The Next Challenge: Model Counting Modulo Theories

Lucas Bang
Harvey Mudd College

Workshop on Counting and Sampling
July 6, 2021

Motivation

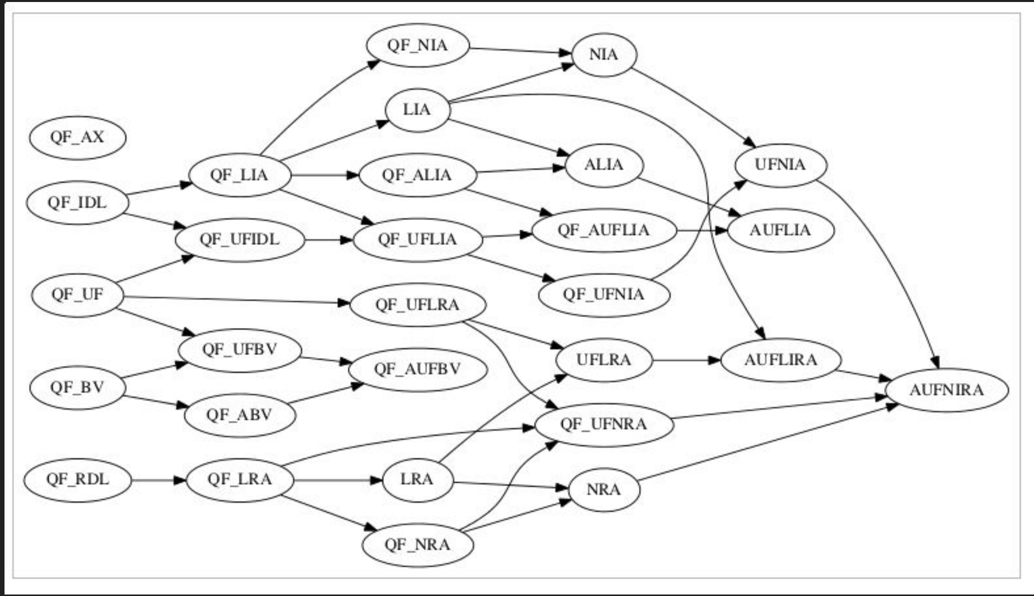
SMT has enabled progress in:

- program verification, automatic test generation, symbolic execution, program synthesis, type inference, motion planning, security exploit detection, ...

SMT covers a rich set of theories:

- Booleans, bitvectors, strings, integers, reals, floats, arrays, uninterpreted functions,...

SMT-LIB
THE SATISFIABILITY MODULO THEORIES LIBRARY



Motivation

There has been **significant** progress in propositional model counting

- Fast and accurate solvers
- Lots of cool theory
- Sampling
- Approximations
- Caching
- Hashing
- Applications in AI, formal methods, etc.

Motivation

I consider myself a consumer of SMT & Model Counting.

My research pipeline in Quantitative Information Flow (QIF):

code → static analysis (SMT) → constraints → model counting → info theory

More than Booleans! Strings, integers, arrays, data structures, pointers, floats, ...

Existing tools mostly propositional.

When I needed to do QIF for other domains, I had to go implement my own model counters!

- Arrays: VSTTE 2020, FSE 2020
- Strings: CAV 2015, FSE 2015
- Strings + Integers: FSE 2018
- Intervals: ATVA 2018

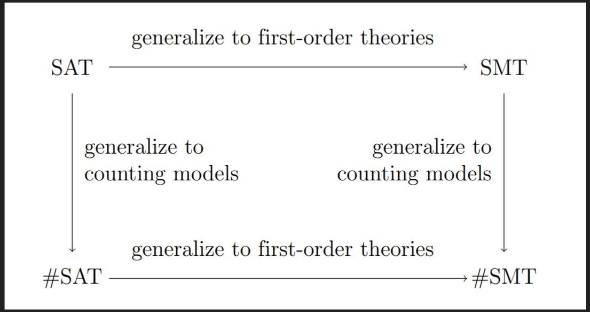
Motivation

To enable quantitative analysis for other very interesting domains, we need a framework and practical tools for

Model Counting Modulo Theories*

This talk:

- some observations on DPLL model counting
- overview of my work in strings and arrays
- a call to action for #SMT.



*Sang Phan's Dissertation, 2015

#SAT

The DPLL Algorithm

Function : DPLL(ϕ)
Input : CNF formula ϕ over n variables
Output : true or false, the satisfiability of F

```
begin
    UnitPropagate(φ)
    if φ has false clause then return false
    if all clauses of φ satisfied then return true
    x ← SelectBranchVariable(φ)
    return DPLL(φ[x ↦ true]) ∨ DPLL(φ[x ↦ false])
end
```

The #DPLL Algorithm

"The Good Old Davis-Putnam Procedure Helps Counting Models". Birnbaum & Lozinskii, JAIR 1999

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** *false*
  if all clauses of ϕ satisfied **then return** *true*
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}]$, $t-1$) $\vee$ DPLL($\phi[x \mapsto \mathit{true}]$, $t-1$)
end

The #DPLL Algorithm

"The Good Old Davis-Putnam Procedure Helps Counting Models". Birnbaum & Lozinskii, JAIR 1999

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** *false*
  if all clauses of ϕ satisfied **then return** *true*
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}], t-1$) $\vee$ DPLL($\phi[x \mapsto \mathit{true}], t-1$)
end

The #DPLL Algorithm

"The Good Old Davis-Putnam Procedure Helps Counting Models". Birnbaum & Lozinskii, JAIR 1999

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** 0
  if all clauses of ϕ satisfied **then return** *true*
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}], t-1$) $\vee$ DPLL($\phi[x \mapsto \mathit{true}], t-1$)
end

The #DPLL Algorithm

"The Good Old Davis-Putnam Procedure Helps Counting Models". Birnbaum & Lozinskii, JAIR 1999

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** 0
  if all clauses of ϕ satisfied **then return** *true*
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}], t - 1$) $\vee$ DPLL($\phi[x \mapsto \mathit{true}], t - 1$)
end

The #DPLL Algorithm

"The Good Old Davis-Putnam Procedure Helps Counting Models". Birnbaum & Lozinskii, JAIR 1999

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
    UnitPropagate(ϕ)
    if ϕ has false clause **then return** 0
    if all clauses of ϕ satisfied **then return** 2^t
    x $\leftarrow$ SelectBranchVariable(ϕ)
    return DPLL($\phi[x \mapsto \mathit{true}], t - 1$) $\vee$ DPLL($\phi[x \mapsto \mathit{true}], t - 1$)
end

The #DPLL Algorithm

“The Good Old Davis-Putnam Procedure Helps Counting Models”. Birnbaum & Lozinskii, JAIR 1999

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** 0
  if all clauses of ϕ satisfied **then return** 2^t
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \textit{true}], t-1$) $+$ DPLL($\phi[x \mapsto \textit{true}], t-1$)
end

The #DPLL Algorithm

"The Good Old Davis-Putnam Procedure Helps Counting Models". Birnbaum & Lozinskii, JAIR 1999

SAT Check → Model Count

Simple transformation

Function : DPLL(ϕ)
Input : CNF formula ϕ over n variables
Output : true or false, the satisfiability of F
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** false
  if all clauses of ϕ satisfied **then return** true
  x ← SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}]$) $\vee$ DPLL($\phi[x \mapsto \mathit{false}]$)
end

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** 0
  if all clauses of ϕ satisfied **then return** 2^t
  x ← SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}], t-1$) + DPLL($\phi[x \mapsto \mathit{true}], t-1$)
end

#Strings

Tons of model counting work for strings!

- “Parameterized model counting for string and numeric constraints”. Aydin, Eiers, Bang, Brennan, Gavrilov, Bultan, Yu. FSE 2018
- “Model Counting for Recursively-Defined Strings.” Minh-Thai Trinh, Duc-Hiep Chu, Joxan Jaffar: CAV 2017
- “Automata-Based Model Counting for String Constraints.” Aydin, Bang, Bultan CAV 2015
- “A model counter for constraints over unbounded strings”. Luu, Shinde, Saxena, Demsky. PLDI 2014
- “Accurate String Constraints Solution Counting with Weighted Automata”. Sherman & Harris. ASE 2019

Automata-Based Counter (ABC), CAV 2015

$$X \in (0|(1(01^*0)^*1))^*$$

Q: How many solutions for X? A: Infinitely many!

Q: How many solutions for X of length k?

A counting sequence for language $\mathcal{L}$ encodes

$$a_k = |\{s : s \in \mathcal{L}, \text{len}(s) = k\}|$$

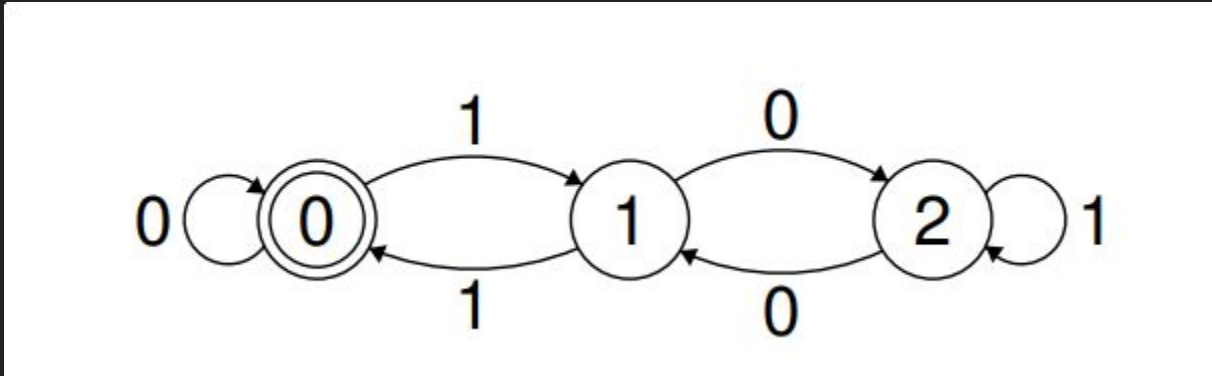
$$a_0 = 1, a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 3, a_5 = 5, \ldots$$

k	X	a_k
0	ε	1
1	0	1
2	11	1
3	110	1
4	1001, 1100, 1111	3
5	10010, 10101, 11000, 11011, 11110	5

Automata-Based Counter (ABC), CAV 2015

Q: How to CHECK if a string satisfies a regex?
A: McNaughton–Yamada–Thompson Algorithm

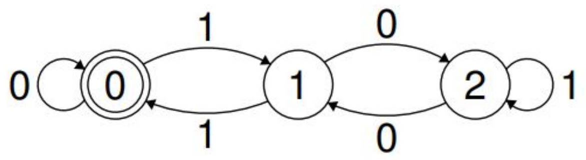
$$X \in (0|(1(01^*0)^*1))^*$$



Automata-Based Counter (ABC), CAV 2015

Q: How to **COUNT** strings satisfying a regex?

1)



2)

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

3)

$$g(z) = \frac{\det(I - zA : i, j)}{(-1)^n \det(I - zA)}$$

4)

$$g(z) = \frac{1 - z - z^2}{(z - 1)(2z^2 + z - 1)}$$

5)

$$g(z) = 1z^0 + 1z^1 + 1z^2 + 1z^3 + 3z^4 + 5z^5 + \dots$$

Automata-Based Counter (ABC), CAV 2015

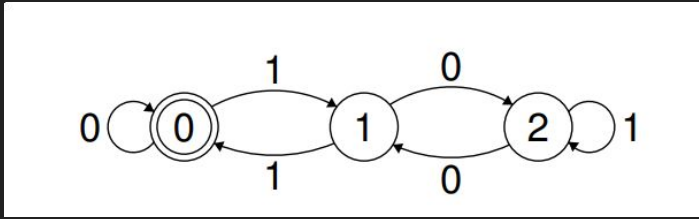
$$
\begin{array}{lll}
\varphi & \longrightarrow & \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi_{\mathbb{Z}} \mid \varphi_{\mathbb{S}} \mid \top \mid \bot \\
\varphi_{\mathbb{Z}} & \longrightarrow & \beta = \beta \mid \beta < \beta \mid \beta > \beta \\
\varphi_{\mathbb{S}} & \longrightarrow & \gamma = \gamma \mid \gamma < \gamma \mid \gamma > \gamma \mid \mathbf{match}(\gamma, \rho) \mid \mathbf{contains}(\gamma, \gamma) \\
 & \mid & \mathbf{begins}(\gamma, \gamma) \mid \mathbf{ends}(\gamma, \gamma) \\
\beta & \longrightarrow & v_i \mid n \mid \beta + \beta \mid \beta - \beta \mid \beta \times n \\
 & \mid & \mathbf{length}(\gamma) \mid \mathbf{toint}(\gamma) \mid \mathbf{indexof}(\gamma, \gamma) \mid \mathbf{lastindexof}(\gamma, \gamma) \\
\gamma & \longrightarrow & v_s \mid \rho \mid \gamma \cdot \gamma \mid \mathbf{reverse}(\gamma) \mid \mathbf{tostring}(\beta) \mid \mathbf{charat}(\gamma, \beta) \mid \\
 & \mid & \mathbf{substring}(\gamma, \beta, \beta) \mid \mathbf{replacefirst}(\gamma, \gamma, \gamma) \mid \mathbf{replacelast}(\gamma, \gamma, \gamma) \\
 & \mid & \mathbf{replaceall}(\gamma, \gamma, \gamma) \\
\rho & \longrightarrow & \varepsilon \mid s \mid \rho \cdot \rho \mid \rho \,|\, \rho \mid \rho^*
\end{array}
$$

Automata-Based Counter (ABC), CAV 2015

Containment Check → Model Count

Simple transformation

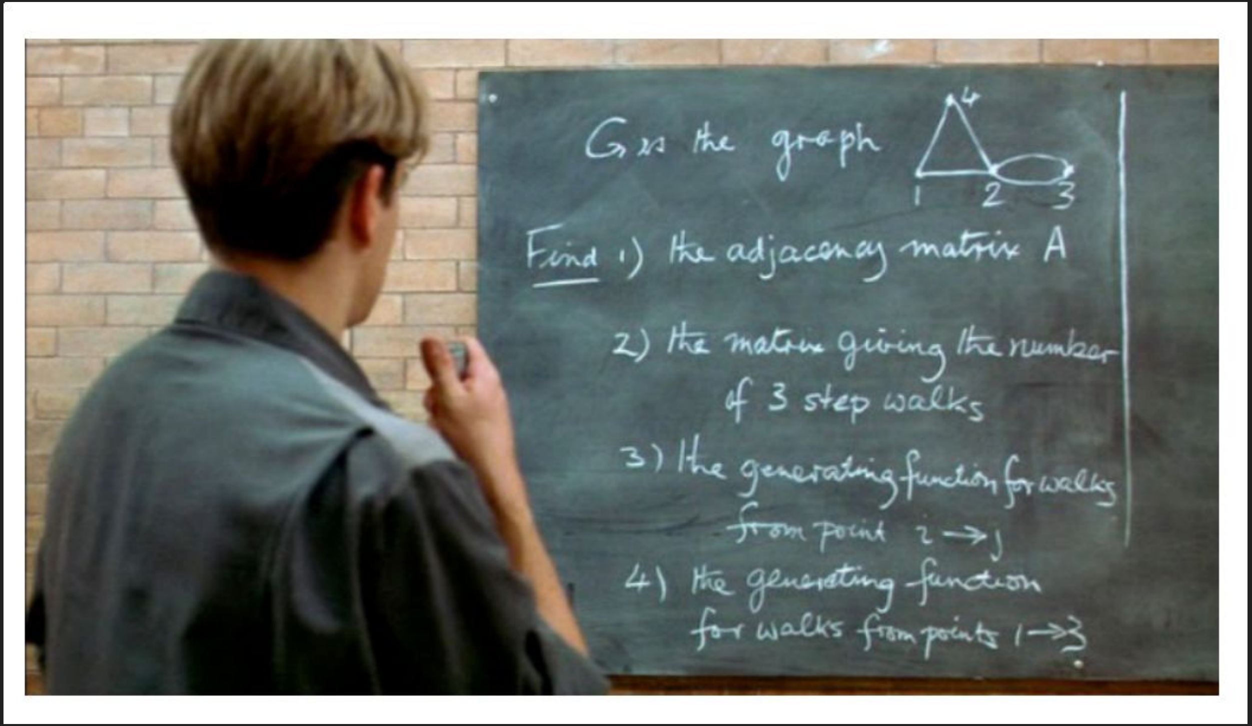
$$X \in (0|(1(01^*0)^*1))^*$$



$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

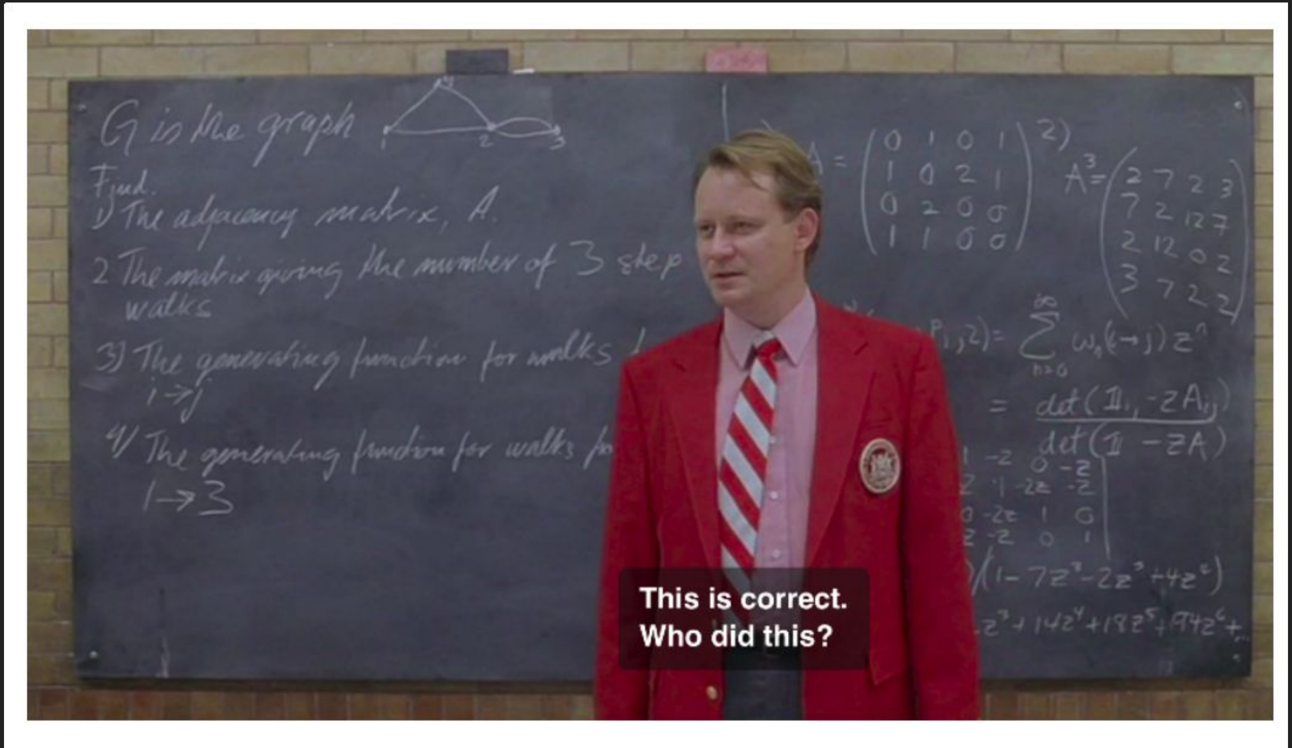
$$g(z) = \frac{\det(I - zA : i, j)}{(-1)^n \det(I - zA)}$$

Automata-Based Counter (ABC), CAV 2015



Good Will Hunting, 1997

Automata-Based Counter (ABC), CAV 2015

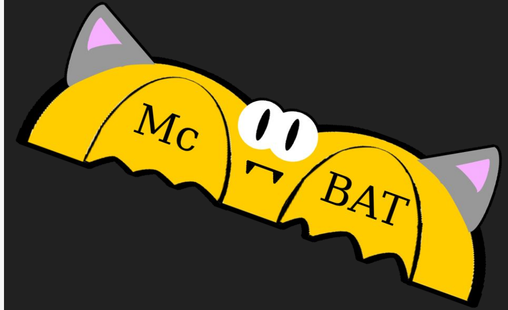


Good Will Hunting, 1997

#Arrays

Arrays: Model Counting Bounded Array Theory (MCBAT)

$$
\begin{aligned}
\textit{formula} : \quad & \textit{formula} \wedge \textit{formula} \\
& \mid \textit{formula} \vee \textit{formula} \\
& \mid \textit{formula} \rightarrow \textit{formula} \\
& \mid \neg \textit{formula} \\
& \mid \forall(\textit{int-id}).(\textit{formula}) \\
& \mid \textsc{length}(\textit{array-id},\ \mathbb{Z}^{\geq}) \\
& \mid \textit{atom} \\
\textit{atom} : \quad & \textit{term} = \textit{term} \mid \textit{term} < \textit{term} \mid \textit{array} = \textit{array} \\
\textit{array} : \quad & \textit{array-id} \mid \textit{array}\{\textit{term} \leftarrow \textit{term}\} \\
\textit{term} : \quad & \textit{int-id} \mid \mathbb{Z} \mid \mathbb{Z} \times \textit{term} \mid \textit{term} + \textit{term} \mid \textit{array}[\textit{term}]
\end{aligned}
$$



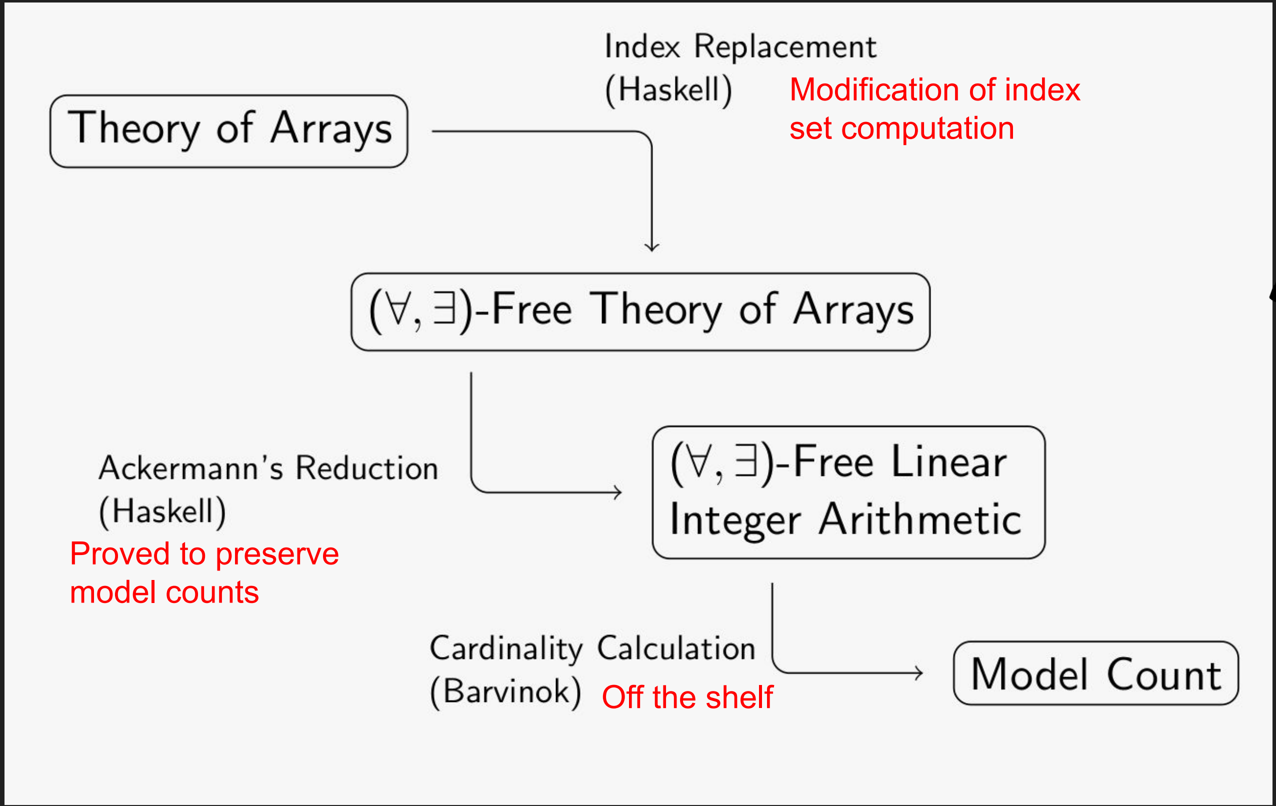
Arrays: Key Observations in MCBAT

- Our (SAT procedure) inspirations
  - "What's Decidable About Arrays?"
    Aaron R. Bradley, Manna, & Sipma, VMCAI 2006
  - "Decision Procedures: an Algorithmic Point of View"
    Kroening & Strichman, Springer 2008

- The sticking points for model counting:
  - **Index Set**: set of all indices that might be used in array formula, used in SAT-preserving quantifier elimination transformation
  - **Ackermann Reduction** (Uninterpreted Functions → Equality Logic)

Both help in making transformations to preserve satisfiability, but neither (known) to preserve counts.

Arrays: Model Counting Bounded Array Theory (MCBAT)



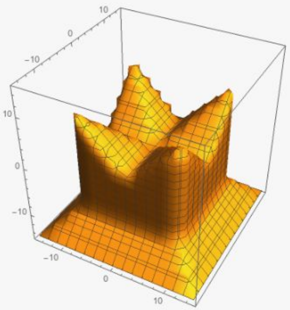
Arrays: Model Counting Bounded Array Theory (MCBAT)

$$\text{LENGTH}(a, 2) \wedge (k \geq -15) \wedge \forall(i).(k \leq a[i] \leq 10 \vee k \leq -a[i] \leq 10)$$

$$\Downarrow$$

$$\begin{aligned} &(k \geq -15) \\ \wedge&(k \leq a_0 \leq 10 \vee -k \leq a_0 \leq -10) \\ \wedge&(k \leq a_1 \leq 10 \vee -k \leq a_1 \leq -10) \end{aligned}$$

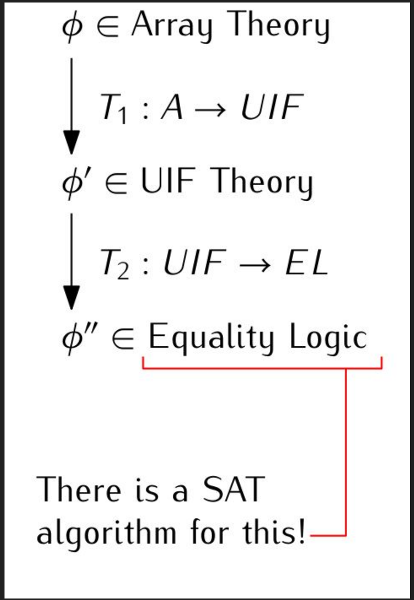
$$\Downarrow$$



Count: 10076

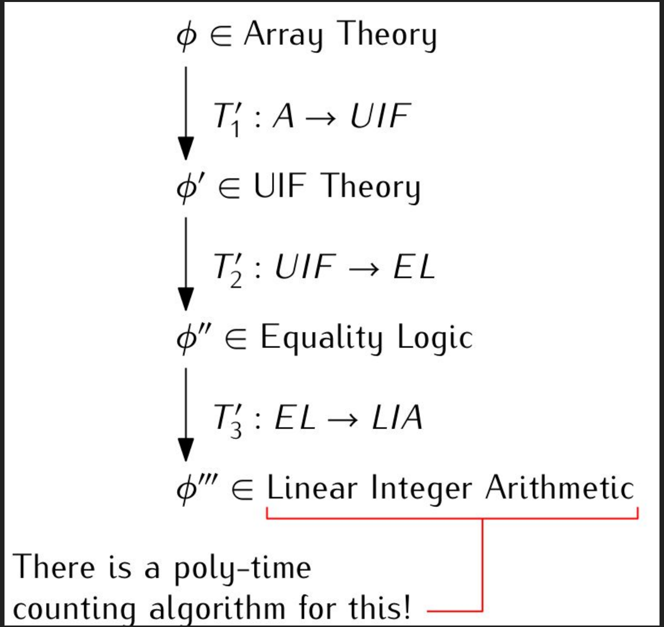
Arrays: Model Counting Bounded Array Theory (MCBAT)

Array Decision Procedure



Simple transformation

Array Model Count



Challenges

We need benchmarks

There are lots of #SAT benchmarks! Yay!

Plenty of string counting benchmarks are available

- Kaluza (original benchmark PLDI 2014), Kausler, SMC, symbolic execution benchmarks… see earlier references on string model counters

Array counting benchmarks are rare :(

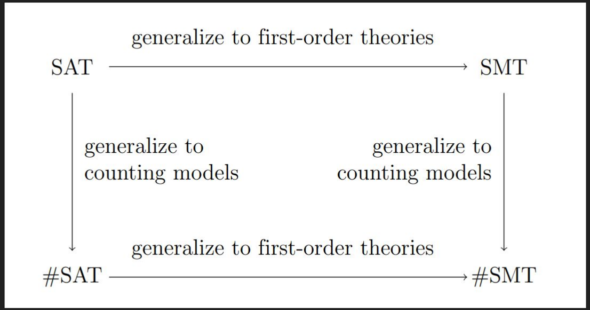
- MCBAT VSTTE 2020 generated an available benchmark from
    - loop invariants for array-manipulating programs
    - symbolic execution of common array-based algorithms (sorting, searching, etc.)
    - verified counts by comparing MCBAT and Z3-based enumeration

Competitions for other theories? 2022? :)

We need Model Counting Modulo Theories (MCMT, #SMT)

We now have model counting algorithms and tools for

- Propositional logic
- Strings
- Arrays
- Integers



Two challenges to the model counting community

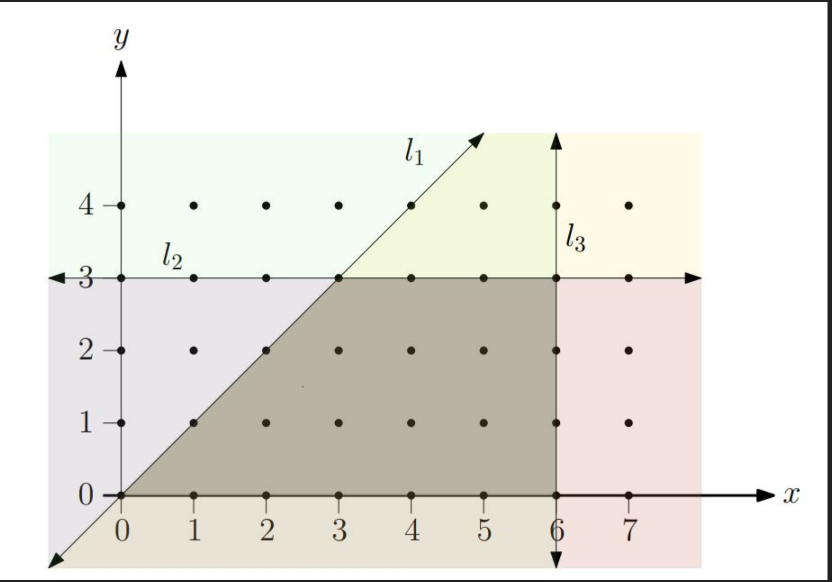
1. Develop a generic framework for model counting combinations of theories
2. Model counting for more theories!
    - Data types
    - Recursive Data Structures
    - Pointer Logic
    - More expressive theories of strings, arrays, integers
    - Apply hashing, caching, approximations to other theories

Go forth and #SMT!

Integers

LattE: model counting for linear integer arithmetic

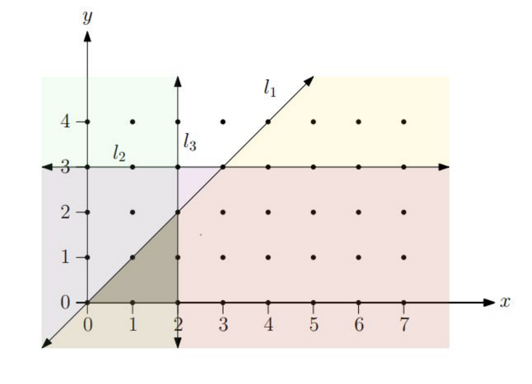
$$x \geq 0 \ \land \ y \geq 0 \ \land \ y \leq x \ \land \ 2y \leq 6 \ \land \ x \leq 6$$

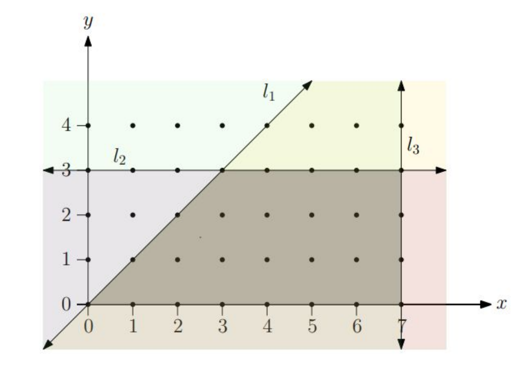


22 solutions

Barvinok: model counting for linear integer arithmetic

$$x \geq 0 \ \wedge \ y \geq 0 \ \wedge \ y \leq x \ \wedge \ 2y \leq 6 \ \wedge \ x \leq t$$





$$f(t) = \begin{cases} \frac{1}{2}t^2 + t + 1 & 0 \leq t \leq 3 \\ 10 + 4t & 0 \leq t > 3 \\ 0 & \text{otherwise} \end{cases}$$