The Power of Literal Equivalence in Model Counting

Yong Lai, Kuldeep S. Meel and Roland H. C. Yap

College of Computer Science and Technology – JLU
Department of Computer Science – NUS

Model Counting

- Model Counting (#SAT): computing the number of satisfying assignments of a propositional formula φ.
- An important problem in computer science: Valiant showed that the problem of model counting is #P-complete; Toda showed that PH $\subseteq P^{\#P}$.
- Applications: probabilistic inference, neural network verification, network reliability, computational biology, and the like.

Exact Model Counters

Search-based

- Cachet, 2004
- sharpSAT, 2006
- Ganak, 2019

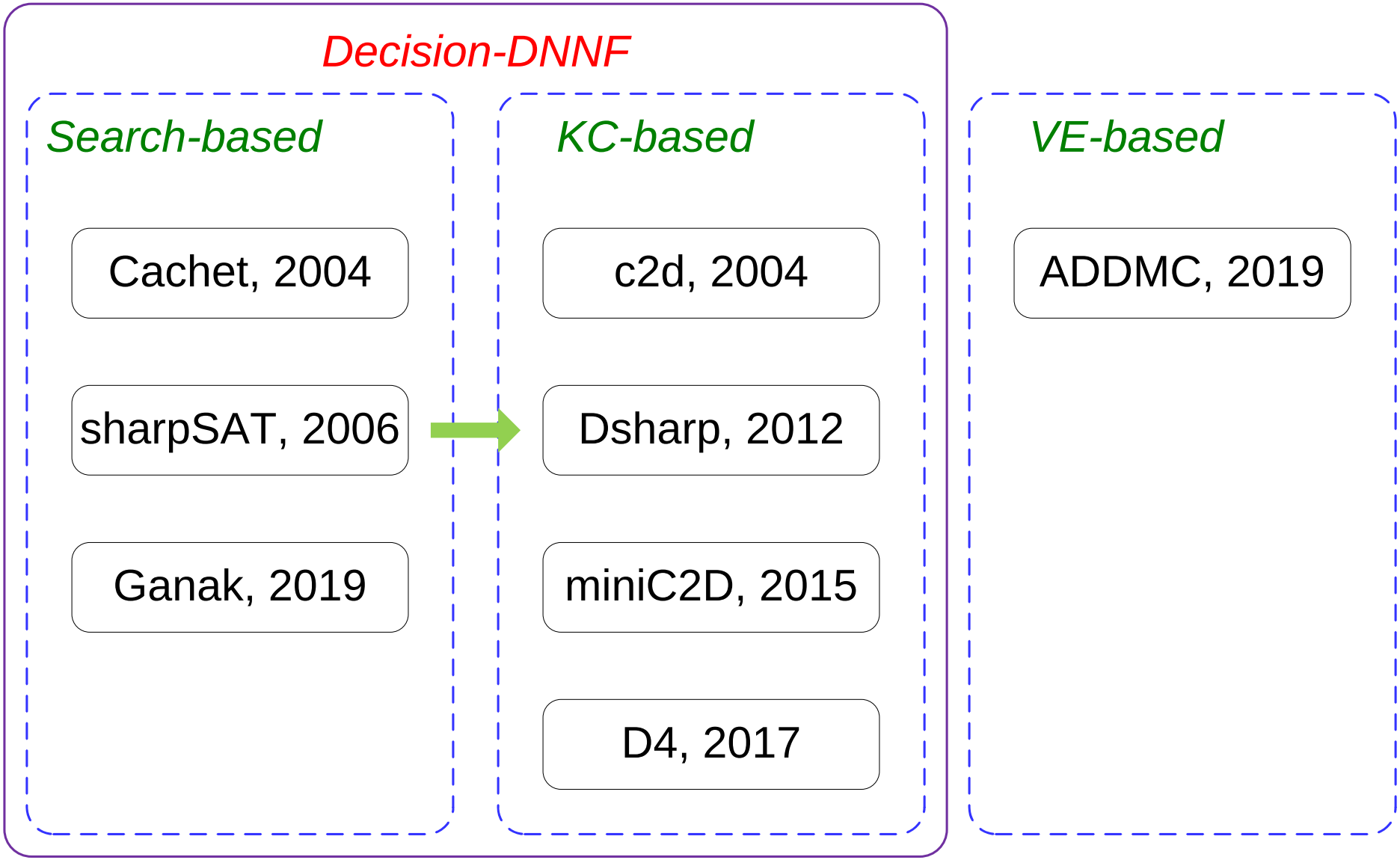
KC-based

- c2d, 2004
- Dsharp, 2012
- miniC2D, 2015
- D4, 2017

VE-based

- ADDMC, 2019

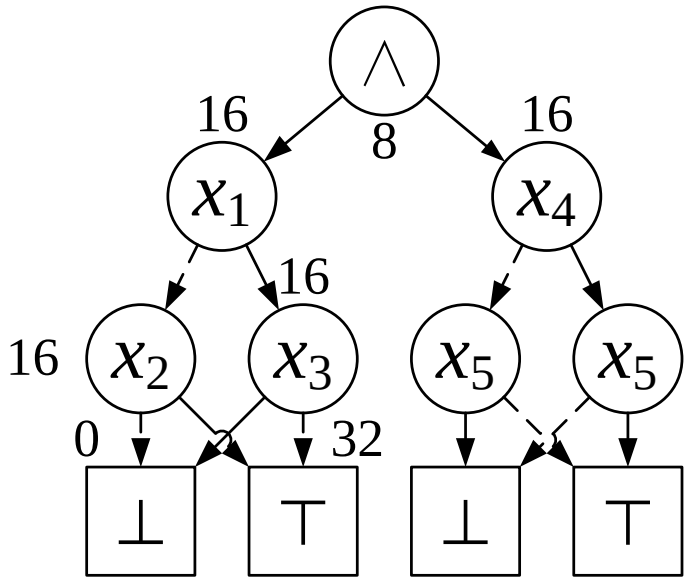
Exact Model Counters



Decision-DNNF

Proposition

Given a Decision-DNNF formula, the model count can be recursively computed in linear time.



Decision-DNNF

Motivation

- There are a great number of sets of literal equivalences in solving model counting problems, e.g.,
  - Circuit 2bitadd_11: 11580 literal equivalences
  - Planning instance logistics.c: 757 literal equivalences
  - Program synthesis instance sygus_09A-1: 21851 literal equivalences

Decision-DNNF

Motivation

- There are a great number of sets of literal equivalences in solving model counting problems, e.g.,
  - Circuit 2bitadd_11: 11580 literal equivalences
  - Planning instance logistics.c: 757 literal equivalences
  - Program synthesis instance sygus_09A-1: 21851 literal equivalences
- Decision-DNNF is not expressive enough to capture literal equivalences.

Decision-DNNF

Motivation

- There are a great number of sets of literal equivalences in solving model counting problems, e.g.,
  - Circuit 2bitadd_11: 11580 literal equivalences
  - Planning instance logistics.c: 757 literal equivalences
  - Program synthesis instance sygus_09A-1: 21851 literal equivalences
- Decision-DNNF is not expressive enough to capture literal equivalences.
- Can we design an efficient model counting technique such that its trace is a generalization of Decision-DNNF to capture literal equivalences?

Contents

1. Capturing Literal Equivalence
2. Identifying New Language CCDD
3. ExactMC: A Scalable Model Counter
4. Experiments
5. Conclusion and Future Work

Contents

1. Capturing Literal Equivalence
2. Identifying New Language CCDD
3. ExactMC: A Scalable Model Counter
4. Experiments
5. Conclusion and Future Work

Capturing Literal Equivalence

- Literal equivalence: $l \leftrightarrow l'$ for two literals.
- Prime literal equivalences (unique representation): Given a set of literal equivalences E, for each equivalence class of literals with a minimum positive literal x, we pick a $x \leftrightarrow l$ for each $l \neq x$ in the class.

Capturing Literal Equivalence

Example

Given $E = \{\neg x_1 \leftrightarrow x_3, \neg x_4 \leftrightarrow x_3, \neg x_2 \leftrightarrow \neg x_6, x_5 \leftrightarrow x_5\}$, we have

- $[x_1] = [\neg x_3] = [x_4] = \{x_1, \neg x_3, x_4\}$,
  $[\neg x_1] = [x_3] = [\neg x_4] = \{\neg x_1, x_3, \neg x_4\}$,
  $[x_2] = [x_6] = \{x_2, x_6\}$,
  $[\neg x_2] = [\neg x_6] = \{\neg x_2, \neg x_6\}$,
  $[x_5] = \{x_5\}$,
  $[\neg x_5] = [\neg x_5]$,
  $[x_7] = [x_7]$,
  $[\neg x_7] = [\neg x_7]$,
  ...
- $\lfloor E \rfloor = \{x_1 \leftrightarrow \neg x_3, x_1 \leftrightarrow x_4, x_2 \leftrightarrow x_6\}$.

Capturing Literal Equivalence

Claim

Let φ be a formula and let E be a set of prime literal equivalences implied by φ. We can obtain another formula φ' by performing a *literal-substitution*: replace each l (resp. $\neg l$) in φ with x (resp. $\neg x$) for each $x \leftrightarrow l \in E$. Note that, $\varphi \equiv \varphi' \wedge \bigwedge_{x \leftrightarrow l \in E} x \leftrightarrow l$.

Capturing Literal Equivalence

Claim

Let φ be a formula and let E be a set of prime literal equivalences implied by φ. We can obtain another formula φ' by performing a *literal-substitution*: replace each l (resp. $\neg l$) in φ with x (resp. $\neg x$) for each $x \leftrightarrow l \in E$. Note that, $\varphi \equiv \varphi' \wedge \bigwedge_{x \leftrightarrow l \in E} x \leftrightarrow l$.

Example

$(x_1 \vee \neg x_3 \vee x_4 \vee x_7) \wedge (x_1 \vee x_3 \vee x_5) \wedge$
$(\neg x_1 \leftrightarrow x_3) \wedge (\neg x_4 \leftrightarrow x_3) \wedge (\neg x_2 \leftrightarrow \neg x_6) \wedge (x_5 \leftrightarrow x_5)$

Capturing Literal Equivalence

Claim

Let φ be a formula and let E be a set of prime literal equivalences implied by φ. We can obtain another formula φ' by performing a *literal-substitution*: replace each l (resp. $\neg l$) in φ with x (resp. $\neg x$) for each $x \leftrightarrow l \in E$. Note that, $\varphi \equiv \varphi' \wedge \bigwedge_{x \leftrightarrow l \in E} x \leftrightarrow l$.

Example

$(x_1 \vee \neg x_3 \vee x_4 \vee x_7) \wedge (x_1 \vee x_3 \vee x_5) \wedge$
$(\neg x_1 \leftrightarrow x_3) \wedge (\neg x_4 \leftrightarrow x_3) \wedge (\neg x_2 \leftrightarrow \neg x_6) \wedge (x_5 \leftrightarrow x_5)$
$\equiv (x_1 \vee \neg x_3 \vee x_4 \vee x_7) \wedge (x_1 \vee x_3 \vee x_5) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

Capturing Literal Equivalence

Claim

Let φ be a formula and let E be a set of prime literal equivalences implied by φ. We can obtain another formula φ' by performing a *literal-substitution*: replace each l (resp. $\neg l$) in φ with x (resp. $\neg x$) for each $x \leftrightarrow l \in E$. Note that, $\varphi \equiv \varphi' \wedge \bigwedge_{x \leftrightarrow l \in E} x \leftrightarrow l$.

Example

$(x_1 \vee \neg x_3 \vee x_4 \vee x_7) \wedge (x_1 \vee x_3 \vee x_5) \wedge$
$(\neg x_1 \leftrightarrow x_3) \wedge (\neg x_4 \leftrightarrow x_3) \wedge (\neg x_2 \leftrightarrow \neg x_6) \wedge (x_5 \leftrightarrow x_5)$
$\equiv (x_1 \vee \neg x_3 \vee x_4 \vee x_7) \wedge (x_1 \vee x_3 \vee x_5) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$
$\equiv (x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

Capturing Literal Equivalence

Claim

Let φ be a formula and let E be a set of prime literal equivalences implied by φ. We can obtain another formula φ' by performing a *literal-substitution*: replace each l (resp. $\neg l$) in φ with x (resp. $\neg x$) for each $x \leftrightarrow l \in E$. Note that, $\varphi \equiv \varphi' \wedge \bigwedge_{x \leftrightarrow l \in E} x \leftrightarrow l$.

Example

$(x_1 \vee \neg x_3 \vee x_4 \vee x_7) \wedge (x_1 \vee x_3 \vee x_5) \wedge$
$(\neg x_1 \leftrightarrow x_3) \wedge (\neg x_4 \leftrightarrow x_3) \wedge (\neg x_2 \leftrightarrow \neg x_6) \wedge (x_5 \leftrightarrow x_5)$
$\equiv (x_1 \vee \neg x_3 \vee x_4 \vee x_7) \wedge (x_1 \vee x_3 \vee x_5) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$
$\equiv (x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

simpler :)

Capturing Literal Equivalence

Definition (kernelized conjunction)

A kernelized conjunction node v is a conjunction node consisting of a core child and a set of prime literal equivalences where for each literal equivalence $x \leftrightarrow l$, $var(l)$ does not appear in the sub-graph rooted at the core child.

Capturing Literal Equivalence

Definition (kernelized conjunction)

A kernelized conjunction node v is a conjunction node consisting of a core child and a set of prime literal equivalences where for each literal equivalence $x \leftrightarrow l$, $var(l)$ does not appear in the sub-graph rooted at the core child.

Example

$(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

Capturing Literal Equivalence

Proposition

For a kernelized conjunction v over X with n prime literal equivalences, if $\vartheta(ch_{core}(v))$ has m models over X, then $\vartheta(v)$ has $\frac{m}{2^n}$ models over X.

Capturing Literal Equivalence

Proposition

For a kernelized conjunction v over X with n prime literal equivalences, if $\vartheta(ch_{core}(v))$ has m models over X, then $\vartheta(v)$ has $\frac{m}{2^n}$ models over X.

Example

$(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

Capturing Literal Equivalence

Proposition

For a kernelized conjunction v over X with n prime literal equivalences, if $\vartheta(ch_{core}(v))$ has m models over X, then $\vartheta(v)$ has $\frac{m}{2^n}$ models over X.

Example

$(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

- $x_1 \vee x_7$: 96

Capturing Literal Equivalence

Proposition

For a kernelized conjunction v over X with n prime literal equivalences, if $\vartheta(ch_{core}(v))$ has m models over X, then $\vartheta(v)$ has $\frac{m}{2^n}$ models over X.

Example

$(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

- $x_1 \vee x_7$: 96
- $\left[x_1 \vee x_7\right] \wedge (x_1 \leftrightarrow \neg x_3)$: 48

Capturing Literal Equivalence

Proposition

For a kernelized conjunction v over X with n prime literal equivalences, if $\vartheta(ch_{core}(v))$ has m models over X, then $\vartheta(v)$ has $\frac{m}{2^n}$ models over X.

Example

$(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

- $x_1 \vee x_7$: 96
- $\left[x_1 \vee x_7\right] \wedge (x_1 \leftrightarrow \neg x_3)$: 48
- $\left[(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3)\right] \wedge (x_1 \leftrightarrow x_4)$: 24

Capturing Literal Equivalence

Proposition

For a kernelized conjunction v over X with n prime literal equivalences, if $\vartheta(ch_{core}(v))$ has m models over X, then $\vartheta(v)$ has $\frac{m}{2^n}$ models over X.

Example

$(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4) \wedge (x_2 \leftrightarrow x_6)$

- $x_1 \vee x_7$: 96
- $\left[x_1 \vee x_7\right] \wedge (x_1 \leftrightarrow \neg x_3)$: 48
- $\left[(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3)\right] \wedge (x_1 \leftrightarrow x_4)$: 24
- $\left[(x_1 \vee x_7) \wedge (x_1 \leftrightarrow \neg x_3) \wedge (x_1 \leftrightarrow x_4)\right] \wedge (x_2 \leftrightarrow x_6)$: 12

Contents

1. Capturing Literal Equivalence
2. Identifying New Language CCDD
3. ExactMC: A Scalable Model Counter
4. Experiments
5. Conclusion and Future Work

Conjunction & Decision Diagram

Definition (Conjunction & Decision Diagram, CDD)

A Conjunction & Decision Diagram (CDD) is a rooted DAG wherein each node u is labeled with a symbol $sym(u)$:

- Leaf node: $sym(u) = \bot$ (*false*) or $\top$ (*true*).
- Conjunction node: $sym(u) = \wedge$, representing $\bigwedge_{v \in Ch(v)} \vartheta(v)$.
- Decision node: $sym(u)$ is a variable, representing $[\neg sym(u) \wedge \vartheta(lo(u))] \vee [sym(u) \wedge \vartheta(hi(u))]$

Constrained CDD

Definition (Constrained CDD, CCDD)

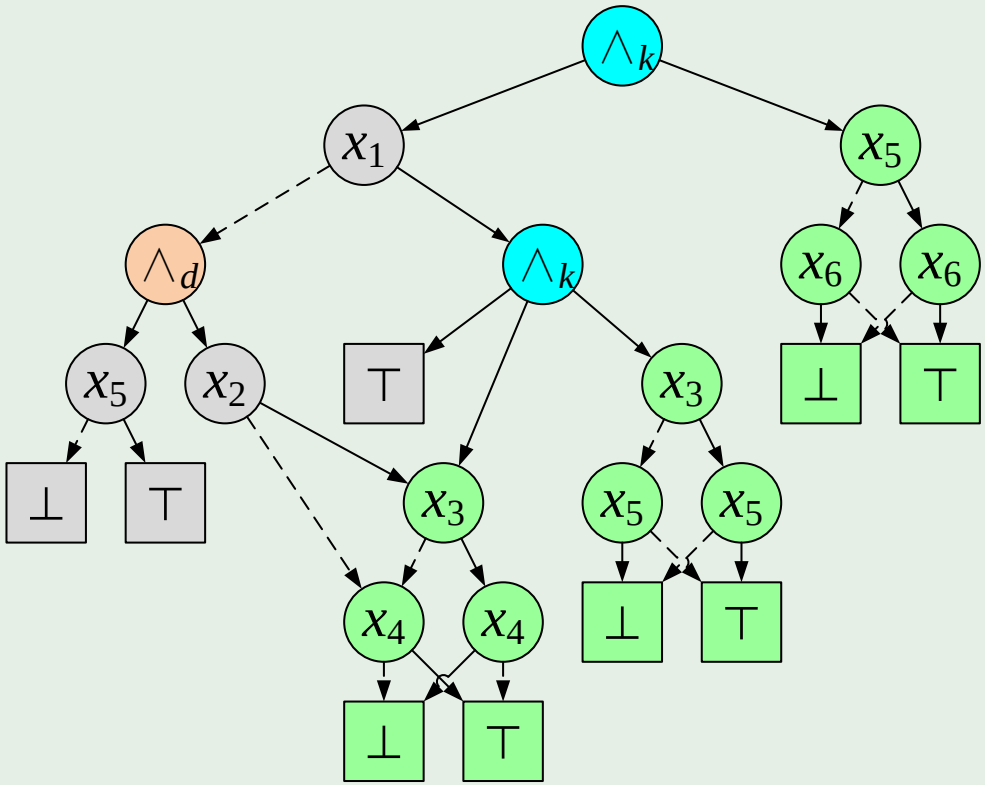
A CDD is called *constrained* if

- Each decision node u and its decision descendant v satisfy $sym(u) \neq sym(v)$, and
- Each conjunction node v is either: (i) decomposed; or (ii) kernelized.

Constrained CDD

Example

The formula $\Big[\big[\neg x_1 \wedge x_5 \wedge [(\neg x_2 \wedge x_4) \vee (x_2 \wedge (x_3 \leftrightarrow \neg x_4))]\big] \vee \big[x_1 \wedge (x_3 \leftrightarrow \neg x_4) \wedge (x_3 \leftrightarrow x_5)\big]\Big] \wedge (x_5 \leftrightarrow x_6)$ can be represented as follows:



Linear Model Counting

Proposition

Given a CCDD node u over X and a node v in $\mathcal{D}_u$, we use $CT(v)$ to denote the number of models of $\vartheta(v)$ over X. Then $CT(u)$ can be recursively computed in linear time:

$$CT(u) = \begin{cases} 0 \text{ or } 2^{|X|} & sym(u) = \bot \text{ or } \top \\ c^{-1} \cdot \prod_{v \in Ch(u)} CT(v) & sym(u) = \wedge_d \\ \dfrac{CT(ch_{core}(v))}{2^{|Ch(u)|-1}} & sym(u) = \wedge_k \\ \dfrac{CT(lo(u)) + CT(hi(u))}{2} & otherwise \end{cases}$$

where $c = 2^{(|Ch(u)|-1)\cdot|X|}$.

Contents

1. Capturing Literal Equivalence
2. Identifying New Language CCDD
3. ExactMC: A Scalable Model Counter
4. Experiments
5. Conclusion and Future Work

Counting Algorithm

Algorithm 1: ExactMC(φ, X)

1 **if** $\varphi = \mathit{false}$ **then return** 0
2 **if** $\varphi = \mathit{true}$ **then return** $2^{|X|}$
3 **if** $\mathit{Cache}(\varphi) \neq \mathit{nil}$ **then return** $\mathit{Cache}(\varphi)$
4 **if** ShouldKernelize(φ) **then**
5 $E \leftarrow$ DetectLitEqu(φ)
6 **if** $||\lfloor E \rfloor|| > 0$ **then**
7 $\hat{\varphi} \leftarrow$ ConstructCore($\varphi, \lfloor E \rfloor$); $c \leftarrow$ ExactMC($\hat{\varphi}, X$)
8 **return** $\mathit{Cache}(\varphi) \leftarrow \frac{c}{2^{||\lfloor E \rfloor||}}$
9 $\Psi \leftarrow$ Decompose(φ)
10 **if** $|\Psi| > 1$ **then**
11 $c \leftarrow \prod_{\psi \in \Psi} \{\text{ExactMC}(\psi, X)\}$
12 **return** $\mathit{Cache}(\varphi) \leftarrow \frac{c}{2^{(|\Psi|-1)\cdot|X|}}$
13 **else**
14 $x \leftarrow$ PickGoodVar(φ)
15 $c_0 \leftarrow \text{ExactMC}(\varphi[x \mapsto \mathit{false}], X)$; $c_1 \leftarrow \text{ExactMC}(\varphi[x \mapsto \mathit{true}], X)$
16 **return** $\mathit{Cache}(\varphi) \leftarrow \frac{c_0+c_1}{2}$

Counting Algorithm

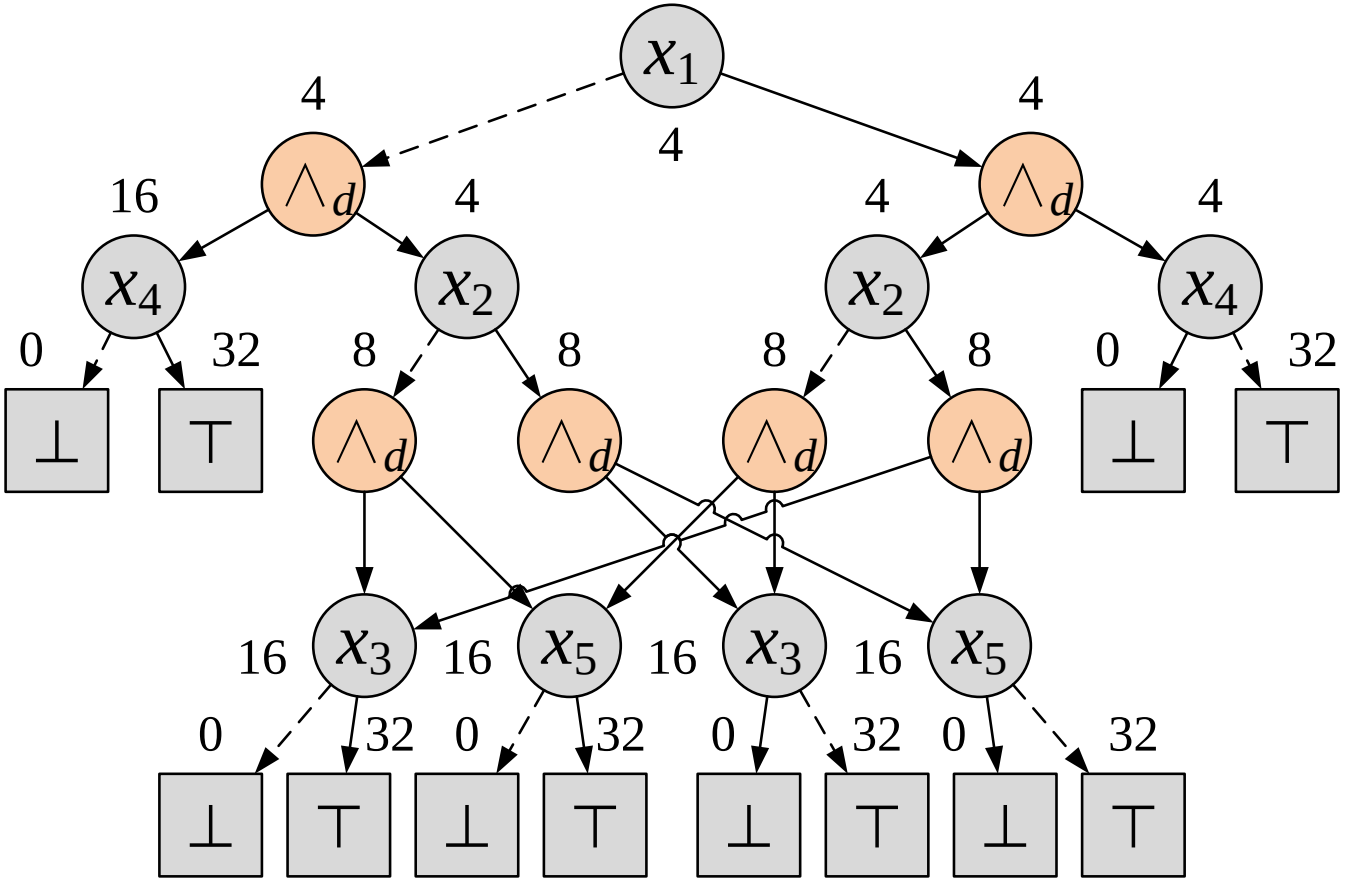
Example

Consider the CNF formula φ:

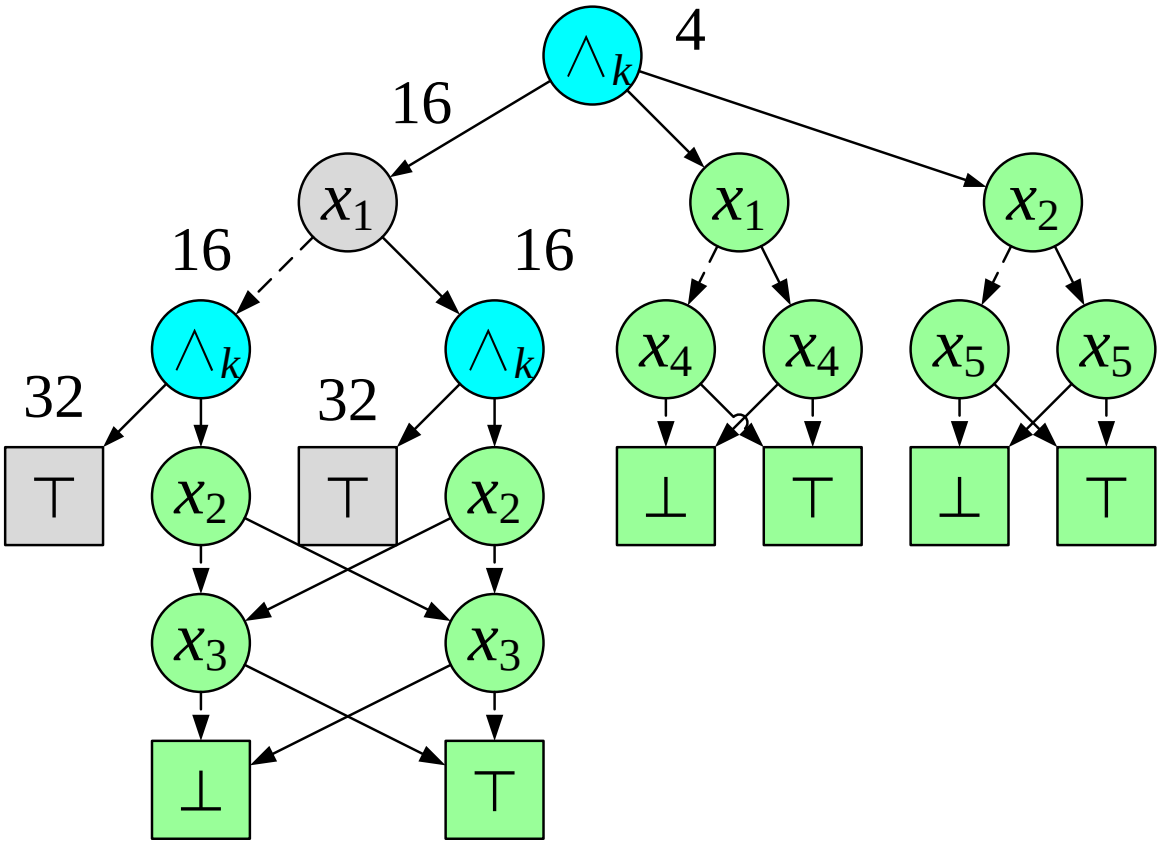
$$\varphi = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \\ \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_4) \\ \wedge (x_1 \vee x_4) \wedge (\neg x_2 \vee \neg x_5) \wedge (x_2 \vee x_5)$$

with $X = \{x_1, \ldots, x_5\}$.

Counting Algorithm



Compiling Algorithm



Contents

1. Capturing Literal Equivalence
2. Identifying New Language CCDD
3. ExactMC: A Scalable Model Counter
4. Experiments
5. Conclusion and Future Work

Experiment Setup

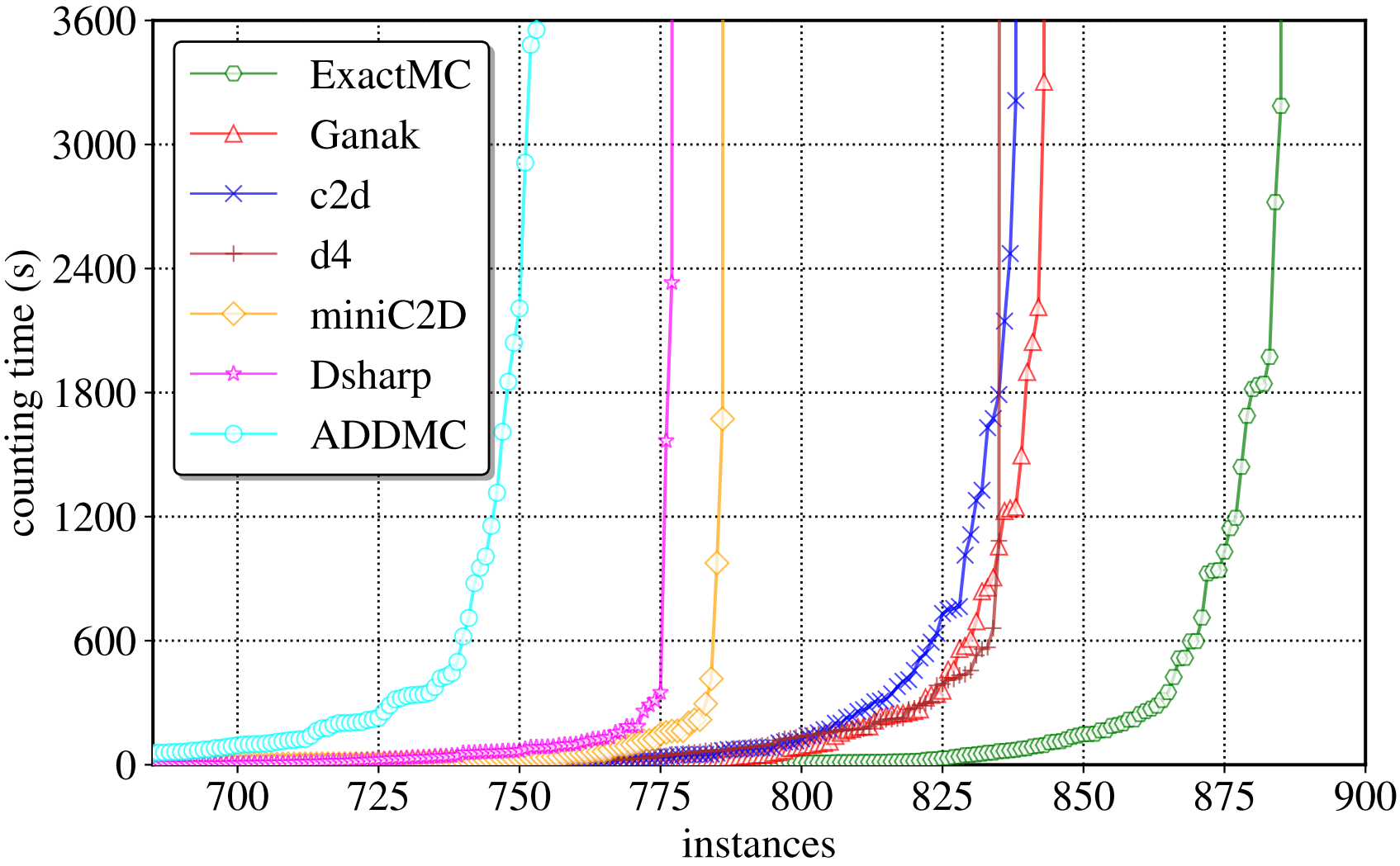
- Computer: 2xE5-2690v3 CPUs with 24 cores and 96GB of RAM
- Timeout: 3600 seconds
- Memory limit: 4GB

Total Performance

Table: Comparative counting performance between Ganak, c2d, Dsharp, miniC2D, D4, ADDMC, and ExactMC, where each cell below tool refers to the number of solved instances

domain	Ganak	c2d	Dsharp	miniC2D	D4	ADDMC	ExactMC
Bayesian-Networks	170	183	168	183	179	**191**	186
BlastedSMT	163	160	163	155	162	166	**169**
Circuit	49	50	47	48	49	45	**51**
Configuration	**35**	**35**	21	28	33	21	31
Inductive-Inference	18	19	15	15	18	3	**22**
Model-Checking	73	**74**	66	71	72	64	**74**
Planning	207	209	192	201	206	187	**212**
Program-Synthesis	96	76	84	68	90	52	**108**
QIF	**32**	**32**	21	17	26	24	**32**
Total (1114)	843	838	777	786	835	753	**885**

Time Efficiency



Contents

1. Capturing Literal Equivalence
2. Identifying New Language CCDD
3. ExactMC: A Scalable Model Counter
4. Experiments
5. Conclusion and Future Work

Conclusion and Future Work

- The main contribution: the new language CCDD supporting linear model counting and the practical model counter ExactMC.
- Future work: new decision heuristics, new caching schemes, etc.

Conclusion and Future Work

- The main contribution: the new language CCDD supporting linear model counting and the practical model counter ExactMC.
- Future work: new decision heuristics, new caching schemes, etc.
- Open Source: `https://github.com/meelgroup/KCBox`

Conclusion and Future Work

- The main contribution: the new language CCDD supporting linear model counting and the practical model counter ExactMC.
- Future work: new decision heuristics, new caching schemes, etc.
- Open Source: `https://github.com/meelgroup/KCBox`

Thank you for your attention!