MC

Example: WMC Rewriting

Express $p_{z_1} = 0.25$ with $z_1 \equiv z_2 \wedge z_3$:



- Unweighted: $p_{x_1} = p_{x_2} = p_{x_3} = p_{z_2} = p_{z_3} = 0.5$

WMC Rewriting Procedure

Input: A formula ϕ, a base set X_d for ϕ, a wt. func. ω s.t. $\forall x \in X_d . \omega(x) = \frac{k}{2^n}$

Output: A formula ϕ', a base set X'_d for ϕ', a wt. func. ω' s.t. $\forall x \in X'_d . \omega'(x) = \frac{1}{2}$

$\phi' := \phi, X'_d := X_d$
for all $(x \in X_d)$ **do**
    $var := x, wt := \omega(x)$
    while $(wt \neq \frac{1}{2})$ **do**
        $inv := \bot$
        if $(wt > \frac{1}{2})$ **then**
            $wt := 1 - wt, inv := \top$
        end if
        $\phi' := \phi' \wedge ((inv \oplus var) \equiv (y_{var} \wedge z_{var}))$
        $X'_d := X'_d \setminus \{var\} \cup \{y_{var}\}$
        $\omega'(y_{var}) = \frac{1}{2}$
        $var = z_{var}, wt = 2 \cdot wt$
    end while
    $X'_d := X'_d \cup \{var\}, \omega'(var) = \frac{1}{2}$
end for
return (ϕ', X'_d, ω')

Results for PEC ($\delta = 0.1$)

	BDDsp		DC-SSAT		Cachet		ApproxMC	
CIRCUIT	T (s)	Pr	T (s)	Pr	T (s)	Pr	T (s)	Pr
adder	1.77e+0	1.00e+0	–	–	–	–	–	–
c1355	–	–	–	–	–	–	2.92e+2	9.22e−1
c1908	–	–	–	–	–	–	2.49e+2	9.22e−1
c432	1.12e+1	4.99e−1	–	–	–	–	7.13e+1	4.84e−1
c499	–	–	–	–	–	–	1.89e+2	8.13e−1
c880	–	–	–	–	–	–	1.85e+2	8.75e−1
cavlc	2.58e−1	6.89e−1	–	–	1.77e+0	6.89e−1	3.32e+2	6.72e−1
ctrl	4.57e−2	8.22e−1	4.88e−2	8.22e−1	5.01e−2	8.22e−1	3.33e+1	8.28e−1
dec	5.38e−2	9.87e−1	1.84e+2	9.87e−1	1.34e+0	9.87e−1	8.17e+1	1.00e+0
int2float	5.70e−2	4.32e−1	4.30e+0	4.32e−1	3.55e−1	4.32e−1	7.27e+1	4.30e−1
router	3.60e−1	1.76e−1	–	–	–	–	1.56e+2	1.76e−1

Results for MPEC ($\delta = 0.1$)

	BDDsp		BDDsp-nr		DC-SSAT	
Circuit	T (s)	Pr	T (s)	Pr	T (s)	Pr
cavlc	–	–	$5.62e{-}2$	$9.78e{-}1$	–	–
ctrl	$4.87e{-}2$	$8.65e{-}1$	$4.61e{-}2$	$8.65e{-}1$	$5.83e{-}2$	$8.65e{-}1$
dec	$6.10e{-}2$	$9.88e{-}1$	$4.85e{-}2$	$9.88e{-}1$	$1.81e{+}2$	$9.88e{-}1$
int2float	–	–	$4.70e{-}2$	$9.01e{-}1$	$4.21e{+}0$	$9.01e{-}1$
router	–	–	$2.75e{+}0$	$8.96e{-}1$	–	–

Generalization of SAT Minterms

Example: SAT Generalization

Consider $\text{Ʀ}^{0.5} r_1, \text{Ʀ}^{0.5} r_2, \text{Ʀ}^{0.5} r_3, \exists e_1, \exists e_2, \exists e_3.\phi$ with ϕ consisting of the following clauses, and $\tau = \neg r_1 r_2 r_3$:

$$C_1 : (r_1 \vee r_2 \vee e_1);\ C_2 : (r_1 \vee \neg r_3 \vee e_2);\ C_3 : (r_2 \vee \neg r_3 \vee \neg e_1 \vee \neg e_2)$$
$$C_4 : (r_3 \vee e_3);\ C_5 : (r_3 \vee \neg e_3)$$

- τ is a SAT minterm: ϕ is satisfiable with τ and $\mu = \neg e_1 e_2 \neg e_3$
- Generalize τ to a SAT cube $\tau^+ = r_2 r_3$

Generalization of UNSAT Minterms

Example: UNSAT Generalization

Consider $\reflectbox{R}^{0.5} r_1, \reflectbox{R}^{0.5} r_2, \reflectbox{R}^{0.5} r_3, \exists e_1, \exists e_2, \exists e_3 . \phi$ with ϕ consisting of the following clauses, and $\tau = \neg r_1 \neg r_2 \neg r_3$:

$$C_1 : (r_1 \vee r_2 \vee e_1);\ C_2 : (r_1 \vee \neg r_3 \vee e_2);\ C_3 : (r_2 \vee \neg r_3 \vee \neg e_1 \vee \neg e_2)$$

$$C_4 : (r_3 \vee e_3);\ C_5 : (r_3 \vee \neg e_3)$$

- τ is an UNSAT minterm: C_4 and C_5 conflict
- Generalize τ to an UNSAT cube $\tau^+ = \neg r_3$

RE-SSAT Solving

Input: $\Phi = \text{\reflectbox{R}} X, \exists Y.\phi(X, Y)$ and a run-time limit $\texttt{TO}$
Output: Lower and upper bounds (P_L, P_U) of $\Pr[\Phi]$

$\psi(X) := \top$
$C_\top := \emptyset$
$C_\bot := \emptyset$
while $(\texttt{SAT}(\psi)$ **and** $\texttt{run-time} < \texttt{TO})$ **do**
    $\tau := \psi.\text{model}$
    if $(\texttt{SAT}(\phi|_\tau))$ **then**
        $\tau^+ := \texttt{MinimalSatisfying}(\phi, \tau)$
        $C_\top := C_\top \cup \{\tau^+\}$
    else
        $\tau^+ := \texttt{MinimalConflicting}(\phi, \tau)$
        $C_\bot := C_\bot \cup \{\tau^+\}$
    end if
    $\psi := \psi \wedge \neg\tau^+$
end while
return $(\texttt{ComputeWeight}(C_\top), 1 - \texttt{ComputeWeight}(C_\bot))$

Results for PEC Formulas ($\delta = 0.1$)

Formula	DC-SSAT T (s)	DC-SSAT Pr	reSSAT T (s)	reSSAT Pr	reSSAT UB	reSSAT-b T (s)	reSSAT-b Pr	reSSAT-b UB
c1908	–	–	–	–	$1.00e{+}0$	–	–	–
c3540	–	–	–	–	$1.00e{+}0$	–	–	–
c432	–	–	–	–	$7.81e{-}1$	–	–	$9.76e{-}1$
c880	–	–	–	–	$9.98e{-}1$	–	–	–
cavlc	–	–	–	–	$7.66e{-}1$	–	–	$8.88e{-}1$
ctrl	$5.27e{-}2$	$8.22e{-}1$	–	–	$8.49e{-}1$	–	–	$8.73e{-}1$
dec	$1.85e{+}2$	$9.87e{-}1$	–	–	$9.88e{-}1$	–	–	$9.99e{-}1$
i2c	–	–	–	–	$9.98e{-}1$	–	–	–
int2float	$4.33e{+}0$	$4.32e{-}1$	–	–	$5.22e{-}1$	–	–	$6.46e{-}1$
max	–	–	–	–	$1.00e{+}0$	–	–	–
priority	–	–	–	–	$1.00e{+}0$	–	–	–
router	–	–	–	–	$2.41e{-}1$	–	–	$3.10e{-}1$

Minimal Clause Selection

- Iteratively solve $\psi(X, S)$ to select a minimal set of clauses
- Recall $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Я}^{0.5} r_1, \text{Я}^{0.5} r_2, \text{Я}^{0.5} r_3 . \phi$ with

  $C_1 : (e_1 \vee r_1 \vee r_2)$ $C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$
  $C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3)$ $C_4 : (\neg e_1 \vee e_3 \vee r_3)$
- $\tau_1 = \neg e_1 \neg e_2 \neg e_3$ selects $\{C_1, C_2\}$
- Solve $\psi \wedge (\neg s_1 \vee \neg s_2)$ under assumptions $s_3 \wedge s_4$
  - $\tau_2 = \neg e_1 e_2 \neg e_3$, which only selects $\{C_1\}$
  - Replace an expensive model-counting call with a SAT call

Induced Clause Subsumption

- C_1 subsumes C_2 if $C_1 \subseteq C_2$ (treat C_1 and C_2 as sets of literals)
- Subsumed clauses can be removed from a CNF formula
- Recall $\Phi = \exists e_1, \exists e_2, \exists e_3, \text{Ⅎ}^{0.5} r_1, \text{Ⅎ}^{0.5} r_2, \text{Ⅎ}^{0.5} r_3.\phi$ with

  $C_1 : (e_1 \vee r_1 \vee r_2)$ $C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$
  $C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3)$ $C_4 : (\neg e_1 \vee e_3 \vee r_3)$
- $\tau_1 = \neg e_1 \neg e_2 \neg e_3$ selects $\{C_1, C_2\}$
- C_1^Y subsumes C_2^Y: remove C_2 from the set of selected clauses
  - Strengthen the learnt clause to $(\neg s_1)$
- Subsumption among C^Y clauses is *induced* by clause selection

Partial Assignment Pruning

- $C_L = \bigvee_{C \in \phi|_\tau} \neg s_C = \bigvee_{C \in \phi|_\tau} C^X$
- Discard literals from C_L by model counting
- Recall $\Phi = \exists e_1, \exists e_2, \exists e_3, \rotatebox[origin=c]{180}{R}^{0.5} r_1, \rotatebox[origin=c]{180}{R}^{0.5} r_2, \rotatebox[origin=c]{180}{R}^{0.5} r_3 . \phi$ with

  $C_1 : (e_1 \vee r_1 \vee r_2)$ $C_2 : (e_1 \vee e_2 \vee r_1 \vee r_2 \vee \neg r_3)$
  $C_3 : (\neg e_2 \vee \neg e_3 \vee r_2 \vee \neg r_3)$ $C_4 : (\neg e_1 \vee e_3 \vee r_3)$
- $\tau_1 = \neg e_1 \neg e_2 \neg e_3$: $C_L = (\neg s_1 \vee \neg s_2) = (e_1 \vee e_2), \Pr[\Phi|_{\tau_1}] = 0.75$
  - Can we discard e_2 from C_L?
  - (e_1) blocks τ that selects C_1: $\Pr[\Phi|_\tau] \leq \Pr[C_1] = 0.75 \leq \Pr[\Phi|_{\tau_1}]$
- Count weight of selected clauses and compare to current maximum

ER-SSAT Solving

Input: $\Phi = \exists X, \reflectbox{R} Y.\phi(X,Y)$
Output: $\Pr[\Phi]$

$\psi(X,S) := \bigwedge_{C \in \phi} (s_C \equiv \neg C^X) \wedge \bigwedge_{\text{pure } l : \texttt{var}(l) \in X} l$
$prob := 0$
$\texttt{s-table} := \texttt{BuildSubsumptionTable}(\phi)$
while $(\texttt{SAT}(\psi))$ **do**
  $\tau := \psi.\mathrm{model}$ (discarding the selection variables)
  if $(\texttt{SAT}(\phi|_\tau))$ **then**
    $\tau' := \texttt{SelectMinimalClauses}(\phi, \psi)$
    $\varphi := \texttt{RemoveSubsumedClauses}(\phi|_{\tau'}, \texttt{s-table})$
    $prob := \max\{prob, \texttt{ComputeWeight}(\reflectbox{R} Y.\varphi)\}$
    $C_S := \bigvee_{C \in \varphi} \neg s_C$
    $C_L := \texttt{DiscardLiterals}(\phi, C_S, prob)$
  else
    $C_L := \texttt{MinimalConflicting}(\phi, \tau)$
  end if
  $\psi := \psi \wedge C_L$
end while
return $prob$

Approximate E-MAJSAT Solving: erSSAT-b on *MPEC*

	DC-SSAT		erSSAT-b			
FORMULA	T (s)	Pr	T (s)	Pr	LB	T-LB (s)
c1355-0.01	–	–	–	–	$6.56e{-1}$	$4.10e{+1}$
c1908-0.01	$4.80e{+1}$	$4.14e{-1}$	–	–	$4.14e{-1}$	$2.23e{+1}$
c2670-0.01	–	–	–	–	$5.51e{-1}$	$5.74e{+0}$
c3540-0.01	–	–	–	–	$5.51e{-1}$	$7.20e{+1}$
c432-0.01	–	–	–	–	$2.34e{-1}$	$2.91e{+0}$
c499-0.01	–	–	–	–	$4.14e{-1}$	$3.80e{-1}$
c880-0.01	–	–	–	–	$3.30e{-1}$	$6.17e{+0}$
cavlc-0.01	$1.49e{-1}$	$5.42e{-1}$	–	–	–	–
ctrl-0.01	$8.87e{-3}$	$2.34e{-1}$	$8.98e{-2}$	$2.34e{-1}$	–	–
ctrl-0.10	$5.77e{-2}$	$8.65e{-1}$	–	–	–	–
dec-0.01	$8.50e{-3}$	$6.56e{-1}$	$4.58e{-2}$	$6.56e{-1}$	–	–
dec-0.10	$1.81e{+2}$	$9.88e{-1}$	–	–	–	–
i2c-0.01	–	–	–	–	$7.21e{-1}$	$1.75e{+2}$
int2float-0.01	$1.17e{-2}$	$2.34e{-1}$	$4.45e{-1}$	$2.34e{-1}$	–	–
int2float-0.10	$4.22e{+0}$	$9.01e{-1}$	–	–	–	–
priority-0.01	–	–	–	–	$5.89e{-1}$	$5.57e{+2}$
router-0.01	–	–	–	–	$1.25e{-1}$	$3.80e{-1}$

Computational Complexity of DSSAT

Theorem: NEXPTIME-Completeness of DSSAT

The decision version of DSSAT is NEXPTIME-complete

- DSSAT is NEXPTIME
- DSSAT is NEXPTIME-hard

Proof Sketch

- NEXPTIME
  1. nondeterministically construct a set of Skolem functions $\mathcal{F}$
  2. compute $\Pr[\Phi|_{\mathcal{F}}]$ and compare it with the threshold θ
- NEXPTIME-hard: DQBF $\leq_P$ DSSAT

$$\Phi_Q = \forall x_1, \ldots, \forall x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi$$

$$\Phi_S = \rotatebox[origin=c]{180}{R}^{0.5} x_1, \ldots, \rotatebox[origin=c]{180}{R}^{0.5} x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi$$

Claim: Φ_Q is satisfiable if and only if $\Pr[\Phi_S|_{\mathcal{F}}] \geq 1$ for some $\mathcal{F}$

Modeling Decentralized POMDP

$$\bigwedge_{0\leq t\leq h-2} [x_p^t \equiv \bot \rightarrow \bigwedge_{i\in I} x_o^{i,t} \equiv 0 \wedge x_s^{t+1} \equiv 0 \wedge x_p^{t+1} \equiv \bot]$$

$$x_p^{h-1} \equiv \bot$$

$$\bigwedge_{s\in S} \bigwedge_{\vec{a}\in\vec{A}} [x_p^0 \equiv \bot \wedge x_s^0 \equiv s \wedge \bigwedge_{i\in I} x_a^{i,0} \equiv a_i \rightarrow x_r^0 \equiv N_r(s,\vec{a})]$$

$$\bigwedge_{1\leq t\leq h-1} \bigwedge_{s\in S} \bigwedge_{\vec{a}\in\vec{A}} [x_p^{t-1} \equiv \top \wedge x_p^t \equiv \bot \wedge x_s^t \equiv s \wedge \bigwedge_{i\in I} x_a^{i,t} \equiv a_i \rightarrow x_r^t \equiv N_r(s,\vec{a})]$$

$$\bigwedge_{0\leq t\leq h-2} \bigwedge_{s\in S} \bigwedge_{\vec{a}\in\vec{A}} \bigwedge_{s'\in S} [x_p^t \equiv \top \wedge x_s^t \equiv s \wedge \bigwedge_{i\in I} x_a^{i,t} \equiv a_i \wedge x_s^{t+1} \equiv s' \rightarrow x_{T_{s,\vec{a}}}^t \equiv s']$$

$$\bigwedge_{0\leq t\leq h-2} \bigwedge_{s'\in S} \bigwedge_{\vec{a}\in\vec{A}} \bigwedge_{\vec{o}\in\vec{O}} [x_p^t \equiv \top \wedge x_s^{t+1} \equiv s' \wedge \bigwedge_{i\in I} x_a^{i,t} \equiv a_i \wedge \bigwedge_{i\in I} x_o^{i,t} \equiv o_i \rightarrow x_{\Omega_{s',\vec{a}}}^t \equiv N_\Omega(\vec{o})]$$