Empowering Domain Recursion in Symmetric Weighted First-Order Model Counting

10th January 2022

1 Basic Definitions

Things I might need to explain.

- notation: Im
- atom
- inequality constraint
- Vars, $\mathrm{Vars}(c) = \mathrm{Vars}(P) \cup \mathrm{Vars}(N) \cup \mathrm{Vars}(C)$
- Doms on both formulas and clauses. $\mathrm{Doms}(c) = \mathrm{Im}\,\delta$, and $\mathrm{Doms}(\phi) = \bigcup_{c \in \phi} \mathrm{Doms}(c)$.
- the hash codes of clauses and formulas. Introduce the # notation.
- substitution
- (strict) equality of clauses
- size of a domain, how each domain is partitioned into two, and how we iterate over all possible integer partitions of length two.
- maybe: endomorphism, notation for partial function, notation for powerset

Let $\mathscr{V}$ be the set of circuit nodes.
TODO: merge κ and ι into one.

Definition 1. A *domain* is a set with elements not used anywhere else.[1] Let $\mathscr{D}$ be the set of all domains (note that this set expands during compilation).

We now define two partial maps π and κ with the same domain $\mathrm{dom}(\pi) = \mathrm{dom}(\kappa) \subset \mathscr{D}$. First, let $\pi\colon \mathscr{D} \nrightarrow \mathscr{D}$ be a partial endomorphism on $\mathscr{D}$ that denotes the *parent* relation, i.e., if $\pi(d) = e$ for some $d, e \in \mathscr{D}$, then we call e the parent (domain) of d, and e a child of d. Intuitively, π arranges all domains into a forest. Second, let $\kappa\colon \mathscr{D} \nrightarrow \mathscr{V}$ be a partial map that assigns a *cause node* to all non-root domains. Third, let $\iota\colon \mathscr{D} \nrightarrow \{\,0, 1\,\}$ unambiguously order the children of any internal node, i.e., $\iota(d) \neq \iota(e)$ whenever $\pi(d) = \pi(e)$ for any $d, e \in \mathscr{D}$.[2]

Definition 2. A *clause* is a triple $c = (P, N, C, \delta)$, where P and N are sets of atoms interpreted as positive and negative literals respectively, C is a set of inequality constraints, and $\delta\colon \mathrm{Vars}(c) \to \mathscr{D}$ is a function that maps all variables in c to their domains. Two clauses c and $d = (P', N', C', \delta')$ are *equivalent* (written $c \equiv d$) if there is a bijection $\beta\colon \mathrm{Vars}(c) \to \mathrm{Vars}(d)$ such that $c\beta = d\beta$.

A *formula* is a set of clauses.

[1] In the context of functions, the domain of a function f retains its usual meaning and is denoted $\mathrm{dom}(f)$.
[2] Here, each internal node has at most two children.

2 Identifying Possibilities for Recursion

For succinctness, let $\mathscr{S} = \mathscr{D} \times \mathscr{D} \times 2^{\mathscr{V} \times \{ 0,1 \}}$. TODO: explain the idea.

Let $\nu\colon \mathscr{S} \to 2^{\mathscr{S}}$ be a map defined as

$$\nu(a, b, S) = \begin{cases} \{ (\pi(a), b, S \cup \{ \kappa(a) \}) \} & \text{if } a \in \mathrm{dom}(\pi) \\ \varnothing & \text{if } a \notin \mathrm{dom}(\pi) \end{cases}$$

for all $(a, b, S) \in \mathscr{S}$. Furthermore, let $\nu'\colon 2^{\mathscr{S}} \to 2^{\mathscr{S}}$ be an endomorphism defined as $\nu'(S) = \bigcup_{s \in S} \nu(s)$ for all $S \subseteq \mathscr{S}$.

Algorithm 1: A recursive function for checking whether one can reuse the circuit for computing $\mathrm{WMC}(\psi)$ to compute $\mathrm{WMC}(\phi)$. Both ϕ and ψ are formulas, and $\rho\colon \mathrm{Doms}(\phi) \nrightarrow \mathrm{Doms}(\psi)$ is a partial map.

Function `identifyRecursion(`ϕ, ψ, $\rho = \varnothing$`)`**:**

  if $\mid \phi \mid \neq \mid \psi \mid$ **or** $\#\phi \neq \#\psi$ **then** TODO: none;

  else if $\phi = \varnothing$ **then**

    TODO

  else

    TODO

Example 1. Let...