Model Counting for Logical Theories

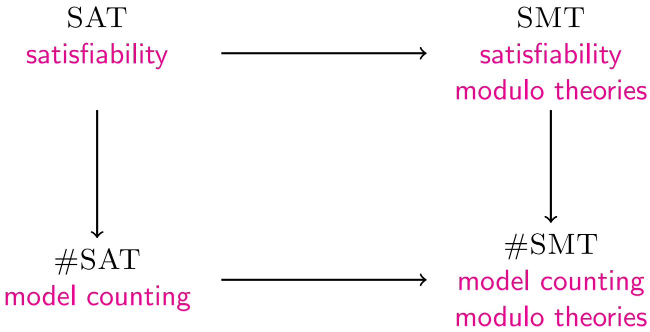
Friday

Dmitry Chistikov Rayna Dimitrova

Department of Computer Science
University of Oxford, UK

Max Planck Institute for Software Systems (MPI-SWS)
Kaiserslautern and Saarbrücken, Germany

ESSLLI 2016


SAT
satisfiability
SMT
satisfiability modulo theories
#SAT
model counting
#SMT
model counting modulo theories

Measures and measured theories

Measure μ for a σ-algebra $(D, \mathcal{F})$
maps each $A \in \mathcal{F}$ to a real number $\mu(A) \geq 0$:

$$\begin{array}{lcl} A \in \mathcal{F} & \Longrightarrow & \mu(A) \geq \mu(\varnothing) = 0 \\ A_i \in \mathcal{F} \text{ disjoint} & \Longrightarrow & \mu(\bigcup_i A_i) = \sum_i \mu(A_i) \end{array}$$

Measure space $(D, \mathcal{F}, \mu)$: σ-algebra $(D, \mathcal{F})$, measure $\mu : \mathcal{F} \to \mathbb{R}$

The **model count** of a formula φ is $\mathsf{mc}(\varphi) = \mu(\llbracket\varphi\rrbracket)$.

A logical theory $\mathcal{T}$ is **measured** if every $\llbracket\varphi\rrbracket$ is measurable.

Measured theories: Examples

Theory	**Domain**	**Connectives**	**Quantifiers**	$\mathsf{mc}(\varphi)$
Boolean satisfiability	$\{\mathsf{T}, \mathsf{F}\}$	$\wedge, \vee, \neg$	None	Number of satisfying assignments
Integer arithmetic	$\mathbb{Z} \cap [a, b]$	$\wedge, \vee, \neg$	$\exists$	Number of models
Linear real arithmetic	$\mathbb{R} \cap [a, b]$	$\wedge, \vee, \neg$	$\exists$	Volume

Agenda

Tuesday	computational complexity, probability theory
Wednesday	randomized algorithms, Monte Carlo methods
Thursday	hashing-based approach to model counting
Friday	from discrete to continuous model counting

Outline

1. Model counting for Integer Arithmetic
2. Model counting for Real Arithmetic
   - Hashing-based approach
   - Computing integrals
3. Other approaches and theories
4. Some applications and challenges

Integer Arithmetic (IA)

Syntax

- constant symbols 0 and 1
- function symbols $+, -, \cdot$
- predicate symbol $\leq$
- equality

Semantics is defined in the structure $\langle \mathbb{Z}, +, -, \cdot, \leq \rangle$

Example formulas

$$even(x) \quad : \quad \exists y.\, x = y + y$$

$$\forall x \forall y \forall z.\, x^3 + y^3 = z^3 \rightarrow (x = 0 \vee y = 0 \vee z = 0),$$

where x^3 is a shortcut for $x \cdot x \cdot x$

Integer Arithmetic (IA)

Syntax

- constant symbols 0 and 1
- function symbols $+, -, \cdot$
- predicate symbol $\leq$
- equality

Semantics is defined in the structure $\langle \mathbb{Z}, +, -, \cdot, \leq \rangle$

With multiplication, checking satisfiability is undecidable.

If the variable domains are bounded, then satisfiability is decidable.

Recap: Hashing-based approximate $\#\mathrm{SAT}$

[Jerrum, Valiant, Vazirani 1986]

$\varphi(\boldsymbol{x}) = \varphi(x_1, \ldots, x_n)$ propositional formula
$\mathsf{mc}(\varphi) = ?$

Idea:

1. Take an appropriate hash function $h\colon \{0,1\}^n \to \{0,1\}^m$.
2. Take $\psi(\boldsymbol{x}) = \varphi(\boldsymbol{x}) \wedge (h(\boldsymbol{x}) = 0^m)$.
3. On expectation, $\mathsf{mc}(\psi) = \mathsf{mc}(\varphi)/2^m$.
4. ψ is satisfiable with high probability if $\mathsf{mc}(\varphi) \gg 2^m$.

Hashing approach for $\#\mathbf{P}$ problems

Theorem [Jerrum, Valiant, Vazirani 1986]

$$\text{approximate } \#\mathbf{P} \subseteq \mathbf{BPP}^{\mathbf{NP}}$$

Approximate $\#\mathrm{SMT}$ for Integer Arithmetic

Example:

$\varphi(u,v) = (0 \le u \le 4) \wedge (1 \le v \le 4) \wedge (u - v \ge 0)$

Hash function: $h(\boldsymbol{x}) = A \cdot \boldsymbol{x} + \boldsymbol{b}$, coefficients from $\{0,1\}$ u.a.r.

Queries to SMT solver:

$\varphi(u,v)$	(in integer variables)
$\wedge\, (\boldsymbol{x} = \mathrm{bin}(u,v))$	(binary encoding)
$\wedge\, (A \cdot \boldsymbol{x} + \boldsymbol{b} = 0^m)$	(hashing into m bits)

What changes compared to $\#\mathrm{SAT}$?

What changes compared to $\#\mathrm{SAT}$?

- Auxiliary variables x from binary encoding
- Since we use an SMT solver for IA, the formula φ can be an arbitrary quantifier-free formula in IA
- In fact, existentially quantified φ are also fine (both here and in the propositional case)
- We can also use hash functions based on integers not bits

Summary: Approximate $\#\mathrm{SMT}$ [IA]

Theorem

$\#\mathrm{SMT}$ for bounded integer arithmetic (IA)
can be approximated with a multiplicative error
by a polynomial-time randomized algorithm
that has oracle access to satisfiability of formulas in IA.

Outline

1. Model counting for Integer Arithmetic

2. Model counting for Real Arithmetic
   - Hashing-based approach
   - Computing integrals

3. Other approaches and theories

4. Some applications and challenges

Real Arithmetic (RA)

Syntax

- constant symbols 0 and 1
- function symbols $+, -, \cdot$
- predicate symbol $\leq$
- equality

Semantics is defined in the structure $\langle \mathbb{R}, +, -, \cdot, \leq \rangle$

Example formula

$$\exists x.\, x > 1 \wedge x \cdot x - x - 1 = 0$$

Real Arithmetic (RA)

Syntax

- constant symbols 0 and 1
- function symbols $+, -, \cdot$
- predicate symbol $\leq$
- equality

Semantics is defined in the structure $\langle \mathbb{R}, +, -, \cdot, \leq \rangle$

Linear fragment

- extend the set of constant symbols with the computable reals
- restrict $\cdot$ so that at least one argument is a constant

Model counting for Real Arithmetic

Which model counting procedures for Real Arithmetic
have we already seen?

Model counting for Real Arithmetic

Which model counting procedures for Real Arithmetic
have we already seen?

- Monte Carlo sampling
- Markov chain Monte Carlo (if $\llbracket \varphi \rrbracket$ is convex)

Model counting: From integers to reals

Discretization:

- Partition the domain $[a, b]^n$ into cubes
- Overapproximate the body with the cubes it intersects

Complexity-theoretic point of view:

- Reduce to a $\#\mathbf{P}$ problem

Model counting: From integers to reals

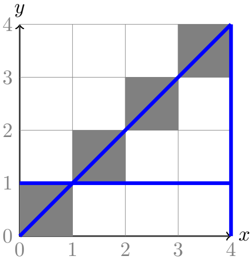
Approximation error: total volume of **cut** cubes

Formula size: log(number of **all** cubes)

Example:

Variables
$x, y \in [0, 4] \subseteq \mathbb{R}$

$$\begin{array}{lcl} x & \leq & 4 \\ y & \geq & 1 \\ x - y & \geq & 0 \end{array}$$



16 cubes

4 cut cubes

Model counting: From integers to reals

Approximation error: total volume of cut cubes

Formula size: log(number of all cubes)

Theorem [Dyer, Frieze 1988]
Approximate volume computation ($\#\mathrm{SMT}$) for polytopes reduces to $\#\mathbf{P}$.

Limitation: applicable only to quantifier-free formulas

RA : Formulas contain existential quantifiers

Model counting for linear real arithmetic

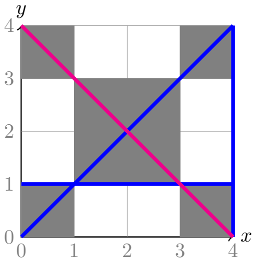
Input: $\varphi(\boldsymbol{x}) = \exists\, \boldsymbol{z}.\, \Phi(\boldsymbol{x}, \boldsymbol{z})$
Output: approximation of mc(φ)

Example:

Variables
$x, y \in [0, 4] \subseteq \mathbb{R}$,
$z \in \mathbb{R}$

$$\begin{array}{lcl} x & \leq & 4 \\ y & \geq & 1 \\ x - y & \geq & 0 \\ x + y - z & \geq & 0 \\ z & \geq & 4 \end{array}$$

Projection on (x, y):



16 cubes

8 cut cubes

Model counting for linear real arithmetic

Input: $\varphi(\boldsymbol{x}) = \exists\, \boldsymbol{z}.\, \Phi(\boldsymbol{x}, \boldsymbol{z})$
Output: approximation of $\mathsf{mc}(\varphi)$

Lemma
Number of cutting hyperplanes is at most 2^l,
where l is the number of atomic predicates in Φ.

Corollary
Number of cubes increases by an exponential factor,
number of bit variables increases by a polynomial.

Summary: Approximate $\#\mathrm{SMT}$ [RA]

Theorem

$\#\mathrm{SMT}$ for linear real arithmetic (RA)
can be approximated with an additive error
by a polynomial-time randomized algorithm
that has oracle access to satisfiability of formulas in IA + RA.

Model counting and computing integrals

A different world:

$$I = \int_a^b f(x)\,dx$$

Now $I = \mu([a, b])$ for a measure that has density f.
Cannot we compute such integrals efficiently?

Numerical integration

Typical theorem:
If $|f''(x)| \leq H$ for all $x \in [a,b]$, then the additive error of the rectangle method is at most $O(H/N^2)$ where N is the number of grid points.

If the dimension n is unbounded, then even small N along each axis leads to an exponential number of cubes.

Outline

SMT = Boolean structure + Theory predicates

[Ma, Liu, Zhang, CADE'09]
[Zhou et al. (2014)]

Quantifier-free integer or real arithmetic:
Boolean structure + Theory predicates

- SAT solver or BDD engine for the Boolean structure
- Model counting oracle for conjunctions of constraints

Integer points in convex polyhedra

[Barvinok, FOCS'93]

Theorem

For every fixed k, there exists a polynomial-time algorithm that computes $\mathsf{mc}(\varphi)$ for ANDs of linear inequalities in the theory of integer arithmetic.

Underlying technique:
Generating functions for polyhedra and cones.

Model counting for strings

[Luu et al., PLDI'14]

Predicates of the logic:

- $s = s_1 \cdot s_2$
- s matches regular expression R
- s contains a fixed string abc
- length$(s_1) \geq$ length(s_2)
- first occurrence of abc in s is at position ≥ 73

Satisfiability undecidable.
Model counting does not provide prior guarantees.

Underlying technique:
Generating functions for sets of strings.

Parametric counting and privacy properties

[Fredrikson, Jha, CSL-LICS'14]

Differential privacy (Dwork et al. '06):
"Neighbouring" inputs should have similar probabilities of producing a particular output.
Counterexamples look like this:

$$(-S < x_1 - x_2 < S) \wedge \frac{\mathsf{count}(r_1, \Phi(x_1, r_1, s), s)}{\mathsf{count}(r_2, \Phi(x_2, r_2, s), s)} > \exp(\epsilon),$$

$$\text{where} \quad \Phi(x, r, s) \equiv ((s = x + r) \wedge (-B < r < B)$$

This corresponds to logics with **parametric** counting.

Decidability for a fragment of such logic.

Model counting for complex data structures

[Filieri, Frias, Pasareanu, Visser, SPIN'15]

Model counting for data structures with numeric fields

- heap constraints ($ref = null$, $ref_1 \neq ref_2$)
- numerical constraints ($in.elem > in.next.elem$)

Combine enumeration and model counting (Barvinok's algorithm):

- enumerate the structures,
- keep the constraints on numeric fields symbolic.

Summary of today's lecture so far

- Hashing-based model counting for integer and real arithmetic
- Discretization and numerical integration
- What is model counting beyond numerical domains

Outline

1. Model counting for Integer Arithmetic
2. Model counting for Real Arithmetic
   - Hashing-based approach
   - Computing integrals
3. Other approaches and theories
4. Some applications and challenges

Theory	Applications	Challenges
Boolean logic	random test generation	efficient reasoning about XOR constraints
Integer arithmetic	probabilistic inference	efficient reasoning about combination of theories and hash functions
Linear real arithmetic	probabilistic inference	improved discretization; MCMC convergence

Theory	**Applications**	**Challenges**
Boolean logic	random test generation	efficient reasoning about XOR constraints
Integer arithmetic	probabilistic inference	efficient reasoning about combination of theories and hash functions
Linear real arithmetic	probabilistic inference	improved discretization; MCMC convergence

Uniform test generation

Manual test generation

- captures testers' knowledge,
- not scalable for large projects.

Random constrained test generation

- uses constraints to capture testers' knowledge,
- uses constraint solvers to find test cases,
- is used in hardware design.

It is desirable to sample uniformly at random from the test cases that satisfy the constraints in an efficient and scalable way.

Uniform test generation

Manual test generation

- captures testers' knowledge,
- not scalable for large projects.

Random constrained test generation

- uses constraints to capture testers' knowledge,
- uses constraint solvers to find test cases,
- is used in hardware design.

Almost uniform test generation based on universal hashing.
Tens (hundreds) of thousands of variables within seconds (minutes).
[Recent papers by Chakraborty, Fremont, Meel, Seshia and Vardi]

Reasoning about XOR constraints

Recall that the hash function constraints are encoded as exclusive-or (XOR) constraints conjuncted with the formula.

XOR constraints are difficult for SAT solvers, and thus remain the big challenge for the scalability of hashing-based approaches.

- SAT solver CryptoMiniSat is specialized for XOR constraints.
- A recent algorithm uses a number of calls to the oracle that is logarithmic in the number of variables in the formula.
[Chakraborty, Meel, and Vardi, IJCAI'16]

Theory	Applications	Challenges
Boolean logic	random test generation	efficient reasoning about XOR constraints
Integer arithmetic	probabilistic inference	efficient reasoning about combination of theories and hash functions
Linear real arithmetic	probabilistic inference	improved discretization; MCMC convergence

Inference for probabilistic programs

Probabilistic programs are a modelling formalism for specifying probability distributions and probabilistic systems.

Combining sampling, model counting and static analysis one can perform inference and establish probabilistic properties.

Examples: medical decision systems and cyber-physical systems.
[Sankaranarayanan, Chakarov, Gulwani, PLDI'13]

Theory	Applications	Challenges
Boolean logic	random test generation	efficient reasoning about XOR constraints
Integer arithmetic	probabilistic inference	efficient reasoning about combination of theories and hash functions
Linear real arithmetic	probabilistic inference	improved discretization; MCMC convergence

Hashing for logical theories

Modern SMT solvers perform reasoning on the theory level, even for formulas over bounded integers or bit-vectors. Their efficiency often depends on making use of the formula's structure.

Hash-function constraints are usually Boolean or contain mod operators, and might cause the solver to resort to bit-blasting.

The development of theory-level families of pairwise-independent hash functions is an important problem that remains a challenge.

[Chakraborty, Meel, Mistry, Vardi, AAAI'16]
[Chistikov, Dimitrova, Majumdar, TACAS'15]

Theory	Applications	Challenges
Boolean logic	random test generation	efficient reasoning about XOR constraints
Integer arithmetic	probabilistic inference	efficient reasoning about combination of theories and hash functions
Linear real arithmetic	probabilistic inference	improved discretization; MCMC convergence

Model counting for continuous domains

- Markov chain Monte Carlo bottleneck: the number of simulation steps before we can start sampling
- Hashing-based method bottleneck: the precision of discretization for achieving approximation guarantees

Theory	Applications	Challenges
Boolean logic	random test generation	efficient reasoning about XOR constraints
Integer arithmetic	probabilistic inference	efficient reasoning about combination of theories and hash functions
Linear real arithmetic	probabilistic inference	improved discretization; MCMC convergence

What we have learned in this course

We have recalled the basics of:

- First-order logic
- Computational complexity
- Probability theory
- Algorithm analysis

What we have learned in this course

Model counting in	**MCMC**	**Universal hashing**
Boolean logic	model counting via uniform sampling	hash functions based on XOR
Integer arithmetic	model counting via uniform sampling	combined integer and Boolean reasoning
Linear real arithmetic	volume estimation via uniform sampling	volume estimation via discretization

Thank you!

chdir@cs.ox.ac.uk
rayna@mpi-sws.org