What Should #SMT Count?

Giuseppe Spallitta

International Workshop on Counting and Sampling
25th July, 2026

RICE

The question

In propositional logic, model counting has a canonical meaning:

$$\#F = |\mu : \mu \models F|. \qquad \text{For example, if } F = (x_1 \vee x_2), \text{ then } \#F = 3.$$

SMT extends propositional logic with atoms interpreted in a background theory, such as

$$x > 0, \qquad x + y \leq 1, \qquad f(a) = b.$$

But then the notation "#SMT" becomes less immediate:

Main question

What does #SMT mean?

Propositional baseline: #SAT is unambiguous

Example formula

$$F = (a \vee b) \wedge (\neg a \vee c)$$

A model is a total Boolean assignment:

$$\mu : \{a, b, c\} \to \{\bot, \top\}.$$

Counting semantics

$$\#F = |\{\mu : \mu \models F\}|.$$

The domain is finite, discrete, and fixed by the variables of the formula.

SMT breaks this simplicity

A very small SMT formula over linear real arithmetic

$$\varphi = (x > 0) \wedge (x < 1)$$

What is its number of models?

Theory-level answer

There are infinitely many real values:

$$x \in (0, 1).$$

If we count the length, it will be:

$$1.$$

Predicate-space answer

There is one Boolean abstraction assignment:

$$p_{x>0} = \top, \qquad p_{x<1} = \top.$$

So the predicate-space count is:

$$1.$$

Message

In both views, the answer can be 1, but this 1 has a different meaning.

#SMT: quantifying the theory solution space

#SMT keeps the semantics of the background theory and quantifies the corresponding solution space directly.

Theory-level view

The SMT formula defines a set of theory-consistent solutions, and its size is evaluated using the natural notion associated with the domain.

No additional weight function is introduced: the solution space is quantified using the underlying unweighted measure.

#SMT **preserves and quantifies the theory-level solution space.**

#SMT: a geometric example

Formula over two real variables

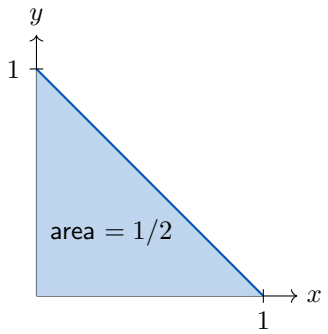
$$\varphi = (0 \leq x \leq 1) \wedge (0 \leq y \leq 1) \wedge (x + y \leq 1).$$

The theory-level solution space is a triangle:

$$\{(x, y) \in [0, 1]^2 : x + y \leq 1\}.$$

With the standard area measure:

$$\#SMT_{\mathrm{LRA}}(\varphi) = \frac{1}{2}.$$



#SMT: a *discrete* geometric example

Formula over two integer variables

$$\varphi = (0 \le x \le 1) \wedge (0 \le y \le 1) \wedge (x + y \le 1), \qquad x, y \in \mathbb{Z}.$$

The theory-level solution space is now the set of integer points:

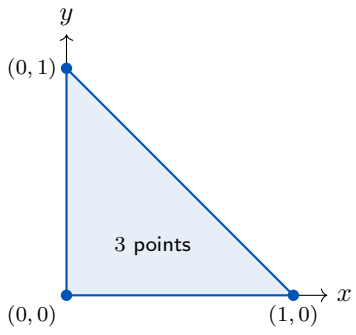
$$\{(x, y) \in \{0, 1\}^2 : x + y \le 1\}.$$

The satisfying assignments are:

$$(0, 0),\ (1, 0),\ (0, 1).$$

Hence:

$$\#SMT_{\mathrm{LIA}}(\varphi) = 3.$$



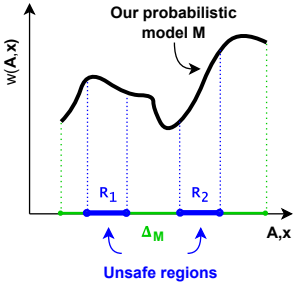
But this is not the only point of view of #SMT...

WMI: hybrid probabilistic inference

Weighted Model Integration (WMI)

- Continuous variables $\mathbf{x} \in \mathbb{R}^N$ and Booleans $\mathbf{A} \in \{0,1\}^M$.
- SMT($\mathcal{LRA}$) constraint $\varphi(\mathbf{x}, \mathbf{A})$ and non-negative weight $w(\mathbf{x}, \mathbf{A})$.
- **WMI** integrates w over all models of φ:

$$\mathrm{WMI}(\varphi, w) = \sum_{\mu_{\mathbf{A}}} \int_{\mathbf{x} \models \varphi(\mathbf{x}, \mu_{\mathbf{A}})} w(\mathbf{x}, \mu_{\mathbf{A}})\, d\mathbf{x}.$$



Goal: hybrid probabilistic inference

$$\Pr(R_1 \vee R_2 \mid M) = \frac{\mathrm{WMI}(\Delta_M \wedge (R_1 \vee R_2), w)}{\mathrm{WMI}(\Delta_M, w)}$$

WMI: weighted volume computation

An SMT formula defines a feasible region: $R_\varphi = \{\mathbf{x} \in \mathbb{R}^N : \mathbf{x} \models_\mathcal{T} \varphi\}$.

Volume computation treats every point in this region equally:

$$\mathrm{Vol}(R_\varphi) = \int_{R_\varphi} 1 \, d\mathbf{x}.$$

Weighted Model Integration additionally considers a non-negative integrable weight function

$$w(\mathbf{x}, \mathbf{A}) : \mathbb{R}^N \times \mathbb{B}^M \longrightarrow \mathbb{R}_{\geq 0}.$$

which determines how much each point contributes.

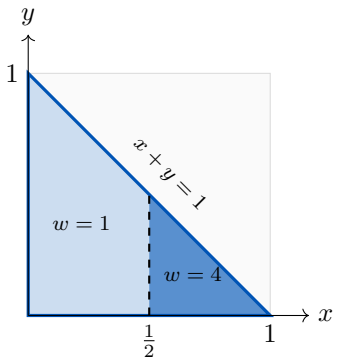
Weighted Model Integration

$$\mathrm{WMI}(\varphi, w) = \int_{R_\varphi} w(\mathbf{x}) \, d\mathbf{x}.$$

WMI computes the weighted mass of its solution space.

WMI: selecting and weighting a region

$$\varphi = (0 \le x \le 1) \wedge (0 \le y \le 1) \wedge (x + y \le 1), \qquad w(x,y) = \begin{cases} 1 & x \le \frac{1}{2}, \\ 4 & x > \frac{1}{2}. \end{cases}$$



The SMT formula determines which points contribute:

$$R_\varphi = \{(x,y) \in [0,1]^2 : x + y \le 1\}.$$

The weight function determines how much each selected point contributes.

Weighted mass

$$\begin{aligned} \mathrm{WMI}(\varphi, w) &= \int_{R_\varphi} w(x,y)\, dx\, dy \\ &= \frac{7}{8}. \end{aligned}$$

WMI is only the tip of the iceberg

The idea behind WMI is more general than integration over piecewise-polynomial weights.

More general weight functions

Nothing prevents us from assigning different constant weights to different regions, exactly as literals or assignments receive weights in WMC:

$$w(x) = \begin{cases} 2 & x \leq 0, \\ 5 & x > 0. \end{cases}$$

The formula selects the feasible models, while the weight function assigns a value to each of them.

Other theories

The same principle can be applied beyond real arithmetic.
For example, over LIA we may compute

$$\sum_{\mathbf{x} \in \mathbb{Z}^n,\, \mathbf{x} \models \varphi} w(\mathbf{x}),$$

thus counting integer solutions while weighting them according to application-specific rules.

WMI is one instance of a broader idea: weighted aggregation of models modulo theories.

Two ways to quantify SMT solutions

Both approaches reason directly about the solutions induced by the background theory, but they aggregate them differently.

#SMT

Quantifies the theory-level solution space by treating all solutions uniformly.

The background theory determines how the resulting quantity is computed, but every theory-level solution is assigned the same weight.

WMI... or w#SMT

Associates a weight or density with the theory-level solutions and aggregates their weighted contribution.

This allows different solutions or regions to contribute differently to the final result.

Common perspective

Both quantify the concrete theory-level solutions of the SMT formula.

But we have another #SMT alternative...

Weak #SMT: counting theory-consistent assignments

Instead of measuring the volume of the theory solution space, weak #SMT counts the Boolean assignments to the theory atoms that both satisfy the formula and are consistent with the theory.

Let

$$\mathsf{At}(\varphi) = \{A_1, \ldots, A_n\}.$$

Each Boolean assignment

$$\alpha : \mathsf{At}(\varphi) \to \{\bot, \top\}$$

represents one possible truth assignment to the theory atoms.

Weak #SMT

$$\#SMT_{\mathcal{T}}^{\text{weak}}(\varphi) = \left|\{\alpha \in \{\bot, \top\}^n \mid \alpha \models \mathsf{Bool}(\varphi) \text{ and } \alpha \text{ is } \mathcal{T}\text{-consistent}\}\right|.$$

The result is always an integer between 0 and 2^n.

Weak #SMT is not simply #SAT

Consider the LRA formula

$$\varphi = (x < 0) \vee (x > 1).$$

Its Boolean abstraction is

$$\text{Bool}(\varphi) = A_1 \vee A_2, \qquad A_1 : x < 0, \quad A_2 : x > 1.$$

Boolean counting

The abstraction has three satisfying assignments:

$$(\top, \bot), \qquad (\bot, \top), \qquad (\top, \top).$$

Therefore, $\#SAT(\text{Bool}(\varphi)) = 3$.

Theory consistency

The assignment $A_1 = \top,\ A_2 = \top$ means

$$x < 0 \wedge x > 1$$

which is inconsistent in LRA.
Therefore, $\#SMT^{\text{weak}}_{\text{LRA}}(\varphi) = 2$.

Weak #SMT counts satisfying Boolean assignments only when they are also theory-consistent.

A second weak-counting example

Formula

$$\varphi = (x \leq 0) \wedge \big((x \geq 1) \vee (x \leq 2)\big).$$

$$A_1 : (x \leq 0), \qquad A_2 : (x \geq 1), \qquad A_3 : (x \leq 2).$$

A_1	A_2	A_3	Bool(φ)?	$\mathcal{T}$-consistent?
$\top$	$\top$	$\bot$	yes	no
$\top$	$\bot$	$\top$	yes	yes
$\top$	$\top$	$\top$	yes	no

Weak count

$$\#SMT^{\text{weak}}_{\text{LRA}}(\varphi) = 1.$$

#SMT: quantifying the theory solution space

#SMT keeps the semantics of the background theory and quantifies the corresponding solution space directly.

Theory-level view

The SMT formula defines a set of theory-consistent solutions, and its size is evaluated using the natural notion associated with the domain.

For example:

- over integers, this means counting satisfying integer points;
- over reals, this means measuring the volume of the satisfying region.

No additional weight function is introduced: the solution space is quantified using the underlying unweighted measure.

#SMT preserves and quantifies the theory-level solution space.

Strong #SMT: quantifying the theory solution space

Strong #SMT keeps the semantics of the background theory and quantifies the corresponding solution space directly.

Theory-level view

The SMT formula defines a set of theory-consistent solutions, and its size is evaluated using the natural notion associated with the domain.

For example:

- over integers, this means counting satisfying integer points;
- over reals, this means measuring the volume of the satisfying region.

No additional weight function is introduced: the solution space is quantified using the underlying unweighted measure.

Strong #SMT preserves and quantifies the theory-level solution space.

Why should we care about the weak view?

Many SMT applications do not need the volume of the real solution space. They need to reason about the feasible combinations of predicates.

Examples

- AllSMT and enumeration of theory-consistent Boolean assignments;
- path feasibility in symbolic execution;
- predicate abstraction and coverage over atoms;
- theory-aware explanations, implicants, and unsat cores;
- knowledge compilation modulo theories.

Message

Weak #SMT (or better, $\mathcal{T}$-**#SAT**) exposes the combinatorial structure of an SMT formula.

$\mathcal{T}$-#SAT and AllSMT are closely related

$\mathcal{T}$-#SAT considers Boolean assignments over the theory atoms that are also consistent with the background theory.

AllSMT

Enumerates the theory-consistent Boolean assignments satisfying the formula.

$$\alpha_1, \alpha_2, \ldots, \alpha_k$$

Return every feasible assignment.

$\mathcal{T}$-#SAT

Counts the theory-consistent Boolean assignments satisfying the formula.

$$\mathcal{T} - \#SAT(\varphi) = k$$

Return how many assignments exist.

Same underlying space

AllSMT enumerates the objects that $\mathcal{T}$-#SAT counts.

Theory lemmas "repair" the Boolean abstraction

When the SMT solver detects that

$$A_2 \wedge A_3$$

is inconsistent in LRA, it can derive the theory-valid clause

$$\neg A_2 \vee \neg A_3.$$

Adding this lemma to the Boolean abstraction removes both impossible assignments:

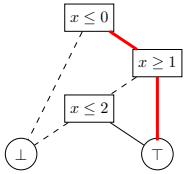
$$\mathsf{Bool}(\varphi) \quad \longrightarrow \quad \mathsf{Bool}(\varphi) \wedge (\neg A_2 \vee \neg A_3).$$

A #SAT approach for $\mathcal{T}$-#SAT

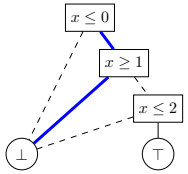
What about SMT Knowledge compilation via Boolean abstraction?

- Build a DD for $\mathcal{T}2\mathcal{B}(\varphi)$, then refine atoms back to theory atoms.
- This can produce **T-inconsistent paths** (branches infeasible in the theory).

$\varphi = (x \leq 0) \wedge ((x \geq 1) \vee (x \leq 2))$



Before: DD with T-inconsistent path



After: T-lemma removes infeasible branch

Key idea

Use **T-lemmas** (T-valid clauses) to rule out all T-inconsistent Boolean assignments *before* Boolean DD compilation.

A simple framework for T-DDs

Framework

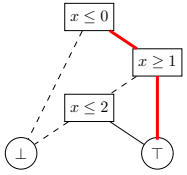
Given an SMT formula φ:

1. Run **AllSMT** to:
   - enumerate all T-consistent assignments to the atoms,
   - collect a set $\mathcal{C}$ of **T-lemmas** ruling out all T-inconsistent assignments.
2. Compile **any Boolean DD** for
$$\mathcal{T}2\mathcal{B}(\varphi \wedge \bigwedge_{C \in \mathcal{C}} C).$$
3. Refine this DD back to theory atoms to obtain a **T-DD**.

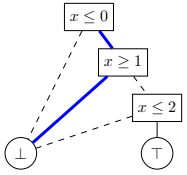
Example: checking $\mathcal{T}$-$\#SAT$ in $\mathcal{T}$-OBBDS

Assuming each path from the root to the node $\top$ contributes to 2^n solutions, with n the number of $\mathcal{T}$-atoms not assigned, then the counts for the following are:

$\varphi = (x \leq 0) \wedge ((x \geq 1) \vee (x \leq 2))$



$1 + 2^1 = 3$



1

Recent progress on weak #SMT

Our recent SAT'26 paper investigated **d-DNNFs modulo theories**[1].

Takeaway from SAT'26

A novel and general technique to leverage d-DNNF compilation and querying at the SMT level:

- theory-agnostic;
- applicable to all forms of d-DNNF;
- preserving polynomial-time SMT queries after compilation.

A central step is the generation of the T-lemmas required to remove the theory-inconsistent Boolean assignments before compilation.

Next step

Our IJCAR'26 paper investigates efficient theory-agnostic approach to T-**lemma enumeration**[2].

[1] G. Masina, E. Civini, M. Michelutti, G. Spallitta, and R. Sebastiani (2026). *d-DNNF Modulo Theories: A General Framework for Polytime SMT Queries.* SAT26: 2603.09975 (cs.LO)

[2] E. Civini, G. Masina, G. Spallitta, and R. Sebastiani (2026). *Beyond Eager Encodings: A Theory-Agnostic Approach to Theory-Lemma Enumeration in SMT.* IJCAR26: 2602.14634 (cs.LO)

Summary: three views of #SMT

The notation "#SMT" can refer to different quantitative questions.

This...	**Should be...**	**What is quantified?**	**Main intuition**
Strong $\#\mathrm{SMT}$	$\#\mathrm{SMT}$	theory-level solution space	unweighted measure
WMI	$w\#\mathrm{SMT}$	weighted theory-level solution space	weighted mass
Weak $\#\mathrm{SMT}$	$\mathcal{T} - \#\mathrm{SAT}$	theory-consistent Boolean assignments	feasible Boolean structure

Thank you for listening!