Towards Practical First-Order Model Counting

Ananth K. Kidambi[1] Guramrit Singh[1] **Paulius Dilkas**[2,3]
Kuldeep S. Meel[4,2]

[1]IIT Bombay, India

[2]University of Toronto, Canada

[3]Vector Institute, Canada

[4]Georgia Tech, USA

SAT 2025

Motivation

Example Setting

- Let Δ be a set of cardinality n
- Suppose we want to count all $P \subseteq \Delta^2$ (as a function of n) that are:
  - functions,
  - bijections,
  - partial orders,
  - symmetric,
  - transitive,
  - etc.

Motivation

Example Setting

- Let Δ be a set of cardinality n
- Suppose we want to count all $P \subseteq \Delta^2$ (as a function of n) that are:
  - functions,
  - bijections,
  - partial orders,
  - symmetric,
  - transitive,
  - etc.

- Propositional model counting (#SAT) is #P-complete
- But many of these counting problems have efficient solutions
- And we can find them using first-order model counting
  - i.e., reasoning about sets, subsets, and arbitrary elements without grounding them

More Formally: What Is the Input?

Example Input Sentence

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \Rightarrow y = z$$

Many-Sorted Function-Free First-Order Logic with Equality

More Formally: What Is the Input?

Example Input Sentence

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \Rightarrow y = z$$

Many-Sorted Function-Free First-Order Logic with Equality

More Formally: What Is the Input?

Example Input Sentence

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \wedge P(x, z) \Rightarrow y = z$$

Many-Sorted Function-Free First-Order Logic with Equality

More Formally: What Is the Input?

Example Input Sentence

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \Rightarrow y = z$$

Many-Sorted Function-Free First-Order Logic with Equality

More Formally: What Is the Input?

Example Input Sentence

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \Rightarrow y = z$$

Many-Sorted Function-Free First-Order Logic with Equality

- Any number of variables and constants

More Formally: What Is the Input?

Example Input Sentence

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \Rightarrow y = z$$

Many-Sorted Function-Free First-Order Logic with Equality

- Any number of variables and constants
- $\exists$ and $\forall$ quantifiers can be nested arbitrarily deeply
- All domains are finite
  - Solutions are functions that take domain sizes as input

More Formally: What Is the Input?

Example Input Sentence

$$\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \Rightarrow y = z$$

Many-Sorted Function-Free First-Order Logic with Equality

- Any number of variables and constants
- $\exists$ and $\forall$ quantifiers can be nested arbitrarily deeply
- All domains are finite
  - Solutions are functions that take domain sizes as input

First-Order Model Counting (FOMC)

- Each predicate acts like a subset
  - of a domain or product of domains
- Goal: count combinations of subsets that satisfy the sentence

Previous Work: Crane (Dilkas and Belle 2023)

- A knowledge compilation approach:
  - Sentences $\rightarrow$ labelled digraphs $\rightarrow$ function-defining equations
- Capable of constructing recursive solutions
- Two variants: greedy search and breadth-first search (BFS)

Previous Work: CRANE (Dilkas and Belle 2023)

- A knowledge compilation approach:
  - Sentences → labelled digraphs → function-defining equations
- Capable of constructing recursive solutions
- Two variants: greedy search and breadth-first search (BFS)

An Example Solution for Counting Bijections

$$f(m, n) = \sum_{l=0}^{n} \binom{n}{l} (-1)^{n-l} g(l, m),$$
$$g(l, m) = g(l-1, m) + mg(l-1, m-1)$$

Previous Work: CRANE (Dilkas and Belle 2023)

- A knowledge compilation approach:
  - Sentences $\rightarrow$ labelled digraphs $\rightarrow$ function-defining equations
- Capable of constructing recursive solutions
- Two variants: greedy search and breadth-first search (BFS)

An Example Solution for Counting Bijections

$$f(m,n) = \sum_{l=0}^{n} \binom{n}{l} (-1)^{n-l} g(l,m),$$
$$g(l,m) = g(l-1,m) + mg(l-1,m-1)$$

Issues We Are Going to Address

Completeness: recursive functions (like g) have no base cases

Usability: how do I compute, e.g., $f(7,7)$?

Workflow (1/2)

1. Use CRANE to compile ϕ into a set of equations $\mathcal{E}$

Workflow (1/2)

1. Use Crane to compile ϕ into a set of equations $\mathcal{E}$
2. Simplify them, e.g.,

$$g(l, m) = \sum_{k=0}^{m} [0 \le k \le 1] \binom{m}{k} g(l-1, m-k)$$

becomes

$$g(l, m) = g(l-1, m) + mg(l-1, m-1)$$

Workflow (1/2)

1. Use Crane to compile ϕ into a set of equations $\mathcal{E}$
2. Simplify them, e.g.,

$$g(l, m) = \sum_{k=0}^{m} [0 \le k \le 1] \binom{m}{k} g(l-1, m-k)$$

becomes

$$g(l, m) = g(l-1, m) + mg(l-1, m-1)$$

3. ($\Rightarrow$) Identify a sufficient set of base cases
   - e.g., $\{ g(0, m), g(l, 0) \}$

Workflow (2/2)

4. For each base case:

$g(0, m)$ $g(l, 0)$

Workflow (2/2)

4. For each base case:

$$g(0, m) \quad\quad\quad\quad g(l, 0)$$

4.1. ($\Rightarrow$) Construct the corr. sentence

$$\forall y \in \Delta.\ S(y) \vee \neg S(y) \quad\quad\quad\quad \top$$

Workflow (2/2)

4. For each base case:

4.1. ($\Rightarrow$) Construct the corr. sentence

4.2. Recurse

$$\begin{tikzcd}
g(0, m) \arrow[d] & g(l, 0) \arrow[d] \\
\forall y \in \Delta.\ S(y) \vee \neg S(y) \arrow[d] & \top \arrow[d] \\
g(0, m) = ??? & g(l, 0) = ???
\end{tikzcd}$$

Workflow (2/2)

4. For each base case:

4.1. ($\Rightarrow$) Construct the corr. sentence

4.2. Recurse

$$\begin{tikzcd}
g(0, m) \arrow[d] & & g(l, 0) \arrow[d] \\
\forall y \in \Delta.\ S(y) \lor \neg S(y) \arrow[d] & & \top \arrow[d] \\
g(0, m) = 0^m & & g(l, 0) = 1
\end{tikzcd}$$

Workflow (2/2)

4. For each base case:

4.1. ($\Rightarrow$) Construct the corr. sentence

4.2. Recurse

4.3. Add to $\mathcal{E}$

$$\begin{tikzcd}
g(0, m) \arrow[d] & & g(l, 0) \arrow[d] \\
\forall y \in \Delta.\ S(y) \vee \neg S(y) \arrow[d] & & \top \arrow[d] \\
g(0, m) = 0^m \arrow[dr] & & g(l, 0) = 1 \arrow[dl] \\
& \mathcal{E} &
\end{tikzcd}$$

Workflow (2/2)

4. For each base case:

4.1. $(\Rightarrow)$ Construct the corr. sentence

4.2. Recurse

4.3. Add to $\mathcal{E}$

$$\begin{tikzcd}
g(0, m) \arrow[d] & & g(l, 0) \arrow[d] \\
\forall y \in \Delta.\ S(y) \lor \neg S(y) \arrow[d] & & \top \arrow[d] \\
g(0, m) = 0^m \arrow[dr] & & g(l, 0) = 1 \arrow[dl] \\
& \mathcal{E} \arrow[d, "\text{5. Compile to C++}"] & \\
& \boxed{\text{C++ code}} &
\end{tikzcd}$$

Workflow (2/2)

4. For each base case:

$g(0, m)$ $\qquad$ $g(l, 0)$

4.1. $(\Rightarrow)$ Construct the corr. sentence

$\forall y \in \Delta.\ S(y) \vee \neg S(y)$ $\qquad$ $\top$

4.2. Recurse

$g(0, m) = 0^m$ $\qquad$ 4.3. Add to $\mathcal{E}$ $\qquad$ $g(l, 0) = 1$

$$\mathcal{E}$$

$$\mathcal{E} \xrightarrow{\text{5. Compile to C++}} \text{C++ code}$$

$$\text{Domain sizes} \longrightarrow \boxed{\text{C++ code}} \longrightarrow \text{Model count}$$

Finding (a Sufficient Set of) Base Cases

Outline

1. For every function call:
   1.1 For every argument of the form $var - const$:
      1.1.1 Replace the signature parameter with $0, 1, \ldots, const - 1$
   1.2 For every argument of the form $const$:
      1.2.1 Replace the corresponding signature parameter with $const$

Example

The signature of g is $g(l, m)$.

$$\begin{tikzcd}
\text{Function calls:} & g(l-1, m) \arrow[d] & g(l-1, m-1) \arrow[dl] \arrow[d] \\
\text{Base cases:} & g(0, m) & g(l, 0)
\end{tikzcd}$$

Theoretical Guarantees

Theorem

The maximum recursion depth of the compilation algorithm is upper bounded by the number of domains.

Theoretical Guarantees

Theorem

The maximum recursion depth of the compilation algorithm is upper bounded by the number of domains.

Proof (hint).

Each recursive call eliminates a domain. □

Theoretical Guarantees

Theorem

The maximum recursion depth of the compilation algorithm is upper bounded by the number of domains.

Proof (hint).

Each recursive call eliminates a domain. $\square$

Theorem

The evaluation of a recursive function always terminates.

Theoretical Guarantees

Theorem

The maximum recursion depth of the compilation algorithm is upper bounded by the number of domains.

Proof (hint).

Each recursive call eliminates a domain. □

Theorem

The evaluation of a recursive function always terminates.

Proof (hints).

- There exists a topological ordering of functions
- All function calls follow the structure from the previous slide
- Some common-sense assumptions about the evaluation order and previous work

□

From a Base Case to a Sentence

From Previous Work (Dilkas and Belle 2023)

- CRANE associates each function f with a sentence ϕ such that CRANE(ϕ) produces the definition of f
- There is a bijection between the parameters of f and the domains of ϕ

Example

- Base case: $g(0, m)$

From a Base Case to a Sentence

From Previous Work (Dilkas and Belle 2023)

- CRANE associates each function f with a sentence ϕ such that CRANE(ϕ) produces the definition of f
- There is a bijection between the parameters of f and the domains of ϕ

Example

- Base case: $g(\overset{\Gamma}{\overset{\updownarrow}{0}}, \overset{\Delta}{\overset{\updownarrow}{m}})$

From a Base Case to a Sentence

From Previous Work (Dilkas and Belle 2023)

- Crane associates each function f with a sentence ϕ such that Crane(ϕ) produces the definition of f
- There is a bijection between the parameters of f and the domains of ϕ

Example

- Base case: $g(\overset{\Gamma}{\overset{\updownarrow}{0}}, \overset{\Delta}{\overset{\updownarrow}{m}})$
- Part of the sentence of g:

$$\forall x \in \Gamma.\ \forall y \in \Delta.\ S(y) \lor \neg P(x, y) \tag{1}$$

From a Base Case to a Sentence

From Previous Work (Dilkas and Belle 2023)

- CRANE associates each function f with a sentence ϕ such that CRANE(ϕ) produces the definition of f
- There is a bijection between the parameters of f and the domains of ϕ

Example

- Base case: $g(\overset{\Gamma}{\overset{\updownarrow}{0}}, \overset{\Delta}{\overset{\updownarrow}{m}})$
- Part of the sentence of g:

$$\forall x \in \Gamma.\ \forall y \in \Delta.\ S(y) \lor \neg P(x, y) \tag{1}$$

- $g(0, \dots)$ means we need to simplify (1) by assuming $|\Gamma| = 0$

From a Base Case to a Sentence

From Previous Work (Dilkas and Belle 2023)

- CRANE associates each function f with a sentence ϕ such that CRANE(ϕ) produces the definition of f
- There is a bijection between the parameters of f and the domains of ϕ

Example

- Base case: $g(\overset{\Gamma}{\overset{\updownarrow}{0}}, \overset{\Delta}{\overset{\updownarrow}{m}})$
- Part of the sentence of g:

$$\forall x \in \Gamma.\ \forall y \in \Delta.\ S(y) \lor \neg P(x, y) \tag{1}$$

- $g(0, \dots)$ means we need to simplify (1) by assuming $|\Gamma| = 0$
- Result: $\forall y \in \Delta.\ S(y) \lor \neg S(y)$ (Smoothing)

An Outline of the Resulting C++ Program

initialise $\text{Cache}_{g(0,m)}$, $\text{Cache}_{g(l,0)}$, Cache_{g}, and Cache_{f};

An Outline of the Resulting C++ Program

initialise $\text{Cache}_{g(0,m)}$, $\text{Cache}_{g(l,0)}$, Cache_{g}, and Cache_{f};
Function $g_{0,m}(m)$**:** …
Function $g_{l,0}(l)$**:** …

An Outline of the Resulting C++ Program

initialise $\text{Cache}_{g(0,m)}$, $\text{Cache}_{g(l,0)}$, Cache_g, and Cache_f;
Function $g_{0,m}(m)$**:** …
Function $g_{l,0}(l)$**:** …
Function $g(l, m)$**:**
    if $(l, m) \in \text{Cache}_g$ **then return** $\text{Cache}_g(l, m)$;
    if $l = 0$ **then return** $g_{0,m}(m)$;
    if $m = 0$ **then return** $g_{l,0}(l)$;
    $r \leftarrow g(l-1, m) + mg(l-1, m-1)$;
    $\text{Cache}_g(l, m) \leftarrow r$;
    return r;
Function $f(m, n)$**:** …
Function `Main`**:**
    $(m, n) \leftarrow$ `ParseCommandLineArguments()`;
    return $f(m, n)$;

Benchmarks

- Friends & Smokers

$$(\forall x, y \in \Delta.\ S(x) \land F(x, y) \to S(y)) \land (\forall x \in \Delta.\ S(x) \to C(x))$$

Benchmarks

- Friends & Smokers

$$(\forall x, y \in \Delta.\ S(x) \land F(x, y) \to S(y)) \land (\forall x \in \Delta.\ S(x) \to C(x))$$

- Functions

$$(\forall x \in \Gamma.\ \exists y \in \Delta.\ P(x, y)) \land \\ (\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \to y = z)$$

Benchmarks

- Friends & Smokers

$$(\forall x, y \in \Delta.\ S(x) \land F(x, y) \to S(y)) \land (\forall x \in \Delta.\ S(x) \to C(x))$$

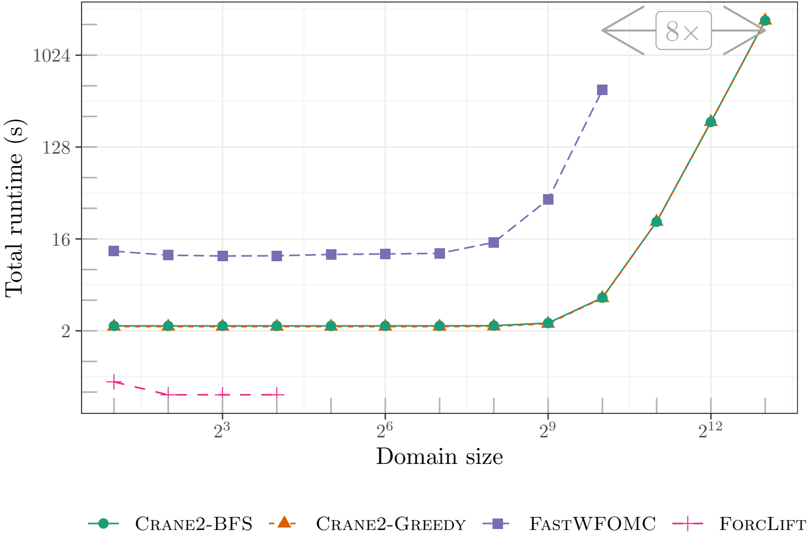
- Functions

$$(\forall x \in \Gamma.\ \exists y \in \Delta.\ P(x, y)) \land$$
$$(\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \to y = z)$$

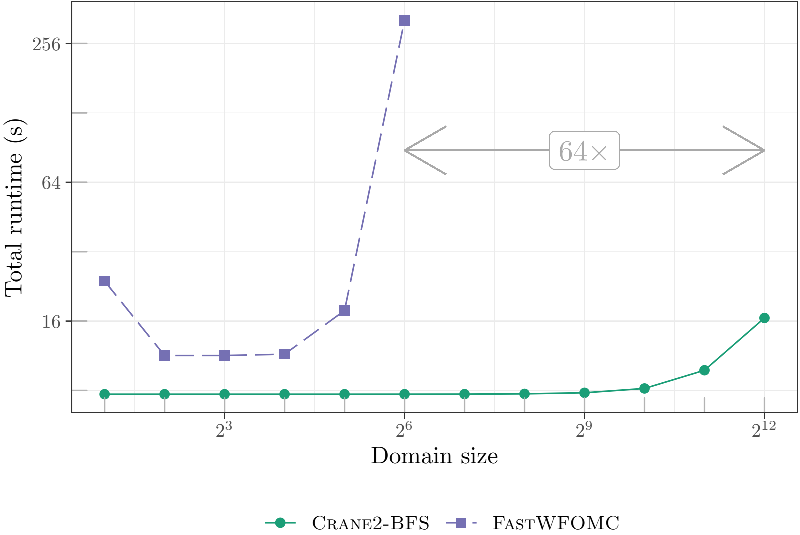
- Bijections

$$(\forall x \in \Gamma.\ \exists y \in \Delta.\ P(x, y)) \land$$
$$(\forall y \in \Delta.\ \exists x \in \Gamma.\ P(x, y)) \land$$
$$(\forall x \in \Gamma.\ \forall y, z \in \Delta.\ P(x, y) \land P(x, z) \to y = z) \land$$
$$(\forall x, z \in \Gamma.\ \forall y \in \Delta.\ P(x, y) \land P(z, y) \to x = z)$$

Friends & Smokers

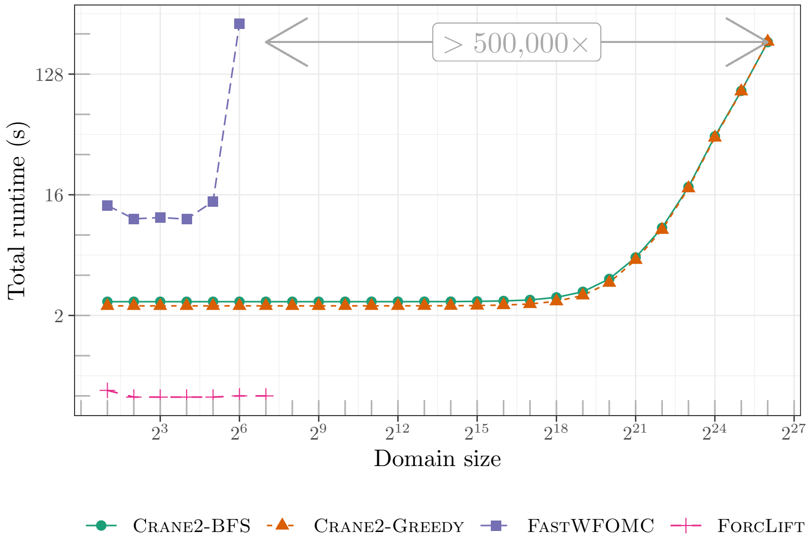


Bijections



256
64
16
Total runtime (s)
64×
2^3
2^6
2^9
2^{12}
Domain size
CRANE2-BFS
FASTWFOMC

Functions



> 500,000×
Total runtime (s)
128
16
2
2^3
2^6
2^9
2^12
2^15
2^18
2^21
2^24
2^27
Domain size
Crane2-BFS
Crane2-Greedy
FastWFOMC
ForcLift

Summary & Future Work

Contributions

Completeness: recursive solutions now come with base cases

Usability: compilation to C++ programs

Scalability compared to other FOMC algorithms

- 8 to 500,000 times higher domain sizes

Summary & Future Work

Contributions

Completeness: recursive solutions now come with base cases

Usability: compilation to C++ programs

Scalability compared to other FOMC algorithms

- 8 to 500,000 times higher domain sizes

Future Work

- Support for weighted counting (trivial)
- Experiments on a large set of benchmarks
- Completeness for fragments of first-order logic
- Fine-grained complexity