Projected d-DNNF Compilation for Feature Models

1. **What is the problem?**
   - Feature Models and Conjunctive Normal Forms
   - d-DNNF
   - Feature-Model Slicing / Projection of Formulas
   - Scalability Issues with State-of-the-Art Reasoning
2. **Why should you care about this thesis?**
   - Is it novel?
   - Is it significant?
   - Is it sound?
   - Is it verifiable?
   - Is it clear?
3. **More Details Wanted?**
   - Model Counting
   - Projected Model Counting
   - d-DNNF Compilation
   - Heuristics
   - Heuristics: Dual Hypergraph
   - Integration and Optimization in D4
   - Dual Weighted Hypergraph Partitioning
   - Partial Resolution
   - Generated Projection for Feature Models (Table)
   - The End

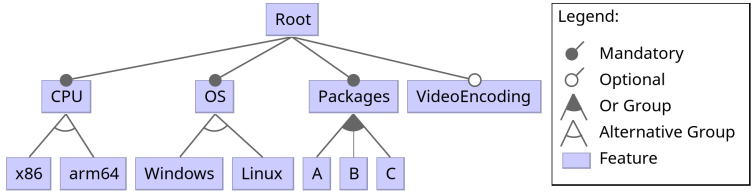
Projected d-DNNF Compilation for Feature Models

Master's Thesis by Jacob Loth, supervised by Chico Sundermann | Thomas Thüm | April 9, 2024

PADERBORN UNIVERSITY

1. What is the problem?

Feature Models and Conjunctive Normal Forms



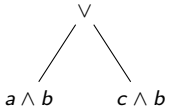
We can convert feature models to propositional formulas!

$$(x86 \wedge \neg arm64) \vee (\neg x86 \wedge arm64)...$$

Deterministic Decomposable Negation Normal Form (d-DNNF)

Any propositional formula which is:

deterministic
Exclusive or-operators $F = A \vee B$
Never simultaneous $A = 1$ and $B = 1$
If-then-else
$|F| = |A| + |B|$

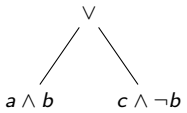


Not a d-DNNF ✗

Deterministic Decomposable Negation Normal Form (d-DNNF)

Any propositional formula which is:

deterministic
Exclusive or-operators $F = A \vee B$
Never simultaneous $A = 1$ and $B = 1$
If-then-else
$|F| = |A| + |B|$



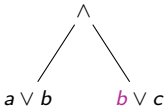
A d-DNNF ✓

Deterministic Decomposable Negation Normal Form (d-DNNF)

Any propositional formula which is:

deterministic
Exclusive or-operators $F = A \vee B$
Never simultaneous $A = 1$ and $B = 1$
If-then-else
$|F| = |A| + |B|$

Decomposable
And-operands $F = A \wedge B$ never share variables
$|F| = |A| * |B|$



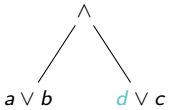
Not a d-DNNF ✗

Deterministic Decomposable Negation Normal Form (d-DNNF)

Any propositional formula which is:

deterministic
Exclusive or-operators $F = A \vee B$
Never simultaneous $A = 1$ and $B = 1$
If-then-else
$|F| = |A| + |B|$

Decomposable
And-operands $F = A \wedge B$ never share variables
$|F| = |A| * |B|$



A d-DNNF ✓

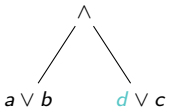
Deterministic Decomposable Negation Normal Form (d-DNNF)

Any propositional formula which is:

deterministic
Exclusive or-operators $F = A \vee B$
Never simultaneous $A = 1$ and $B = 1$
If-then-else
$|F| = |A| + |B|$

Decomposable
And-operands $F = A \wedge B$ never share variables
$|F| = |A| * |B|$

Negation **N**ormal **F**orm



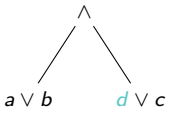
Deterministic Decomposable Negation Normal Form (d-DNNF)

Any propositional formula which is:

deterministic
Exclusive or-operators $F = A \vee B$
Never simultaneous $A = 1$ and $B = 1$
If-then-else
$|F| = |A| + |B|$

Decomposable
And-operands $F = A \wedge B$ never share variables
$|F| = |A| * |B|$

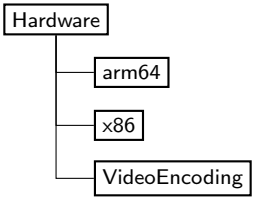
Negation **N**ormal **F**orm

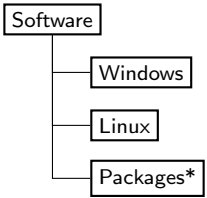


d-DNNF formulas allow linear-time model counting
d-DNNF compilation: CNF → d-DNNF

Feature-Model Slicing / Projection of Formulas

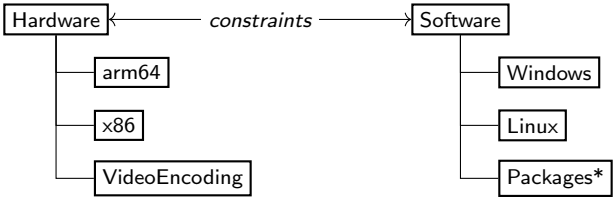
Feature Model





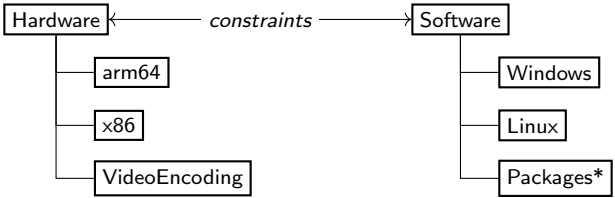
Feature-Model Slicing / Projection of Formulas

Feature Model



Feature-Model Slicing / Projection of Formulas

Feature Model

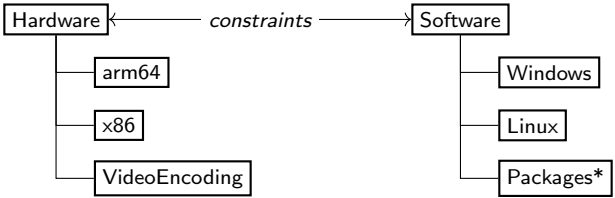


Problem

How many hardware configurations?

Feature-Model Slicing / Projection of Formulas

Feature Model



Problem

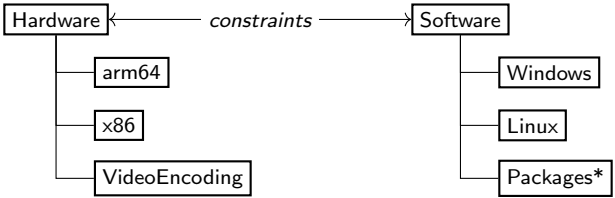
How many hardware configurations?

Transitive Constraints

$$\text{VideoEncoding} \implies \text{Windows}$$
$$\text{Windows} \implies \text{x86}$$

Feature-Model Slicing / Projection of Formulas

Feature Model



Problem

How many hardware configurations?

Transitive Constraints

VideoEncoding $\Longrightarrow$ Windows

Windows $\Longrightarrow$ x86

Sliced: VideoEncoding $\Longrightarrow$ x86

Feature-Model Slicing / Projection of Formulas

[Krieter et al. SPLC16]

Resolve all clauses with v with all clauses with $\neg v$

Resolving Two Clauses

$$(\neg \mathit{VideoEncoding} \vee \mathit{Windows}), (x86 \vee \neg \mathit{Windows}) \rightarrow (\neg \mathit{VideoEncoding} \vee x86)$$

Feature-Model Slicing / Projection of Formulas

[Krieter et al. SPLC16]

Resolve all clauses with v with all clauses with $\neg v$

Resolving Two Clauses

$$(\neg \mathit{VideoEncoding} \vee \mathit{Windows}), (x86 \vee \neg \mathit{Windows}) \rightarrow (\neg \mathit{VideoEncoding} \vee x86)$$

$$(a_1 \vee a_2 \vee ... \vee v), (b_1 \vee b_2 \vee ... \vee \neg v) \rightarrow (a_1 \vee a_2 \vee ... \vee b_1 \vee b_2 \vee ...)$$

Feature-Model Slicing / Projection of Formulas

[Krieter et al. SPLC16]

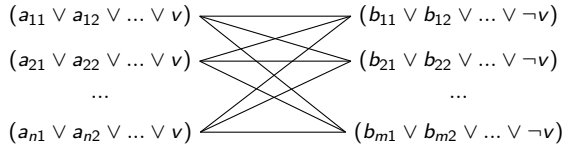
Resolve all clauses with v with all clauses with $\neg v$

Resolving Two Clauses

$$(\neg \mathit{VideoEncoding} \vee \mathit{Windows}), (x86 \vee \neg \mathit{Windows}) \rightarrow (\neg \mathit{VideoEncoding} \vee x86)$$

$$(a_1 \vee a_2 \vee ... \vee v), (b_1 \vee b_2 \vee ... \vee \neg v) \rightarrow (a_1 \vee a_2 \vee ... \vee b_1 \vee b_2 \vee ...)$$

Resolving Many Clauses



Exponential clause count increase for multiple variables.

Scalability Issues with State-of-the-Art Reasoning

2. Why should you care about this thesis?

Why should you care about this thesis?

Standard Evaluation Criteria

1. Novelty
2. Significance
3. Soundness
4. Verifyability
5. Clarity

Let me try to convince you . . .

Is it novel?

Projection + Compilation = Projected Compilation

Is it novel? Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables $= a, b, d$

Sliced variables $= c, e$

$$(a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e)$$

Is it novel? Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e

$$(a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e)$$

$$\xrightarrow{\;a\;}$$

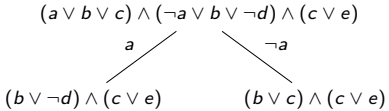
$$(b \vee \neg d) \wedge (c \vee e)$$

Is it novel? Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e

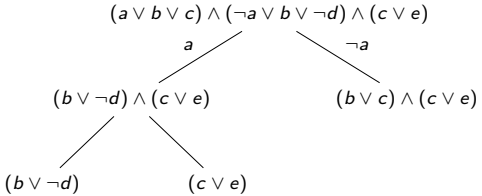


Is it novel? Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e

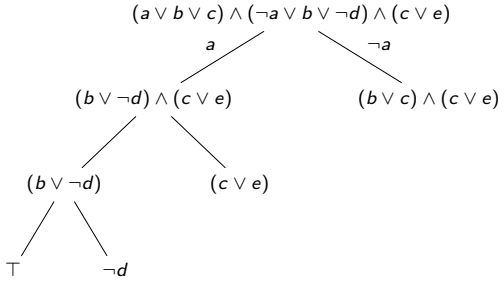


Is it novel? Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e

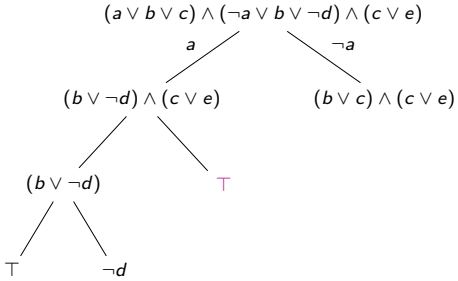


Is it novel? Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e

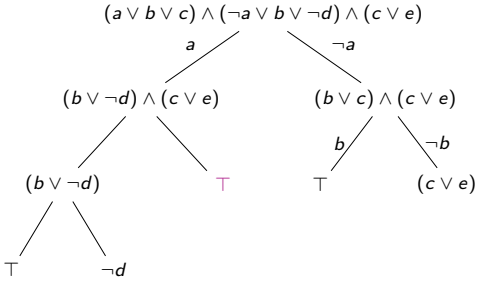


Is it novel? Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e

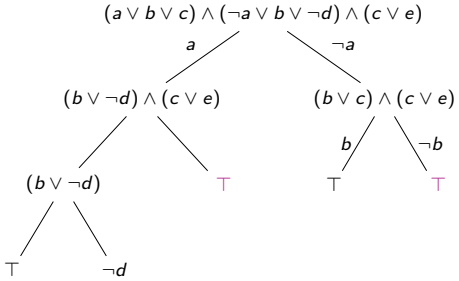


Is it novel? Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e



Is it significant?

Experimental Design

Solvers

- **pD4**: Our approach
- **slice**: Slicing followed by d-DNNF compilation
- **gpmc**: 1st place projected model counter in MC2022
- **D4-pmc**: 2nd place projected model counter in MC2022
- **arjun**: 3rd place projected model counter in MC2022

Data

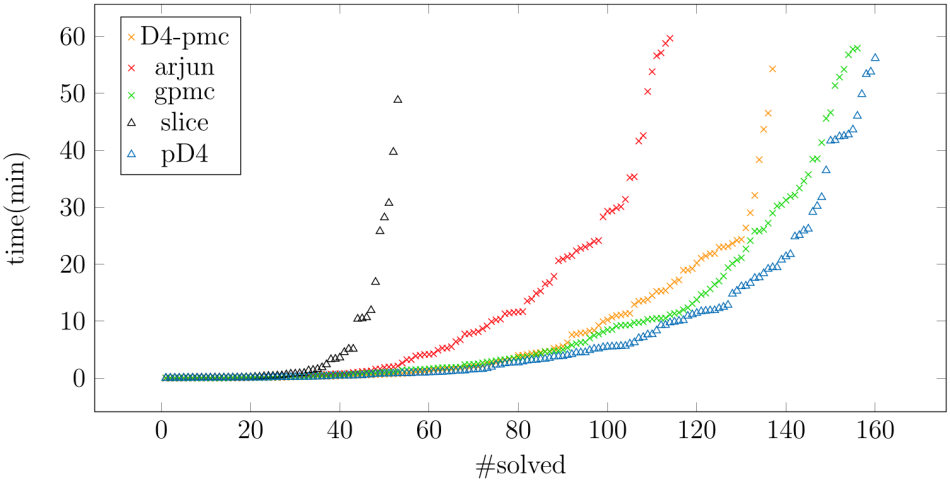
- **Industrial Projection**: Real feature model slicing problems
- **Generated Projection**: Adding randomly selected projected variables to real feature models
- **MC2022**: Private+public instances from the MC2022 (many unknown sources...)

Questions

Compare runtime performance (and d-DNNF size)

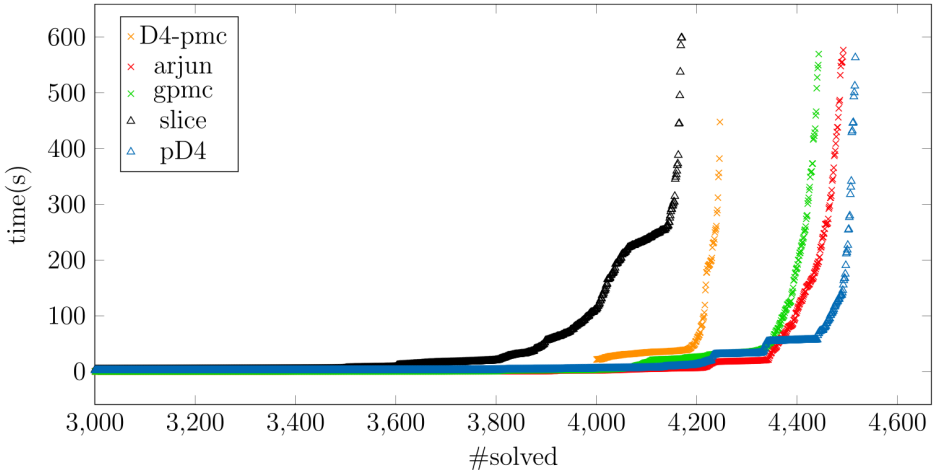
Is it significant?

Projected Model Counting Competition 2022



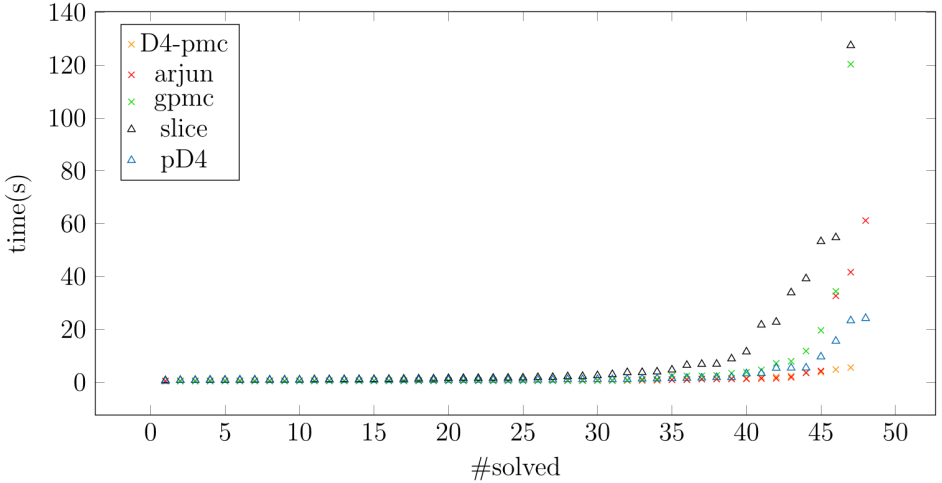
Is it sound?

Random Projections on Feature Models



Is it sound?

Real Projections from Automotive Industry



Is it verifiable?

pD4 on Github

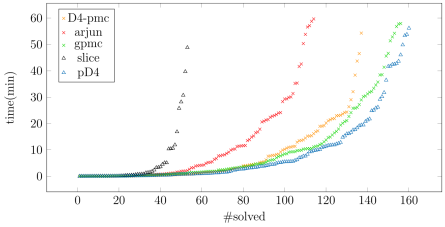
Replication Package

Is it clear? Projected d-DNNF Compilation for Feature Models

Projected Compilation = Projection + Compilation

projected d-DNNF compilation ...

- is faster than projection / slicing (our goal)
- slicing Linux feasible if slice small enough
- is faster than projected model counting (unexpected, due to many optimizations)
- is much faster for multiple queries (expected)
- produces d-DNNFs of comparable/smaller size
- is the first instance of projected compilation



Projected d-DNNF Compilation for Feature Models

Master's Thesis by Jacob Loth, supervised by Chico Sundermann | Thomas Thüm | April 9, 2024

PADERBORN UNIVERSITY

3. More Details Wanted?

Model Counting

Problem

- How many hardware configurations?

Model Counting

Problem

- How many hardware configurations?
- Counting the number of satisfiable assignments of a propositional formula F. Denoted as $|F|$.

p	q	$F = a \wedge b$
1	1	1
1	0	0
0	1	0
0	0	0

Model Counting

Problem

- How many hardware configurations?
- Counting the number of satisfiable assignments of a propositional formula F. Denoted as $|F|$.

p	q	$F = a \wedge b$
1	1	1
1	0	0
0	1	0
0	0	0

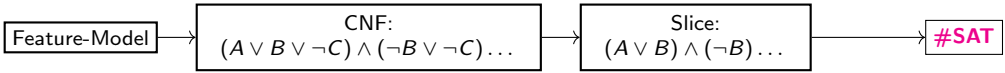
#SAT

Counts the number of solutions of a propositional formula.
Worst-case exponential complexity!

Projected Model Counting

Problem

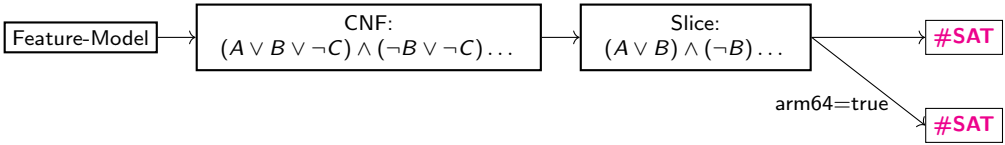
How many hardware configurations?



Projected Model Counting

Problem

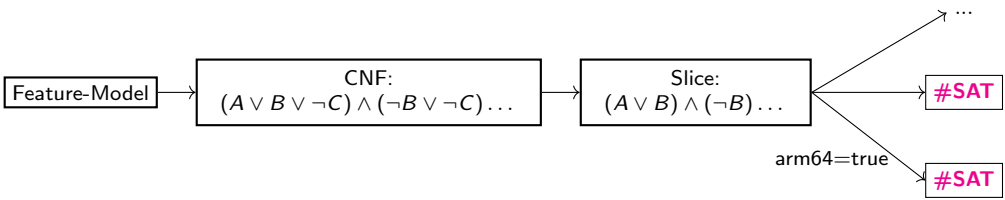
How many hardware configurations with arm64?



Projected Model Counting

Problem

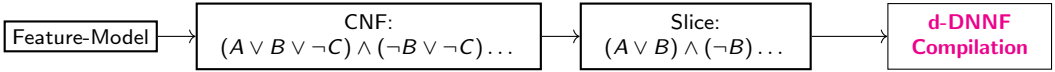
How many hardware configurations with X ?



Projected Model Counting

Problem

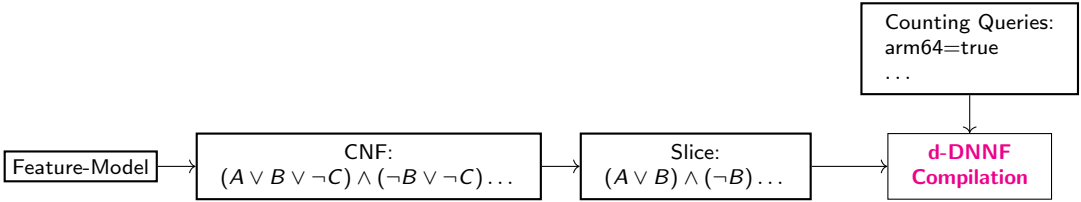
How many hardware configurations with X ?



Projected Model Counting

Problem

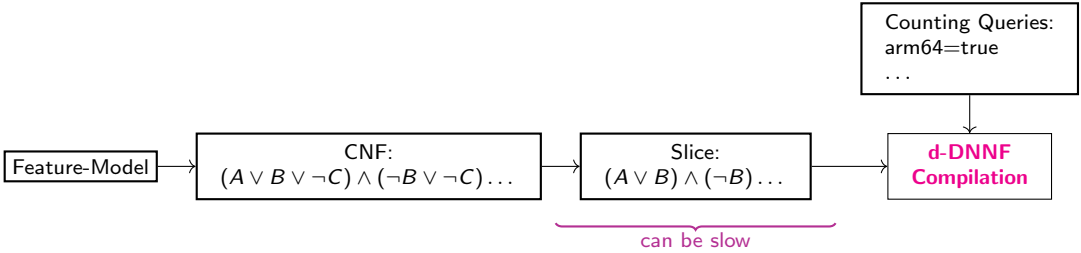
How many hardware configurations with X ?



Projected Model Counting

Problem

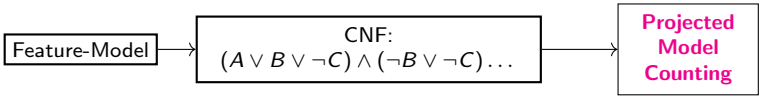
How many hardware configurations with X ?



Projected Model Counting

Problem

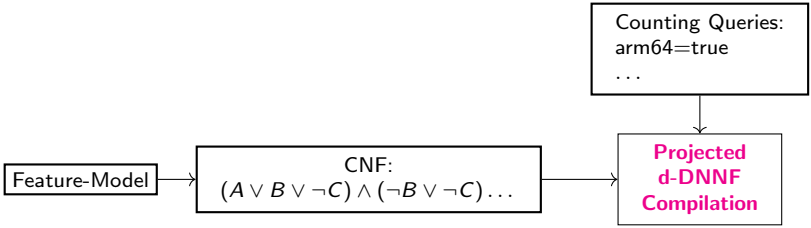
How many hardware configurations with X ?



Projected Model Counting

Problem

How many hardware configurations with X ?



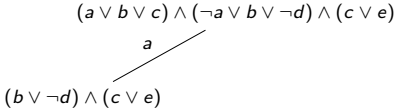
d-DNNF Compilation

DPLL

$$(a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e)$$

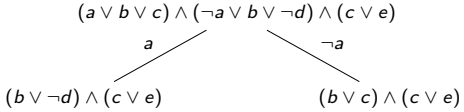
d-DNNF Compilation

DPLL



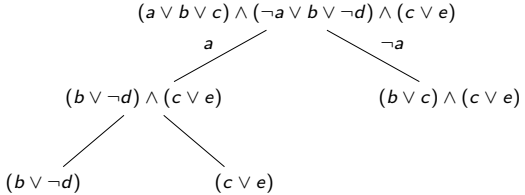
d-DNNF Compilation

DPLL



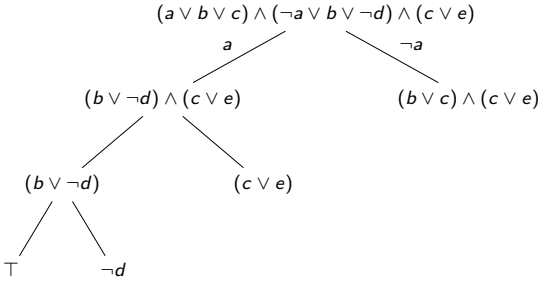
d-DNNF Compilation

DPLL



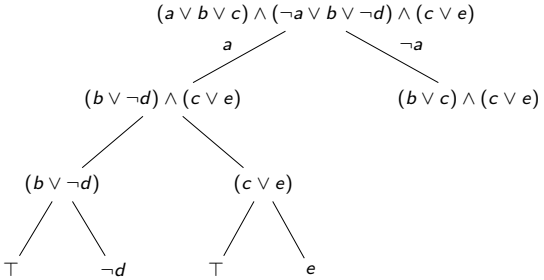
d-DNNF Compilation

DPLL



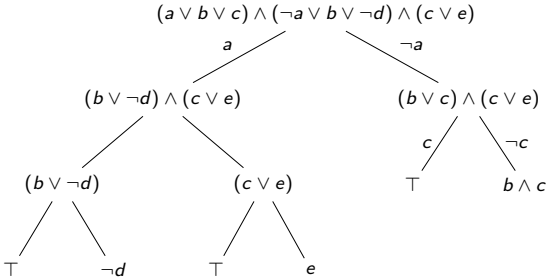
d-DNNF Compilation

DPLL



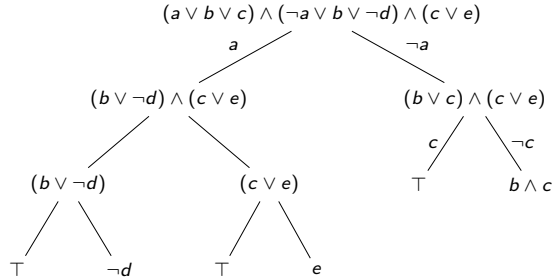
d-DNNF Compilation

DPLL

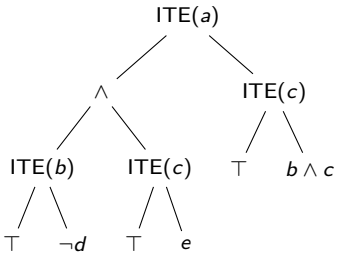


d-DNNF Compilation

DPLL



d-DNNF



ITE = If Then Else

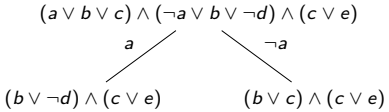
Heuristics

Vanilla DPLL is very slow

Heuristics

Vanilla DPLL is very slow

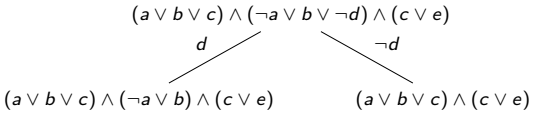
Variable Odering



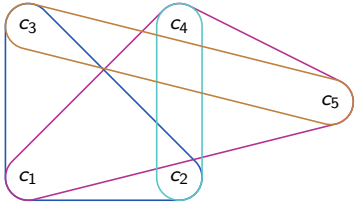
Heuristics

Vanilla DPLL is very slow

Variable Odering



Heuristics: Dual Hypergraph

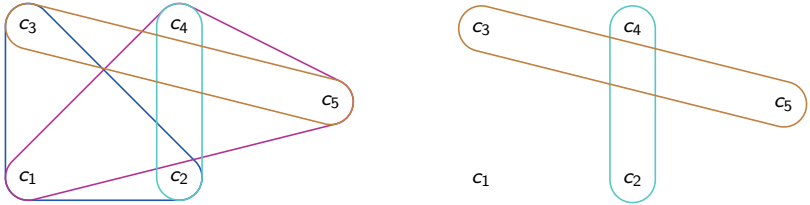


Construction

$$F = \underbrace{(a \vee b)}_{c_1} \wedge \underbrace{(a \vee \neg c)}_{c_2} \wedge \underbrace{(a \vee \neg d)}_{c_3} \wedge \underbrace{(b \vee \neg c)}_{c_4} \wedge \underbrace{(b \vee \neg d)}_{c_5}$$

Split formula into independent sub-problems of roughly equal size.

Heuristics: Dual Hypergraph



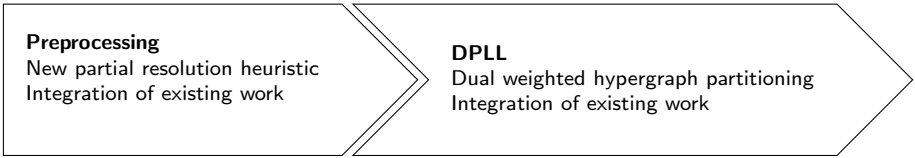
Construction

$F = (\underset{c_1}{a \vee b}) \wedge (\underset{c_2}{a \vee \neg c}) \wedge (\underset{c_3}{a \vee \neg d}) \wedge (\underset{c_4}{b \vee \neg c}) \wedge (\underset{c_5}{b \vee \neg d})$

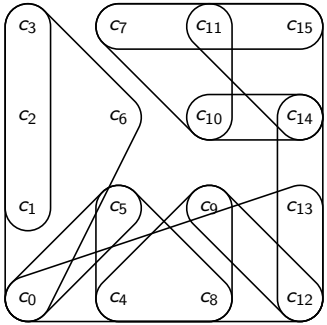
Split formula into independent sub-problems of roughly equal size.

Integration and Optimization in D4

[An Improved Decision-DNNF Compiler, Lagniez et al.]

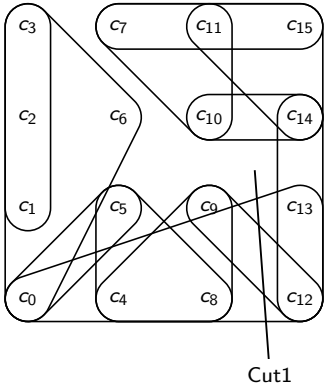


Dual Weighted Hypergraph Partitioning



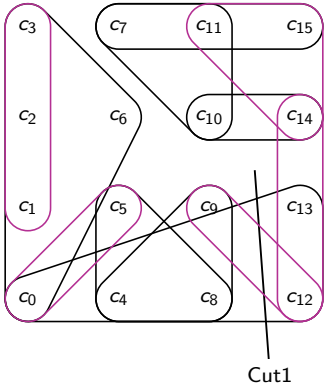
$F = c_0 \wedge c_1 \wedge ... \wedge c_{15}$

Dual Weighted Hypergraph Partitioning



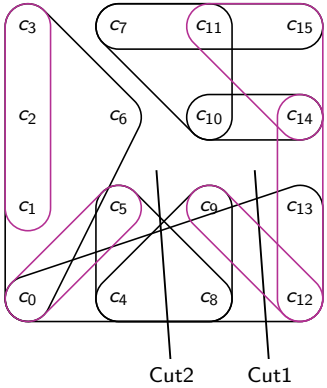
$F = c_0 \wedge c_1 \wedge ... \wedge c_{15}$

Dual Weighted Hypergraph Partitioning



$F = c_0 \wedge c_1 \wedge ... \wedge c_{15}$

Dual Weighted Hypergraph Partitioning



$F = c_0 \wedge c_1 \wedge ... \wedge c_{15}$

Partial Resolution

Greedily resolve "easy" sliced variables until the clause count increases

Partial Resolution

Greedily resolve "easy" sliced variables until the clause count increases

Clause Ratio

$F = (a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e)$

Sliced variables $= c, e$

Resolution of c and e is trivial $\implies F = (\neg a \vee b \vee \neg d)$

Partial Resolution

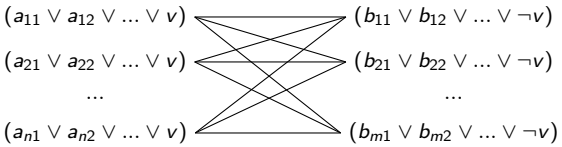
Greedily resolve "easy" sliced variables until the clause count increases

Clause Ratio

$F = (a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e)$

Sliced variables $= c, e$

Resolution of c and e is trivial $\implies F = (\neg a \vee b \vee \neg d)$



Partial Resolution

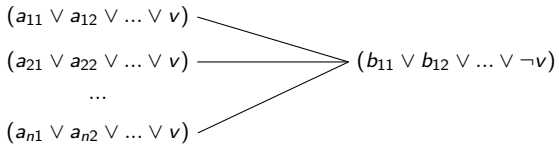
Greedily resolve "easy" sliced variables until the clause count increases

Clause Ratio

$F = (a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e)$

Sliced variables $= c, e$

Resolution of c and e is trivial $\implies F = (\neg a \vee b \vee \neg d)$



Partial Resolution

Greedily resolve "easy" sliced variables until the clause count increases

Connectivity

$(\neg a \vee b \vee v), (a \vee b \vee \neg v) \rightarrow (\neg a \vee b \vee b \vee c) \equiv \top$

Partial Resolution

Greedily resolve "easy" sliced variables until the clause count increases

Connectivity

$(\neg a \vee b \vee v), (a \vee b \vee \neg v) \rightarrow (\neg a \vee b \vee b \vee c) \equiv \top$

Simpical Variable [1]: neighbors form a clique through clauses

[1] from GPMC source code

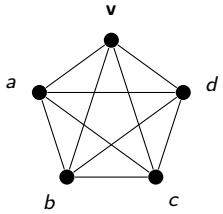
Partial Resolution

Greedily resolve "easy" sliced variables until the clause count increases

Connectivity

$(\neg a \vee b \vee v), (a \vee b \vee \neg v) \rightarrow (\neg a \vee b \vee b \vee c) \equiv \top$

Simpical Variable [1]: neighbors form a clique through clauses



[1] from GPMC source code

Partial Resolution

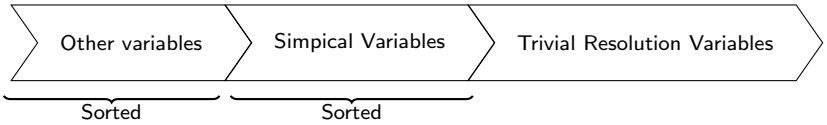
Greedily resolve "easy" sliced variables until the clause count increases

New Combined Heuristic

Group by:

1. Trivial resolution variables
2. Simpical variables
3. Other variables

Sort by: $v_p * v_n$ and average clause length



Generated Projection for Feature Models (Table)

Model	d4-pmc	arjun	pD4	slice	gpmc
Smarch.2.6.32-2var	0	0	0	0	0
Smarch.2.6.28.6-icse11	0	0	0	0	0
Smarch.freetz	0	42	**65**	0	36
Smarch.buildroot	0	92	**100**	0	93
KConfig.linux-2.6.33.3	27	90	**96**	63	55
Smarch.embtoolkit	20	100	100	0	100
Smarch.freebsd-icse11	**100**	69	56	23	60
Smarch.uClinux-config	100	100	100	85	100
automotive02.automotive2_4	100	100	100	100	100

Projected d-DNNF Compilation for Feature Models

Master's Thesis by Jacob Loth, supervised by Chico Sundermann | Thomas Thüm | April 9, 2024

PADERBORN UNIVERSITY