

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

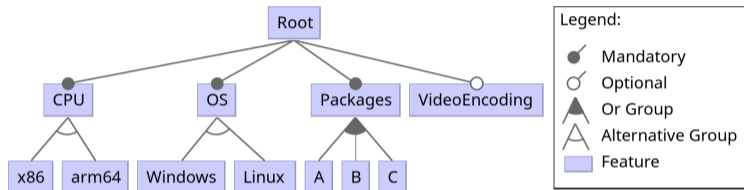


Projected d-DNNF Compilation for Feature Models

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

1. What is the problem?

Feature Models and Conjunctive Normal Forms



We can convert feature models to propositional formulas!

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

Deterministic Decomposable Negation Normal Form (d-DNNF)

Any propositional formula which is:

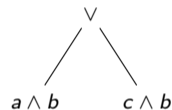
deterministic

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:

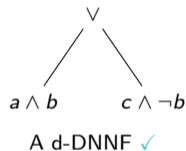
deterministic

Exclusive or-operators $F = A \vee B$

Never simultaneous $A = 1$ and $B = 1$

If-then-else

$|F| = |A| + |B|$



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 $B = 1$

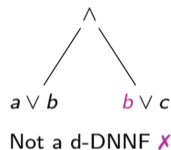
If-then-else

$$|F| = |A| + |B|$$

Decomposable

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

$$|F| = |A| * |B|$$



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 $B = 1$

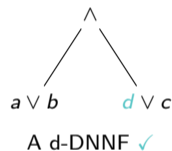
If-then-else

$$|F| = |A| + |B|$$

Decomposable

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

$$|F| = |A| * |B|$$



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

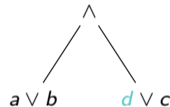
$$|F| = |A| + |B|$$

Decomposable

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

$$|F| = |A| * |B|$$

Negation Normal Form



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

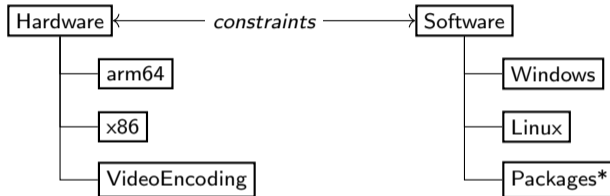
Feature-Model Slicing / Projection of Formulas

Feature Model



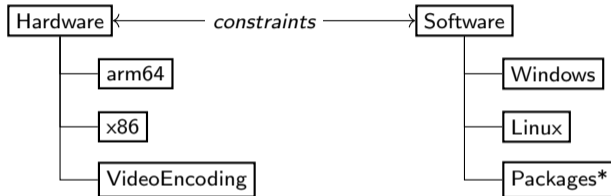
Feature-Model Slicing / Projection of Formulas

Feature Model



Feature-Model Slicing / Projection of Formulas

Feature Model

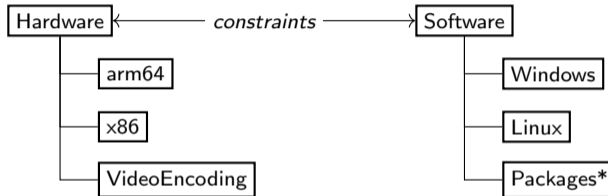


Problem

How many hardware configurations?

Feature-Model Slicing / Projection of Formulas

Feature Model



Problem

How many hardware configurations?

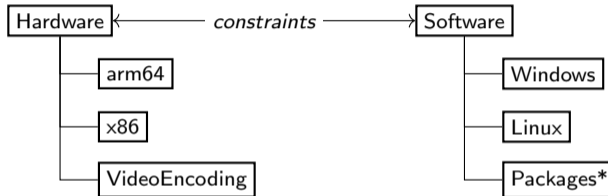
Transitive Constraints

VideoEncoding \implies Windows

Windows \implies x86

Feature-Model Slicing / Projection of Formulas

Feature Model



Problem

How many hardware configurations?

Transitive Constraints

VideoEncoding \implies Windows

Windows \implies x86

Sliced: VideoEncoding \implies x86

Feature-Model Slicing / Projection of Formulas

[Krieter et al. SPLC16]

Resolve all clauses with \vee with all clauses with $\neg\vee$

Resolving Two Clauses

$$(\neg VideoEncoding \vee Windows), (x86 \vee \neg Windows) \rightarrow (\neg VideoEncoding \vee x86)$$

Feature-Model Slicing / Projection of Formulas

[Krieter et al. SPLC16]

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

Resolving Two Clauses

$$\begin{aligned} &(\neg \text{VideoEncoding} \vee \text{Windows}), (\text{x86} \vee \neg \text{Windows}) \rightarrow (\neg \text{VideoEncoding} \vee \text{x86}) \\ &(a_1 \vee a_2 \vee \dots \vee v), (b_1 \vee b_2 \vee \dots \vee \neg v) \rightarrow (a_1 \vee a_2 \vee \dots \vee b_1 \vee b_2 \vee \dots) \end{aligned}$$

Feature-Model Slicing / Projection of Formulas

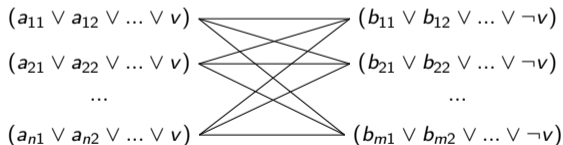
[Krieter et al. SPLC16]

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

Resolving Two Clauses

$$\begin{aligned} &(\neg \text{VideoEncoding} \vee \text{Windows}), (\text{x86} \vee \neg \text{Windows}) \rightarrow (\neg \text{VideoEncoding} \vee \text{x86}) \\ &(a_1 \vee a_2 \vee \dots \vee v), (b_1 \vee b_2 \vee \dots \vee \neg v) \rightarrow (a_1 \vee a_2 \vee \dots \vee b_1 \vee b_2 \vee \dots) \end{aligned}$$

Resolving Many Clauses

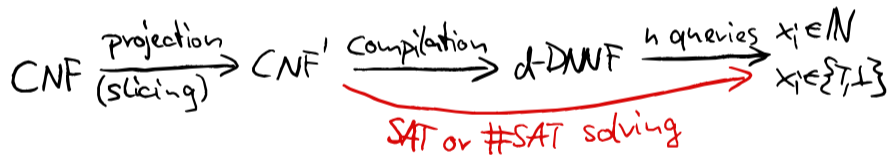


Exponential clause count increase for multiple variables.

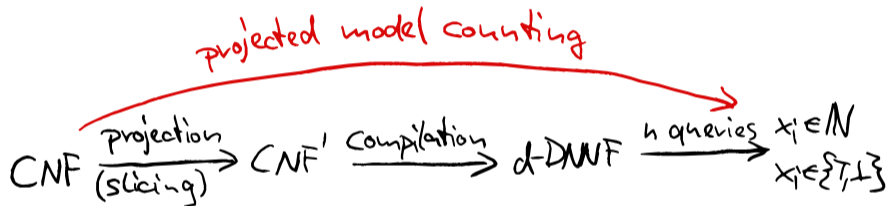
Scalability Issues with State-of-the-Art Reasoning



Scalability Issues with State-of-the-Art Reasoning



Scalability Issues with State-of-the-Art Reasoning



2. Why should you care about this thesis?

Why should you care about this thesis?

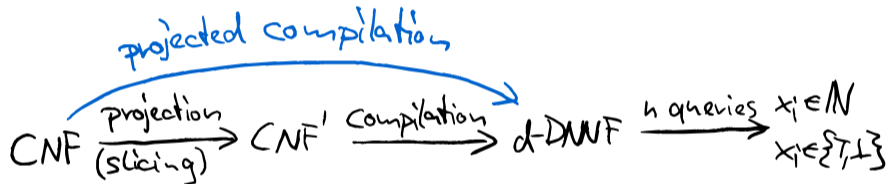
Standard Evaluation Criteria

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

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

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

Is it novel?

Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e

$$\begin{array}{ccc} & (a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e) & \\ & \swarrow \quad \searrow & \\ a & & \neg a \\ (b \vee \neg d) \wedge (c \vee e) & & (b \vee c) \wedge (c \vee e) \end{array}$$

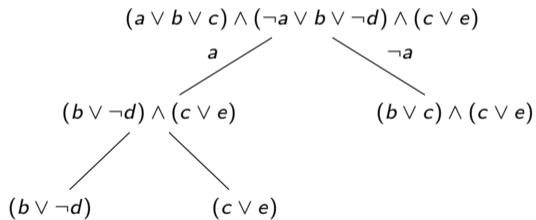
Is it novel?

Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e



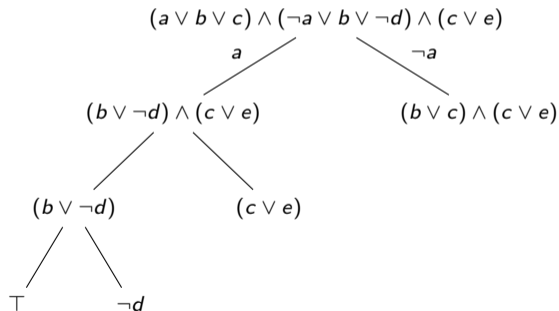
Is it novel?

Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e



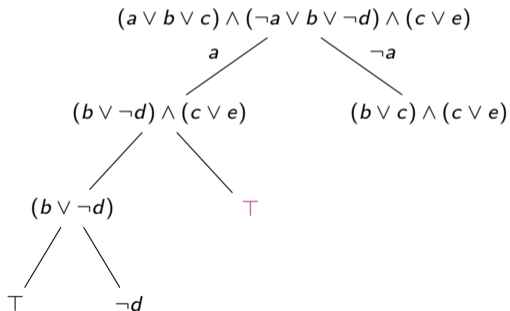
Is it novel?

Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e



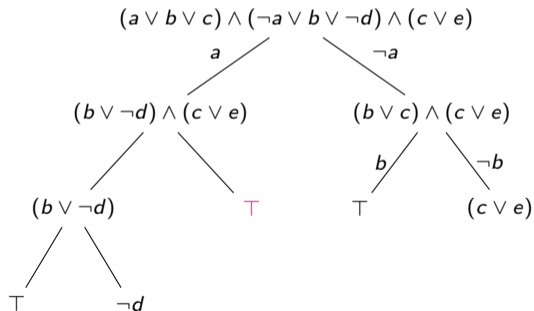
Is it novel?

Projection + Compilation = Projected Compilation

Projected d-DNNF Compilation

Projected variables = a, b, d

Sliced variables = c, e



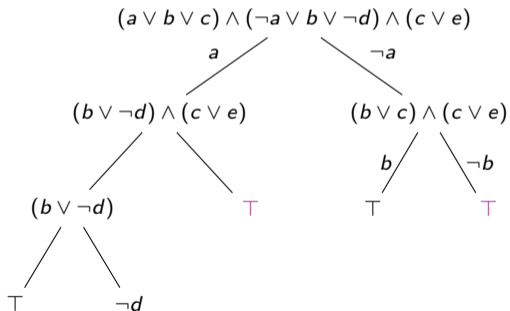
Is it novel?

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

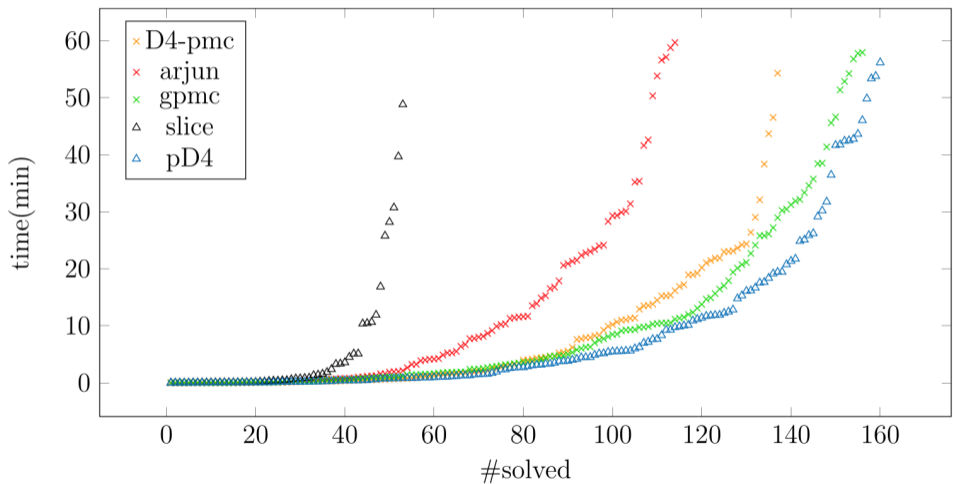
- **MC2022**: Private+public instances from the MC2022 (many unknown sources...)
- **Generated Projection**: Adding randomly selected projected variables to real feature models
- **Industrial Projection**: Real feature model slicing problems

Questions

Compare runtime performance (and d-DNNF size)

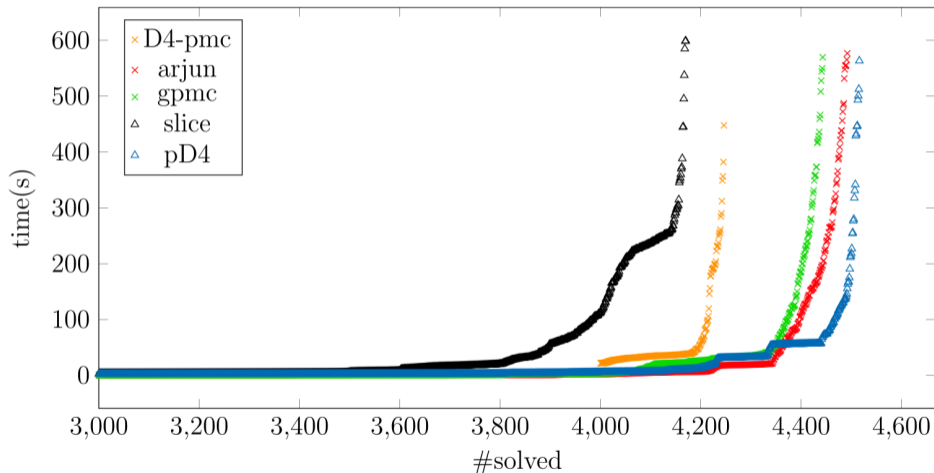
Is it significant?

Projected Model Counting Competition 2022



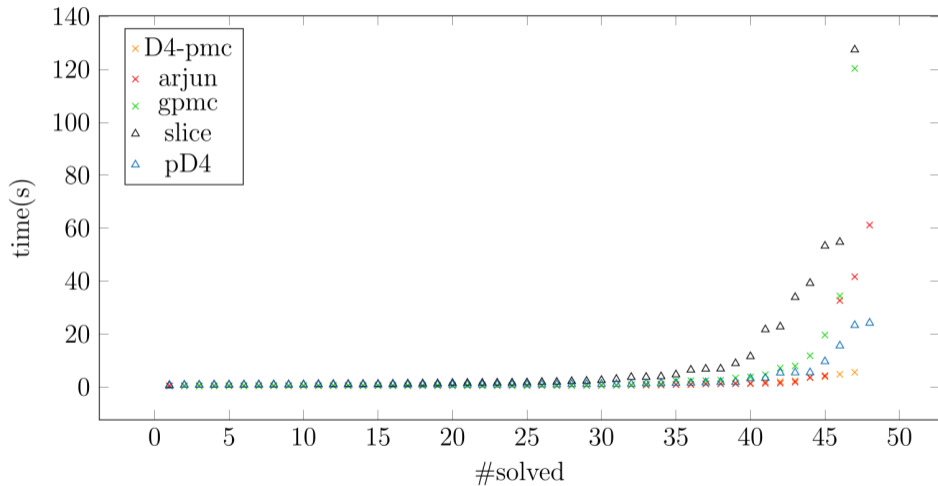
Is it sound?

Random Projections on Feature Models



Is it sound?

Real Projections from Automotive Industry



Is it verifiable?

pD4 on Github

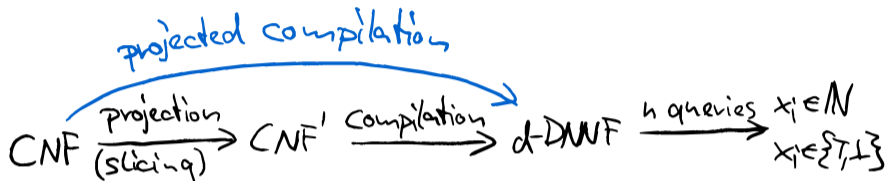


Replication Package



Is it clear?

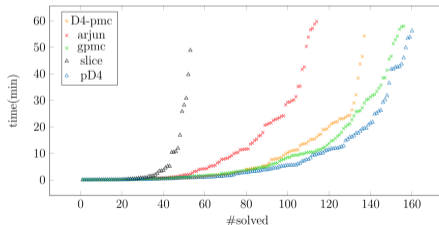
Projected d-DNNF Compilation for Feature Models



Projected Compilation = Projection + Compilation

projected d-DNNF compilation ...

- is faster than 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
- 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

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

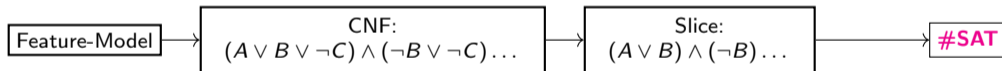
#SAT

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

Projected Model Counting

Problem

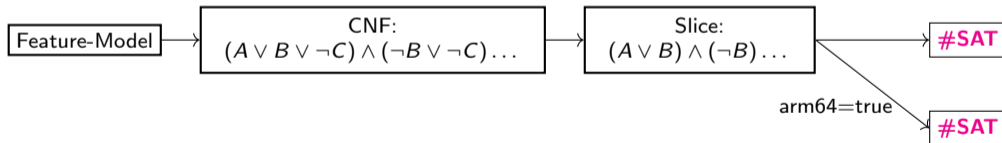
How many hardware configurations?



Projected Model Counting

Problem

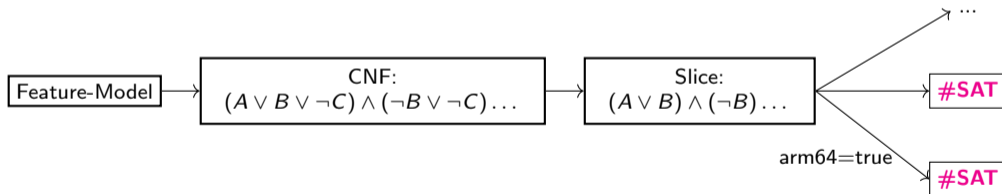
How many hardware configurations **with arm64**?



Projected Model Counting

Problem

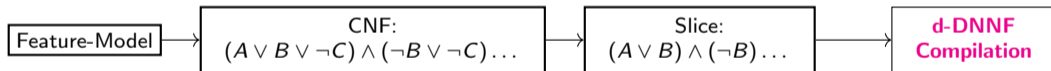
How many hardware configurations **with X** ?



Projected Model Counting

Problem

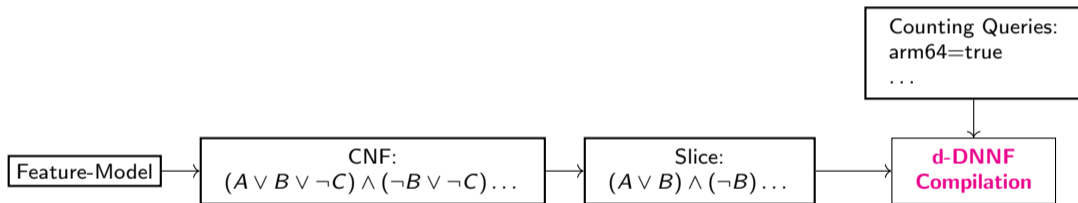
How many hardware configurations **with X** ?



Projected Model Counting

Problem

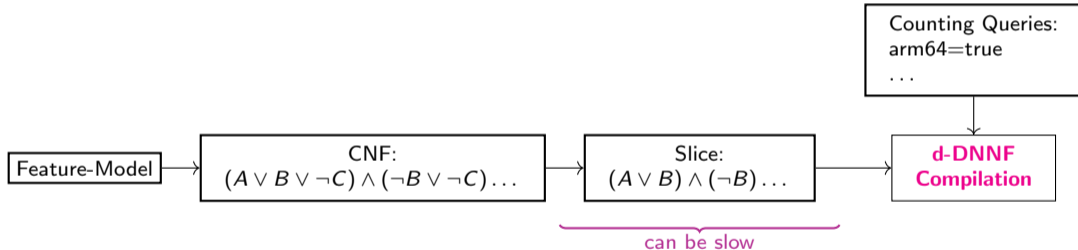
How many hardware configurations **with X** ?



Projected Model Counting

Problem

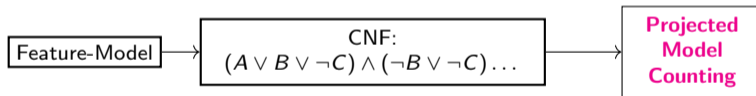
How many hardware configurations **with X** ?



Projected Model Counting

Problem

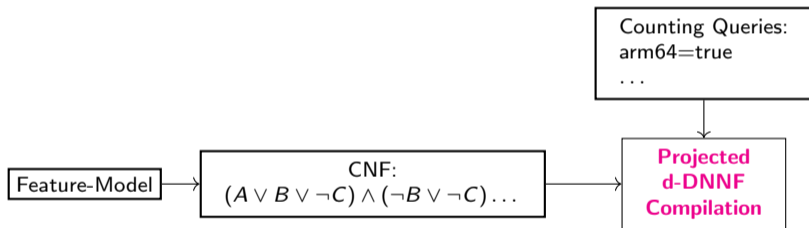
How many hardware configurations **with X** ?



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

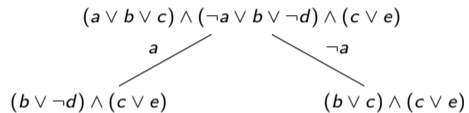
$$(a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e)$$

a

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

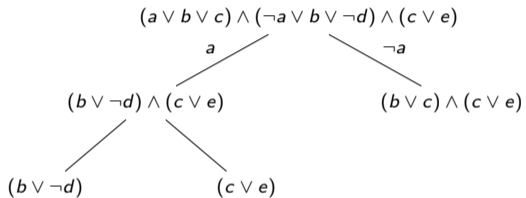
d-DNNF Compilation

DPLL



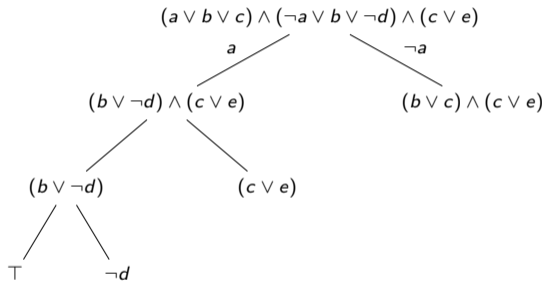
d-DNNF Compilation

DPLL



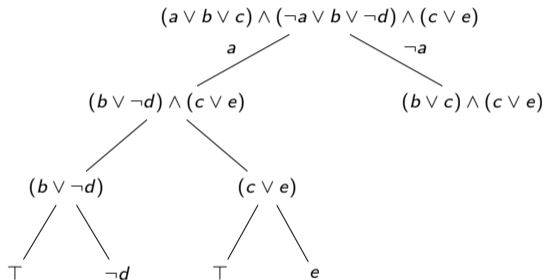
d-DNNF Compilation

DPLL



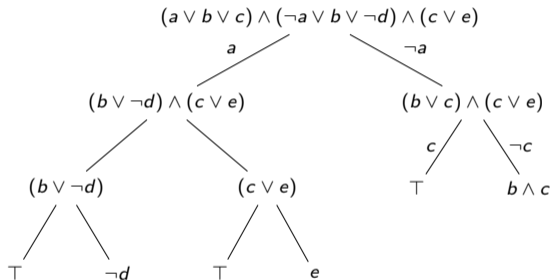
d-DNNF Compilation

DPLL



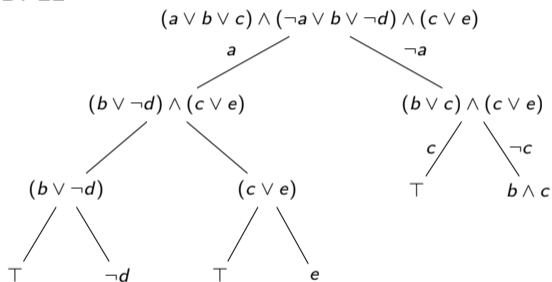
d-DNNF Compilation

DPLL

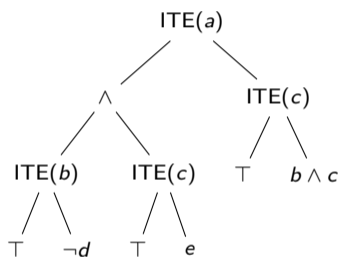


d-DNNF Compilation

DPLL



d-DNNF



ITE = If Then Else

Heuristics

Vanilla DPLL is very slow

Heuristics

Vanilla DPLL is very slow

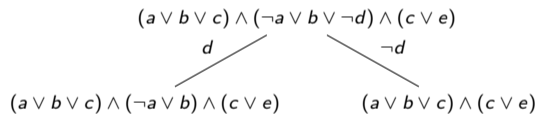
Variable Ordering

$$\begin{array}{ccc} & (a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e) & \\ & \swarrow \quad \searrow & \\ a & & \neg a \\ (b \vee \neg d) \wedge (c \vee e) & & (b \vee c) \wedge (c \vee e) \end{array}$$

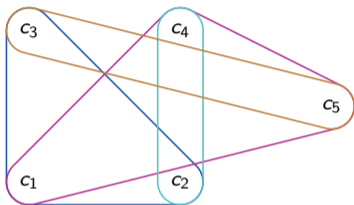
Heuristics

Vanilla DPLL is very slow

Variable Ordering



Heuristics: Dual Hypergraph

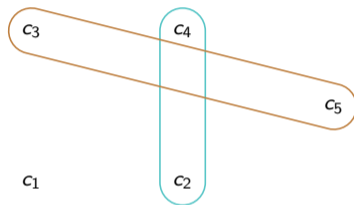
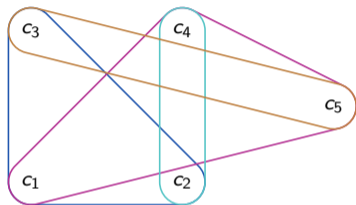


Construction

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

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

Heuristics: Dual Hypergraph



Construction

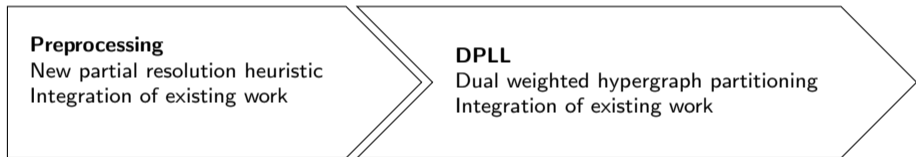
$$F = (a \vee b) \wedge (a \vee \neg c) \wedge (a \vee \neg d) \wedge (b \vee \neg c) \wedge (b \vee \neg d)$$

c_1 c_2 c_3 c_4 c_5

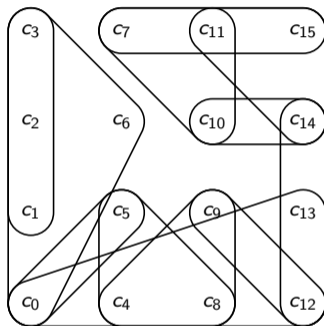
Split formula into independent sub-problems of roughly equal size.

Integration and Optimization in D4

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

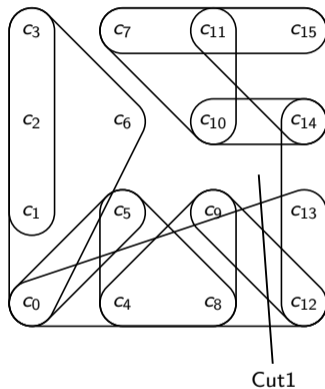


Dual Weighted Hypergraph Partitioning



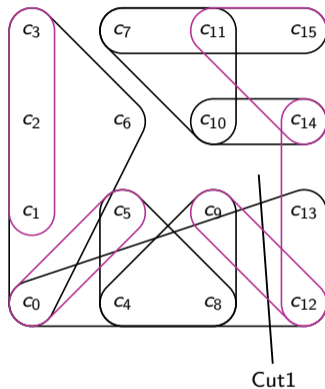
$$F = c_0 \wedge c_1 \wedge \dots \wedge c_{15}$$

Dual Weighted Hypergraph Partitioning



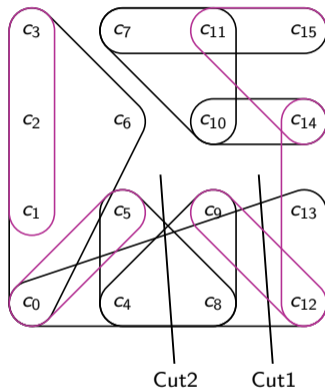
$$F = c_0 \wedge c_1 \wedge \dots \wedge c_{15}$$

Dual Weighted Hypergraph Partitioning



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

Dual Weighted Hypergraph Partitioning



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

Partial Resolution

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

Partial Resolution

Greedy 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

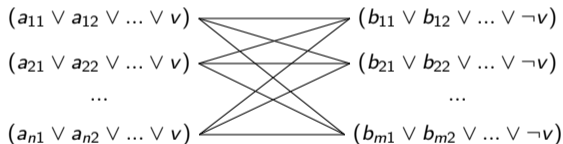
Greedily resolve "easy" sliced variables until the clause count increases

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

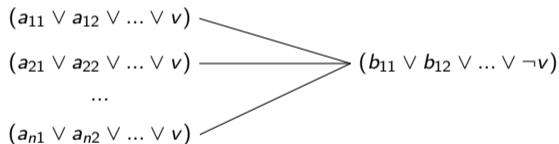
Greedyly resolve "easy" sliced variables until the clause count increases

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 Variable¹: neighbors form a clique through clauses

¹from GPMC source code

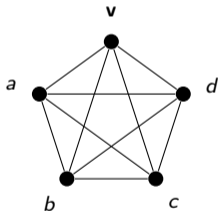
Partial Resolution

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 Variable¹: neighbors form a clique through clauses



¹from GPMC source code

Partial Resolution

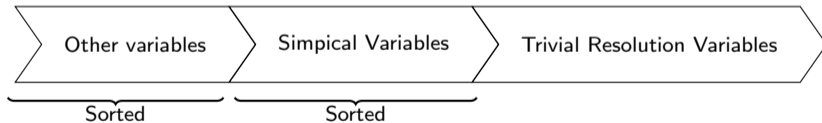
Greedy resolve "easy" sliced variables until the clause count increases

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
<i>Smarch.2.6.32-2var</i>	0	0	0	0	0
<i>Smarch.2.6.28.6-icse11</i>	0	0	0	0	0
<i>Smarch.freetz</i>	0	42	65	0	36
<i>Smarch.buildroot</i>	0	92	100	0	93
<i>KConfig.linux-2.6.33.3</i>	27	90	96	63	55
<i>Smarch.embtoolkit</i>	20	100	100	0	100
<i>Smarch.freebsd-icse11</i>	100	69	56	23	60
<i>Smarch.uClinux-config</i>	100	100	100	85	100
<i>automotive02.automotive2_4</i>	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