Model Counting at CRIL

Stefan Mengel and Pierre Marquis

Université d'Artois
CRIL UMR CNRS 8188
France

Nanjing-Artois Computer Science Seminar, June 6th 2016

Introduction: Model counting

2

Model Counting (#*SAT*)

Input: CNF-formula F

Output: Number of satisfying assignments of F

- generic #P-hard counting problem
- applications, e.g.
  - probabilistic graphical models
  - stochastic planning

On the Hardness of $\#SAT$

- state-of-the-art practical solvers for $\#SAT$ far slower than SAT-solvers

On the Hardness of $\#SAT$

- state-of-the-art practical solvers for $\#SAT$ far slower than SAT-solvers

Toda's Theorem

$$\mathsf{PH} \subseteq \mathsf{P}^{\#SAT[1]}.$$

- in words: $\#SAT$ at least as hard as all problem in polynomial hierarchy

On the Hardness of $\#SAT$

- state-of-the-art practical solvers for $\#SAT$ far slower than SAT-solvers

Toda's Theorem

$$\mathrm{PH} \subseteq \mathrm{P}^{\#SAT[1]}.$$

- in words: $\#SAT$ at least as hard as all problem in polynomial hierarchy
- $2^{n^{1-\epsilon}}$-approximation hard for every $\epsilon > 0$

On the Hardness of $\#SAT$

- state-of-the-art practical solvers for $\#SAT$ far slower than SAT-solvers

Toda's Theorem

$$\mathsf{PH} \subseteq \mathsf{P}^{\#SAT[1]}.$$

- in words: $\#SAT$ at least as hard as all problem in polynomial hierarchy
- $2^{n^{1-\epsilon}}$-approximation hard for every $\epsilon > 0$

$\Rightarrow$ extremely hard problem (theory and practice)

Interaction Between Variables: Primal Graphs

Interaction Between Variables: Primal Graphs

4

- variables are vertices
- edge between variables that share clauses

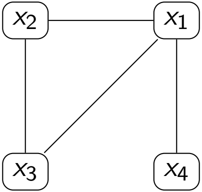
Interaction Between Variables: Primal Graphs

4

- variables are vertices
- edge between variables that share clauses

Example

- $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (x_1)$



Example: #SAT on trees

5

Observation

Tree-shaped #SAT is tractable.

Example: $\#SAT$ on trees

5

Observation

Tree-shaped $\#SAT$ is tractable.

Proof:

- graph of input F is a tree $\rightarrow$ $\#2\text{-}SAT$

Example: $\#SAT$ on trees

5

Observation

Tree-shaped $\#SAT$ is tractable.

Proof:

- graph of input F is a tree $\rightarrow$ $\#2\text{-}SAT$
- idea: dynamic programming from leaves to (arbitrarily chosen) root

Example: $\#SAT$ on trees

5

Observation

Tree-shaped $\#SAT$ is tractable.

Proof:

- graph of input F is a tree $\rightarrow$ $\#2$-SAT
- idea: dynamic programming from leaves to (arbitrarily chosen) root
- compute $S(v, i)$ for each vertex v and $i \in \{0, 1\}$:
  $S(v, i)$: number of solutions of formula rooted in v in which v takes value i

Example: $\#SAT$ on trees

5

Observation

Tree-shaped $\#SAT$ is tractable.

Proof:

- graph of input F is a tree $\rightarrow$ $\#2\text{-}SAT$
- idea: dynamic programming from leaves to (arbitrarily chosen) root
- compute $S(v, i)$ for each vertex v and $i \in \{0, 1\}$:
  $S(v, i)$: number of solutions of formula rooted in v in which v takes value i
- correct output is $S(r, 0) + S(r, 1)$ where r is root

Example: $\#SAT$ on trees

5

Observation

Tree-shaped $\#SAT$ is tractable.

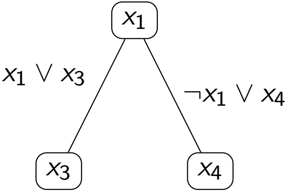
Proof:

- graph of input F is a tree $\rightarrow$ $\#2\text{-}SAT$
- idea: dynamic programming from leaves to (arbitrarily chosen) root
- compute $S(v, i)$ for each vertex v and $i \in \{0, 1\}$:
  $S(v, i)$: number of solutions of formula rooted in v in which v takes value i
- correct output is $S(r, 0) + S(r, 1)$ where r is root
- for every leaf v:

$$S(v, 0) = S(v, 1) = 1$$

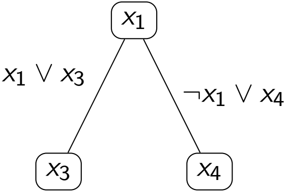
$\#SAT$ on trees, example

6



- already computed: $S(x_3, i), S(x_4, i)$

$\#SAT$ on trees, example

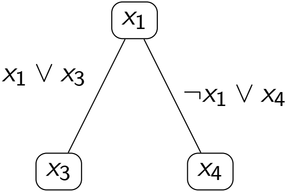


- already computed: $S(x_3, i), S(x_4, i)$

$$S(x_1, 0) = S(x_3, 1) \cdot \big(S(x_4, 0) + S(x_4, 1)\big)$$
$$S(x_1, 1) = \big(S(x_3, 0) + S(x_3, 1)\big) \cdot S(x_4, 1)$$

#SAT on trees, example

6



- already computed: $S(x_3, i), S(x_4, i)$

$$S(x_1, 0) = S(x_3, 1) \cdot \big(S(x_4, 0) + S(x_4, 1)\big)$$
$$S(x_1, 1) = \big(S(x_3, 0) + S(x_3, 1)\big) \cdot S(x_4, 1)$$

also works in general

Structural tractability of $\#SAT$

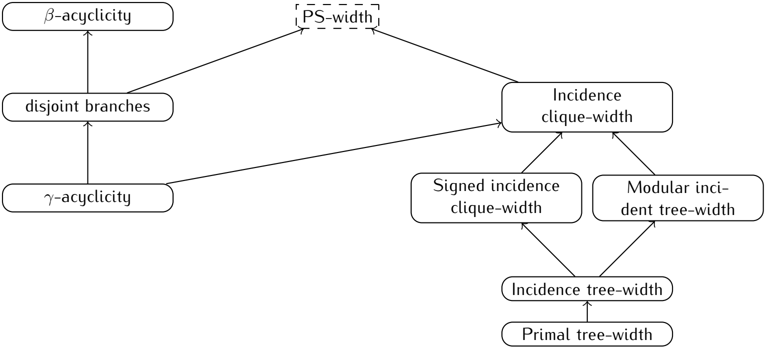
7

Which other graphs give tractable instances of $\#SAT$?

UNIVERSITÉ D'ARTOIS

Structural tractability of #*SAT*

Which other graphs give tractable instances of #*SAT*?



Parameter of the hypergraph **Parameter of the incidence graph**

- all algorithms dynamic programming

(Samer, Szeider '10; Paulusma, Slivovsky, Szeider '13; Fisher, Makowsky, Ravve '08; Slivovsky, Szeider '14; Capelli, Durand, M '14; Brault-Baron, Capelli, M '15; Sæther, Telle, Vatshelle '14)

Treewidth vs. Cliquewidth

8

- runtimes for formula F
  - treewidth k FPT: $2^k|F|$
  - cliquewidth k not FPT: $|F|^k$
- FPT runtime far preferable

Treewidth vs. Cliquewidth

8

- runtimes for formula F
  - treewidth k FPT: $2^k|F|$
  - cliquewidth k not FPT: $|F|^k$
- FPT runtime far preferable

Question

Can runtime for cliquewidth be improved to FPT?

Treewidth vs. Cliquewidth

- runtimes for formula F
  - treewidth k FPT: $2^k|F|$
  - cliquewidth k not FPT: $|F|^k$
- FPT runtime far preferable

Question

Can runtime for cliquewidth be improved to FPT?

- NO, unless strange things happen, i.e. FPT = W[1] (Paulusma, Slivovsky, Szeider '13)
- but can we say something unconditionally?

Exhaustive DPLL and its traces

9

- Exhaustive DPLL
  - backtracking search to find/count solutions of CNF
  - caching (reuse counts of subformulas already solved)
  - components (independent formulas treated independently)
- basis of state of the art model counters

Exhaustive DPLL and its traces

9

- Exhaustive DPLL
  - backtracking search to find/count solutions of CNF
  - caching (reuse counts of subformulas already solved)
  - components (independent formulas treated independently)
- basis of state of the art model counters

Theorem (Huang, Darwiche '05)

Traces of runs of exhaustive DPLL are DNNF.

- DNNF
  - restricted boolean circuits
  - extensively studied in knowledge compilation
- so model counters can also compile: sharpSAT $\rightsquigarrow$ Dsharp

Compilation and Traces of the Algorithms

10

Theorem (Bova, Capelli, M, Slivovsky '15, Capelli '16)

Consider any of the known graph based model counting algorithms.
Then its traces are also DNNF.

- consequence: essentially all known counting algorithms lead to DNNF (except linear algebra...)

Theorem (Bova, Capelli, M, Slivovsky '15, Capelli '16)

Consider any of the known graph based model counting algorithms.
Then its traces are also DNNF.

- consequence: essentially all known counting algorithms lead to DNNF (except linear algebra...)
- to show that known counting algorithms do not solve $\#SAT$, "only" need to show DNNF lower bounds

Theorem (Bova, Capelli, M, Slivovsky '15, Capelli '16)

Consider any of the known graph based model counting algorithms.
Then its traces are also DNNF.

- consequence: essentially all known counting algorithms lead to DNNF (except linear algebra...)
- to show that known counting algorithms do not solve $\#SAT$, "only" need to show DNNF lower bounds
- ...but that was an open problem for 15 years

Lower Bounds for DNNF

11

Connection (Bova, Capelli, M, Slivovsky '16)

Functions with high multipartition communication complexity have no small DNNF.

Lower Bounds for DNNF

11

Connection (Bova, Capelli, M, Slivovsky '16)

Functions with high multipartition communication complexity have no small DNNF.

Theorem (essentially Duris et al. '04)

There are CNF with high multipartition communication complexity.

- lets us show lower bounds for DNNF and thus for counting "easily"

UNIVERSITÉ D'ARTOIS

Lower Bounds for DNNF

11

Connection (Bova, Capelli, M, Slivovsky '16)

Functions with high multipartition communication complexity have no small DNNF.

Theorem (essentially Duris et al. '04)

There are CNF with high multipartition communication complexity.

- lets us show lower bounds for DNNF and thus for counting "easily"

Theorem (roughly, M '16)

Cliquewidth k formulas generally require DNNF size n^k
Thus known counting algorithms require time n^k.

Conclusion

12

- model counting is hard
- graph restrictions help to a certain extent
- connection to compilation (DNNF)
- lower bounds with communication complexity
- cliquewidth harder than treewidth

Model Counting in Practice

- Many model counters (both exact and approximate) have been developed so far
  - **search-based model counters**: `Cachet`, `SharpSAT`, etc.,
  - **compilation-based:** `C2D`, `SDD`, etc.

Model Counting in Practice

13

- Many model counters (both exact and approximate) have been developed so far
  - **search-based model counters**: `Cachet`, `SharpSAT`, etc.,
  - **compilation-based:** `C2D`, `SDD`, etc.

- **Objective at CRIL:** Improving model counters by preprocessing the input (a `CNF` formula)

UNIVERSITÉ D'ARTOIS

Off-line approaches for improving the model counting task

- **knowledge compilation**

$\texttt{CNF}\ \Sigma \longrightarrow \boxed{\text{compilation}} \longrightarrow \texttt{ADT}\ \Psi \longrightarrow \boxed{\text{model counting}} \longrightarrow \|\Sigma\|$

[IJCAI'13 - joint work with F. Koriche, J.-M. Lagniez and S. Thomas]

UNIVERSITÉ D'ARTOIS

Off-line approaches for improving the model counting task

- **knowledge compilation**

  $\texttt{CNF } \Sigma \longrightarrow \boxed{\text{compilation}} \longrightarrow \texttt{ADT } \Psi \longrightarrow \boxed{\text{model counting}} \longrightarrow \|\Sigma\|$

  [IJCAI'13 – joint work with F. Koriche, J.-M. Lagniez and S. Thomas]

- **preprocessing**

  $\texttt{CNF } \Sigma \longrightarrow \boxed{\text{preprocessing}} \longrightarrow \texttt{CNF } \Phi \longrightarrow \boxed{\text{model counting}} \longrightarrow \|\Sigma\|$

  [AAAI'14 – joint work with J.-M. Lagniez]
  [IJCAI'16 – joint work with J.-M. Lagniez and E. Lonca]

Gate Identification and Replacement

15

Gate identification and replacement is a valuable preprocessing for model counting [AAAI'14 - joint work with J.-M. Lagniez]

$$\Sigma = \begin{array}{l} \overline{x} \vee u \vee v \\ \overline{x} \vee \overline{y} \vee u \\ \overline{x} \vee \overline{z} \vee u \\ x \vee \overline{u} \\ y \vee z \vee \overline{u} \end{array}$$

15

Gate identification and replacement is a valuable preprocessing for model counting [AAAI'14 - joint work with J.-M. Lagniez]

$$\Sigma = \begin{array}{l} \overline{x} \lor u \lor v \\ \overline{x} \lor \overline{y} \lor u \\ \overline{x} \lor \overline{z} \lor u \\ x \lor \overline{u} \\ y \lor z \lor \overline{u} \end{array} \qquad u \leftrightarrow (x \land (y \lor z))$$

Gate Identification and Replacement

15

Gate identification and replacement is a valuable preprocessing for model counting [AAAI'14 – joint work with J.-M. Lagniez]

$$\Sigma = \begin{array}{l} \overline{x} \vee u \vee v \\ \overline{x} \vee \overline{y} \vee u \\ \overline{x} \vee \overline{z} \vee u \\ x \vee \overline{u} \\ y \vee z \vee \overline{u} \end{array} \qquad u \leftrightarrow (x \wedge (y \vee z))$$

$$(\overline{x} \vee u \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z)))$$

identification

$$\Sigma \equiv$$

Gate Identification and Replacement

15

Gate identification and replacement is a valuable preprocessing for model counting [AAAI'14 – joint work with J.-M. Lagniez]

$$\Sigma = \begin{array}{l} \overline{x} \vee u \vee v \\ \overline{x} \vee \overline{y} \vee u \\ \overline{x} \vee \overline{z} \vee u \\ x \vee \overline{u} \\ y \vee z \vee \overline{u} \end{array} \qquad u \leftrightarrow (x \wedge (y \vee z))$$

$$\Sigma \equiv \begin{array}{ll} (\overline{x} \vee u \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z))) & \text{identification} \\ (\overline{x} \vee (x \wedge (y \vee z)) \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z))) & \text{replacement} \end{array}$$

Gate Identification and Replacement

15

Gate identification and replacement is a valuable preprocessing for model counting [AAAI'14 – joint work with J.-M. Lagniez]

$$\Sigma = \begin{array}{l} \overline{x} \vee u \vee v \\ \overline{x} \vee \overline{y} \vee u \\ \overline{x} \vee \overline{z} \vee u \\ x \vee \overline{u} \\ y \vee z \vee \overline{u} \end{array} \qquad u \leftrightarrow (x \wedge (y \vee z))$$

$$\Sigma \equiv \begin{array}{ll} (\overline{x} \vee u \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z))) & \text{identification} \\ (\overline{x} \vee (x \wedge (y \vee z)) \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z))) & \text{replacement} \\ (\overline{x} \vee y \vee z \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z))) & \text{normalization} \end{array}$$

UNIVERSITÉ D'ARTOIS

Gate Identification and Replacement

15

Gate identification and replacement is a valuable preprocessing for model counting [AAAI'14 - joint work with J.-M. Lagniez]

$$\Sigma = \begin{array}{l} \overline{x} \vee u \vee v \\ \overline{x} \vee \overline{y} \vee u \\ \overline{x} \vee \overline{z} \vee u \\ x \vee \overline{u} \\ y \vee z \vee \overline{u} \end{array} \qquad u \leftrightarrow (x \wedge (y \vee z))$$

$$\Sigma \equiv \begin{array}{ll} (\overline{x} \vee u \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z))) & \text{identification} \\ (\overline{x} \vee (x \wedge (y \vee z)) \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z))) & \text{replacement} \\ (\overline{x} \vee y \vee z \vee v) \wedge (u \leftrightarrow (x \wedge (y \vee z))) & \text{normalization} \end{array}$$

$$\|\Sigma\| = \|\Sigma[u \leftarrow (x \wedge (y \vee z))]\| = \|\overline{x} \vee y \vee z \vee v\| = 15$$

UNIVERSITÉ D'ARTOIS

Limitations

16

- The replacement phase **requires gates to be identified**
  - The search space for gates is **huge**: if $|Var(\Sigma)| = n$, then for each variable x of Σ, there exist
    $$2^{2^{n-1}} \text{ gates}$$
    with output x
  - The size of a gate can be **huge** as well: not polynomially bounded in $|\Sigma| + |X|$ unless $\text{NP} \cap \text{coNP} \subseteq \text{P/poly}$

Limitations

- The replacement phase **requires gates to be identified**
  - The search space for gates is **huge:** if $|Var(\Sigma)| = n$, then for each variable x of Σ, there exist

$$2^{2^{n-1}} \text{ gates}$$

    with output x
  - The size of a gate can be **huge** as well: not polynomially bounded in $|\Sigma| + |X|$ unless NP $\cap$ coNP $\subseteq$ P/poly

- Identifying "complex gates" is incompatible with the efficiency expected for a preprocessing:
  only "simple" gates are targeted

literal equivalences	$y \leftrightarrow x_1$
AND/OR gates	$y \leftrightarrow (x_1 \wedge \overline{x_2} \wedge x_3)$
XOR gates	$y \leftrightarrow (x_1 \oplus \overline{x_2})$

Overcoming the Limitations

- The (explicit) identification phase can be replaced by an **implicit identification phase**
- There is **no need to identify** f to determine that a gate of the form $y \leftrightarrow f(x_1, \ldots, x_n)$ exists in Σ

Overcoming the Limitations

17

- The (explicit) identification phase can be replaced by an **implicit identification phase**
- There is **no need to identify** f to determine that a gate of the form $y \leftrightarrow f(x_1, \ldots, x_n)$ exists in Σ
- The replacement phase can be replaced by an **output variable elimination phase**
- There is **no need to identify** f to compute $\Sigma[y \leftarrow f(x_1, \ldots, x_n)]$

Evert Willem Beth (1908–1964)

Evert Willem Beth (1908-1964)

- Σ **explicitly defines** y in terms of $X = \{x_1, \ldots, x_n\}$ iff there exists a formula $f(x_1, \ldots, x_n)$ over X such that

$$\Sigma \models y \leftrightarrow f(x_1, \ldots, x_n)$$

Evert Willem Beth (1908-1964)

18

- Σ **explicitly defines** y in terms of $X = \{x_1, \ldots, x_n\}$ iff there exists a formula $f(x_1, \ldots, x_n)$ over X such that

$$\Sigma \models y \leftrightarrow f(x_1, \ldots, x_n)$$

- Σ **implicitly defines** y in terms of $X = \{x_1, \ldots, x_n\}$ iff for every canonical term γ_X over X, we have $\Sigma \wedge \gamma_X \models y$ or $\Sigma \wedge \gamma_X \models \overline{y}$

Evert Willem Beth (1908–1964)

- Σ **explicitly defines** y in terms of $X = \{x_1, \ldots, x_n\}$ iff there exists a formula $f(x_1, \ldots, x_n)$ over X such that

$$\Sigma \models y \leftrightarrow f(x_1, \ldots, x_n)$$

- Σ **implicitly defines** y in terms of $X = \{x_1, \ldots, x_n\}$ iff for every canonical term γ_X over X, we have $\Sigma \wedge \gamma_X \models y$ or $\Sigma \wedge \gamma_X \models \overline{y}$

- Beth's theorem: Σ **explicitly defines** y in terms of X iff Σ **implicitly defines** y in terms of X

Padoa's theorem:

Let Σ'_X be equal to Σ where each variable but those of X have been renamed in a uniform way

If $y \not\in X$, then Σ (implicitly) defines y in terms of X iff $\Sigma \wedge \Sigma'_X \wedge y \wedge \overline{y'}$ is inconsistent

Alessandro Padoa (1868–1937)

Padoa's theorem:

Let Σ'_X be equal to Σ where each variable but those of X have been renamed in a uniform way
If $y \notin X$, then Σ (implicitly) defines y in terms of X iff $\Sigma \wedge \Sigma'_X \wedge y \wedge \overline{y'}$ is inconsistent

Deciding whether Σ (implicitly) defines y in terms of X is "only" coNP-complete

From Replacement to Variable Elimination

20

The replacement phase can be replaced by an **output variable elimination phase:** if $y \leftrightarrow f(x_1, \ldots, x_n)$ is a gate of Σ, then

$$\Sigma[y \leftarrow f(x_1, \ldots, x_n)] \equiv \exists y.\Sigma$$

The $\mathtt{B}+\mathtt{E}$ Preprocessor

A two-step preprocessing

- "Identification = Bipartition":
  compute a **definability bipartition** $\langle I, O \rangle$ of Σ such that $I \cup O = Var(\Sigma)$, $I \cap O = \emptyset$, and Σ defines every variable $o \in O$ in terms of I

The B + E Preprocessor

21

A two-step preprocessing

- "Identification = Bipartition":
  compute a **definability bipartition** $\langle I, O \rangle$ of Σ such that $I \cup O = Var(\Sigma)$, $I \cap O = \emptyset$, and Σ defines every variable $o \in O$ in terms of I
- "Replacement = Elimination":
  compute $\exists E.\Sigma$ for $E \subseteq O$

UNIVERSITÉ D'ARTOIS

Identifying u as an Output Variable and Eliminating it

Identification:

$\Sigma \wedge \Sigma'_{\{x,y,z\}} \wedge u \wedge \overline{u'}$ is inconsistent

$\overline{x} \vee u \vee v$

$\overline{x} \vee \overline{y} \vee u$

$\overline{x} \vee \overline{z} \vee u$

$x \vee \overline{u}$

$y \vee z \vee \overline{u}$

$\overline{x} \vee u' \vee v'$

$\overline{x} \vee \overline{y} \vee u'$

$\overline{x} \vee \overline{z} \vee u'$

$x \vee \overline{u'}$

$y \vee z \vee \overline{u'}$

u

$\overline{u'}$

Identifying u as an Output Variable and Eliminating it

22

Identification:

$\Sigma \wedge \Sigma'_{\{x,y,z\}} \wedge u \wedge \overline{u'}$ is inconsistent

$\overline{x} \vee u \vee v$
$\overline{x} \vee \overline{y} \vee u$
$\overline{x} \vee \overline{z} \vee u$
$x \vee \overline{u}$
$y \vee z \vee \overline{u}$
$\overline{x} \vee u' \vee v'$
$\overline{x} \vee \overline{y} \vee u'$
$\overline{x} \vee \overline{z} \vee u'$
$x \vee \overline{u'}$
$y \vee z \vee \overline{u'}$
u
$\overline{u'}$

Elimination:

computing resolvents over u

$\overline{x} \vee v \vee x$	valid
$\overline{x} \vee v \vee y \vee z$	
$\overline{x} \vee \overline{y} \vee x$	valid
$\overline{x} \vee \overline{y} \vee y \vee z$	valid
$\overline{x} \vee \overline{z} \vee x$	valid
$\overline{x} \vee \overline{z} \vee y \vee z$	valid

Identifying u as an Output Variable and Eliminating it

22

Identification:

$\Sigma \wedge \Sigma'_{\{x,y,z\}} \wedge u \wedge \overline{u'}$ is inconsistent

$\overline{x} \vee u \vee v$
$\overline{x} \vee \overline{y} \vee u$
$\overline{x} \vee \overline{z} \vee u$
$x \vee \overline{u}$
$y \vee z \vee \overline{u}$
$\overline{x} \vee u' \vee v'$
$\overline{x} \vee \overline{y} \vee u'$
$\overline{x} \vee \overline{z} \vee u'$
$x \vee \overline{u'}$
$y \vee z \vee \overline{u'}$
u
$\overline{u'}$

Elimination:

computing resolvents over u

$\overline{x} \vee v \vee x$	valid
$\overline{x} \vee v \vee y \vee z$	
$\overline{x} \vee \overline{y} \vee x$	valid
$\overline{x} \vee \overline{y} \vee y \vee z$	valid
$\overline{x} \vee \overline{z} \vee x$	valid
$\overline{x} \vee \overline{z} \vee y \vee z$	valid

$\|\Sigma\| = \|\overline{x} \vee v \vee y \vee z\| = 15$

Tuning the Computational Effort

Both steps B and E of B + E can be tuned in order to keep the preprocessing phase **light from a computational standpoint**

- It is not necessary to determine a definability bipartition $\langle I, O \rangle$ with $|I|$ minimal
  $\Rightarrow$ B is a **greedy algorithm** (one definability test per variable)
  $\Rightarrow$ Only the minimality of I for $\subseteq$ is guaranteed

Tuning the Computational Effort

Both steps $\texttt{B}$ and $\texttt{E}$ of $\texttt{B} + \texttt{E}$ can be tuned in order to keep the preprocessing phase **light from a computational standpoint**

- It is not necessary to determine a definability bipartition $\langle I, O\rangle$ with $|I|$ minimal
  $\Rightarrow$ $\texttt{B}$ is a **greedy algorithm** (one definability test per variable)
  $\Rightarrow$ Only the minimality of I for $\subseteq$ is guaranteed

- It is not necessary to eliminate in Σ every variable of O but focusing on a subset $E \subseteq O$ is enough
  $\Rightarrow$ Eliminating every output variable could lead to an **exponential blow up**
  $\Rightarrow$ The elimination of $y \in O$ is committed only if $|\Sigma|$ after the elimination step and some additional preprocessing (occurrence simplification and vivification) remains **small enough**

Experiments

Objectives:

- Evaluating the computational benefits offered by $\mathtt{B} + \mathtt{E}$ when used upstream to state-of-the-art model counters:
  - the search-based model counter `Cachet`
  - the search-based model counter `SharpSAT`
  - the compilation-based model counter `C2D` (used with `-count -in_memory -smooth_all`)

Objectives:

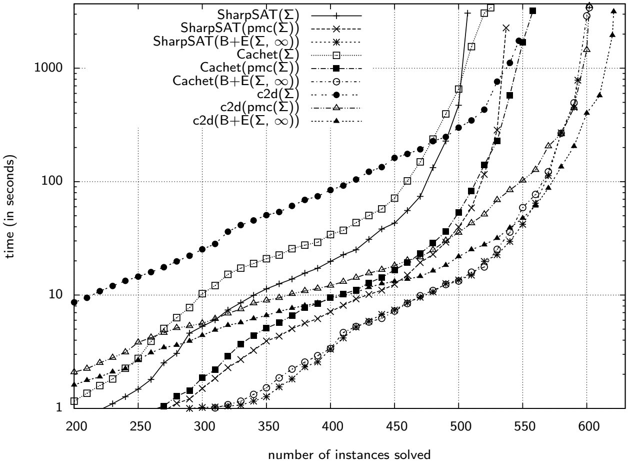
- Evaluating the computational benefits offered by $\mathtt{B}+\mathtt{E}$ when used upstream to state-of-the-art model counters:
  - the search-based model counter `Cachet`
  - the search-based model counter `SharpSAT`
  - the compilation-based model counter `C2D` (used with `-count -in_memory -smooth_all`)
- Comparing the benefits offered by $\mathtt{B}+\mathtt{E}$ with those offered by our previous preprocessor `pmc` (based on gate identification and replacement) or with no preprocessing

Empirical Setting

- 703 `CNF` instances from the SAT LIBrary
- 8 data sets: BN (Bayesian networks) (192), BMC (Bounded Model Checking) (18), Circuit (41), Configuration (35), Handmade (58), Planning (248), Random (104), Qif (7) (Quantitative Information Flow analysis - security)
- Cluster of Intel Xeon E5-2643 (3.30 GHz) processors with 32 GiB RAM on Linux CentOS
- Time-out =1h
- Memory-out = 7.6 GiB

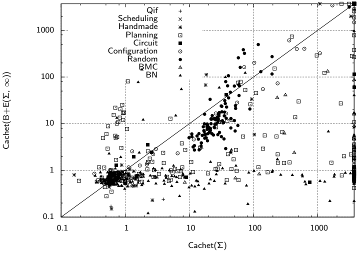
Empirical Results

26

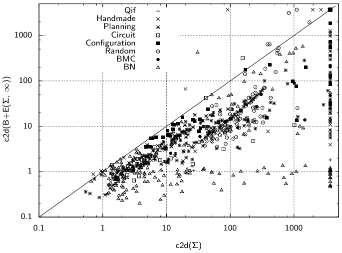


Empirical Results

B + E vs. no preprocessing



(a) B + E+Cachet vs. Cachet

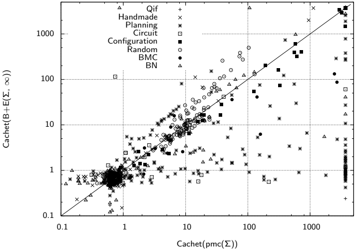


(b) B + E+C2D vs. C2D

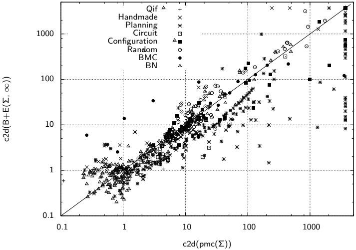
Empirical Results

28

B + E vs. pmc



(c) B + E+Cachet vs. pmc+Cachet



(d) B + E+C2D vs. pmc+C2D

UNIVERSITÉ D'ARTOIS

Conclusion

- Design and implementation of the $\texttt{B} + \texttt{E}$ preprocessor
- Empirical evaluation of $\texttt{B} + \texttt{E}$: for several model counters $\texttt{mc}$, $\texttt{mc}(\texttt{B} + \texttt{E}(.))$ proves computationally more efficient than $\texttt{mc}(.)$

Conclusion and Perspectives

Conclusion

- Design and implementation of the $\texttt{B} + \texttt{E}$ preprocessor
- Empirical evaluation of $\texttt{B} + \texttt{E}$: for several model counters $\texttt{mc}$, $\texttt{mc}(\texttt{B} + \texttt{E}(.))$ proves computationally more efficient than $\texttt{mc}(.)$

Perspectives

- Developing other ordering heuristics for $\texttt{B}$
- Connection to **projected model counting:** computing $\|\exists E.\Sigma\|$ given a set E of variables and a formula Σ
  - compute $\|\Sigma\|$ as $\exists mc(\Sigma, O)$ where $B(\Sigma) = \langle I, O \rangle$
  - compute $\exists mc(\Sigma, E)$ as $\texttt{mc}(\texttt{E}(\Sigma))$