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

First-Order Model Counting

The Problem with CRANE

A Solution Produced for the Bijection-Counting Problem

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

The Problem with Crane

A Solution Produced for the Bijection-Counting Problem

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

Issues

Completeness: what are the base cases of g?

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

Knowledge Compilation Workflow (1/2)

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

Knowledge Compilation 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

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

Knowledge Compilation 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

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

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

Knowledge Compilation Workflow (2/2)

4. For each base case:

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

Knowledge Compilation Workflow (2/2)

4. For each base case:

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

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

Knowledge Compilation Workflow (2/2)

4. For each base case:

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

4.2. Recurse

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

Knowledge Compilation Workflow (2/2)

4. For each base case:

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

4.2. Recurse

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

Knowledge Compilation Workflow (2/2)

4. For each base case:

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

4.2. Recurse

4.3. Add to $\mathcal{E}$

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

Knowledge Compilation Workflow (2/2)

4. For each base case:

$$\begin{tikzcd}
g(0, m) \arrow[d] & \text{4.1. } (\Rightarrow) \text{ Construct the corr. sentence} & g(l, 0) \arrow[d] \\
\forall y \in \Delta.\ S(y) \lor \neg S(y) \arrow[d] & & \top \arrow[d] \\
g(0, m) = 0^m \arrow[dr] & \text{4.2. Recurse} \quad \text{4.3. Add to } \mathcal{E} & g(l, 0) = 1 \arrow[dl] \\
& \mathcal{E} \arrow[d, "\text{5. Compile to C++}"] & \\
& \boxed{\text{C++ code}} &
\end{tikzcd}$$

Knowledge Compilation Workflow (2/2)

$$\begin{tikzcd}
 & \text{4. For each base case:} & \\
g(0, m) \arrow[d] & & g(l, 0) \arrow[d] \\
 & \text{4.1. } (\Rightarrow) \text{ Construct the corr. sentence} & \\
\forall y \in \Delta.\ S(y) \lor \neg S(y) \arrow[d] & & \top \arrow[d] \\
 & \text{4.2. Recurse} & \\
g(0, m) = 0^m \arrow[dr] & \text{4.3. Add to } \mathcal{E} & g(l, 0) = 1 \arrow[dl] \\
 & \mathcal{E} \arrow[d, "\text{5. Compile to C++}"] & \\
\text{Domain sizes} \arrow[r] & \boxed{\text{C++ code}} \arrow[r] & \text{Model count}
\end{tikzcd}$$

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 Results

Theorem

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

Theoretical Results

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 Results

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.

Theoretical Results

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 $\textsc{Crane}(\phi)$ 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 $\textsc{Crane}(\phi)$ produces the definition of f
- There is a bijection between the parameters of f and the domains of ϕ

Example

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

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 $\textsc{Crane}(\phi)$ produces the definition of f
- There is a bijection between the parameters of f and the domains of ϕ

Example

- Base case: $g(\overset{\overset{\Gamma}{\updownarrow}}{0}, \overset{\overset{\Delta}{\updownarrow}}{m})$
- 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 $\textsc{Crane}(\phi)$ produces the definition of f
- There is a bijection between the parameters of f and the domains of ϕ

Example

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

Benchmarks

- Friends & Smokers

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

Benchmarks

- Friends & Smokers

$$(\forall x, y \in \Delta.\ S(x) \wedge F(x, y) \rightarrow S(y)) \wedge (\forall x \in \Delta.\ S(x) \rightarrow C(x))$$

- Functions

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

Benchmarks

- Friends & Smokers

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

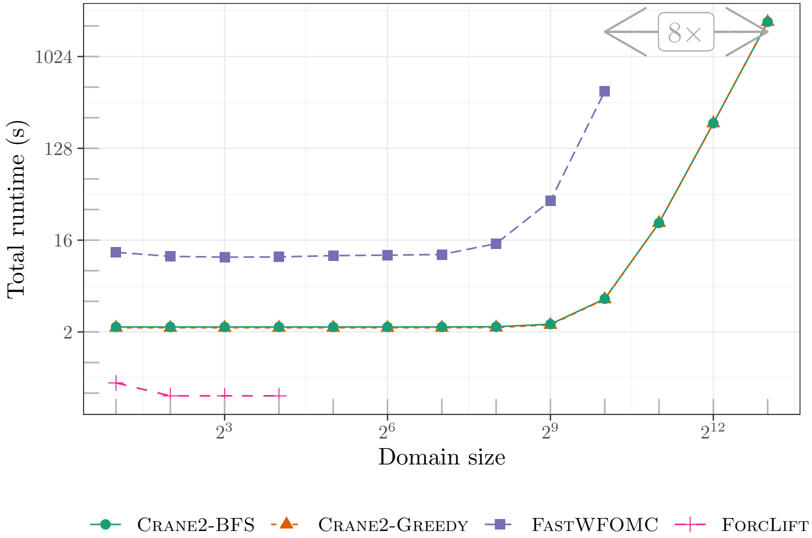
- Functions

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

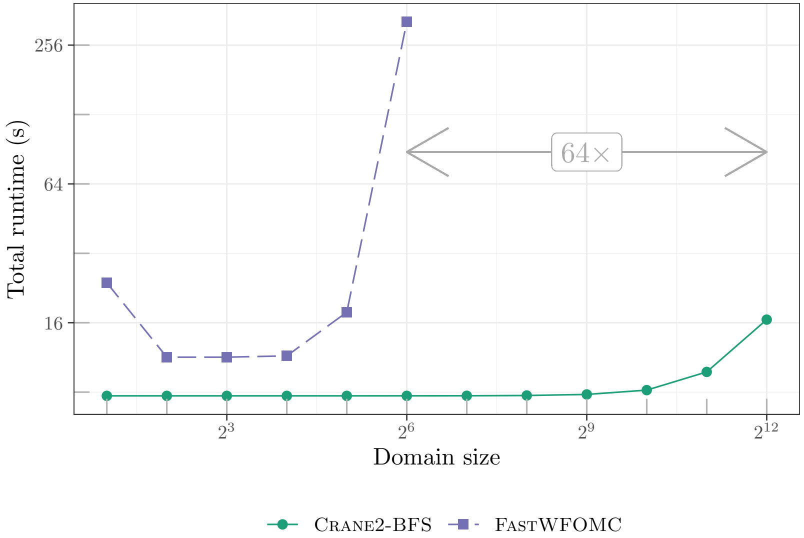
- Bijections

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

Friends & Smokers

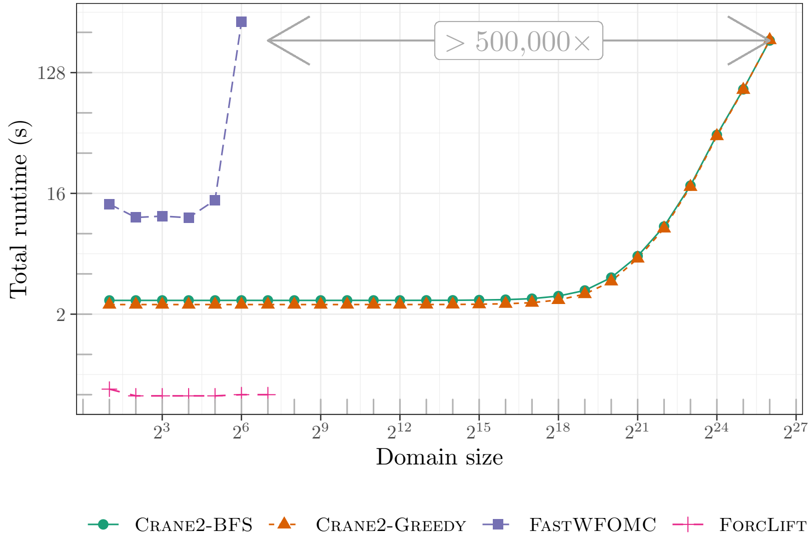


Bijections



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

TODO: and future work