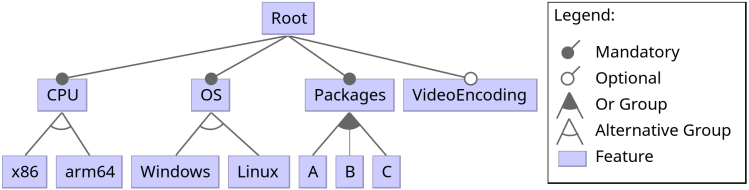
Projected d-DNNF Compilation for Feature Models

Master's Thesis | Jacob Loth | October 30, 2023

universität ulm

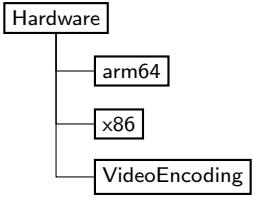
1. Motivation

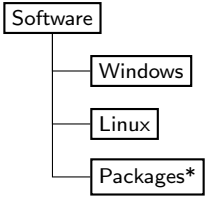
Feature Models



Feature-Model Slicing

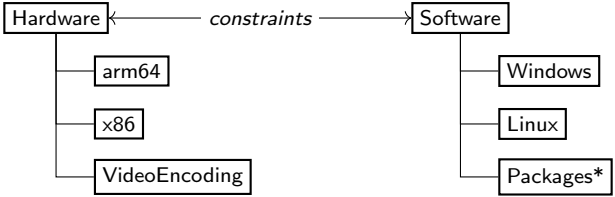
Feature Model





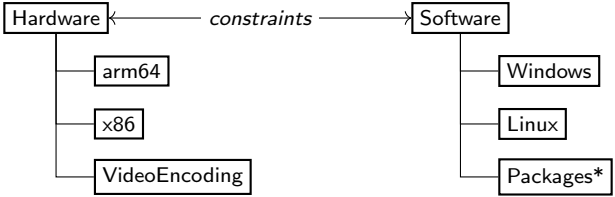
Feature-Model Slicing

Feature Model



Feature-Model Slicing

Feature Model

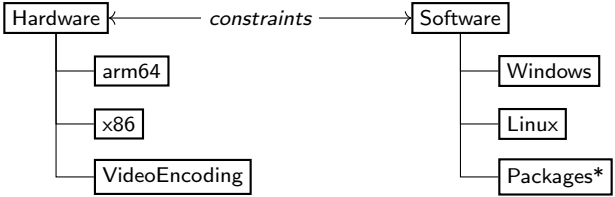


Problem

How many hardware configurations?

Feature-Model Slicing

Feature Model



Problem

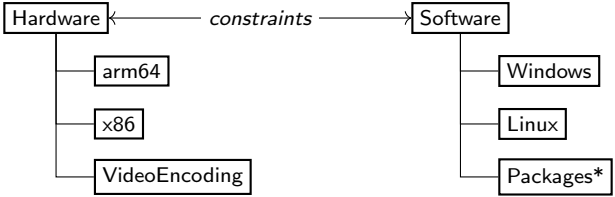
How many hardware configurations?

Transitive Constraints

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

Feature-Model Slicing

Feature Model



Problem

How many hardware configurations?

Transitive Constraints

VideoEncoding $\Longrightarrow$ Windows

Windows $\Longrightarrow$ x86

Sliced: VideoEncoding $\Longrightarrow$ x84

Feature-Model Counting

Problem

How many hardware configurations?

Feature-Model Counting

Problem

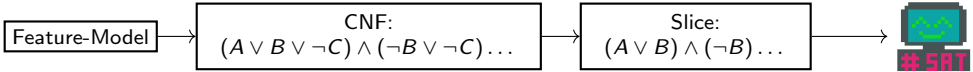
How many hardware configurations?

Counts the number of solutions of a boolean formula

Feature-Model Counting

Problem

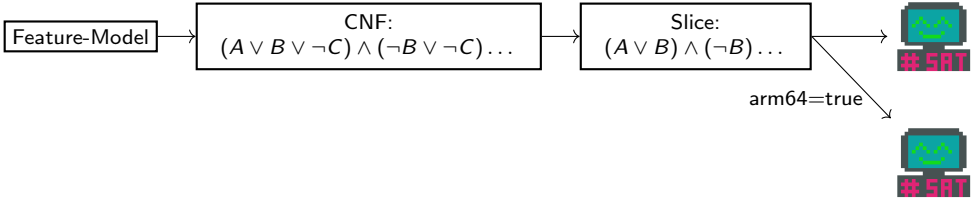
How many hardware configurations?



Feature-Model Counting

Problem

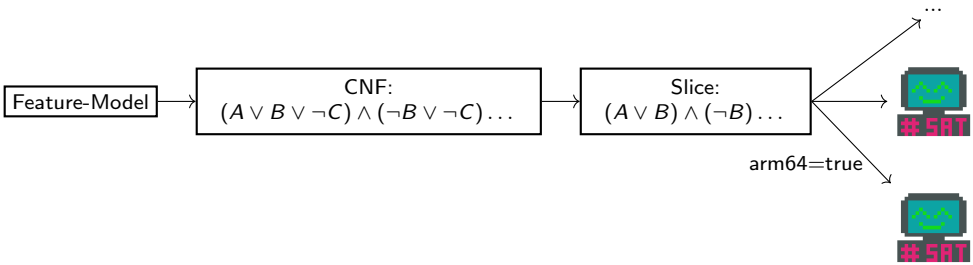
How many hardware configurations with arm64?



Feature-Model Counting

Problem

How many hardware configurations with X ?



Feature-Model Counting

Problem

How many hardware configurations with X ?

Feature-Model → CNF: $(A \vee B \vee \neg C) \wedge (\neg B \vee \neg C) \ldots$ → Slice: $(A \vee B) \wedge (\neg B) \ldots$ →

Feature-Model Counting

Problem

How many hardware configurations with X ?

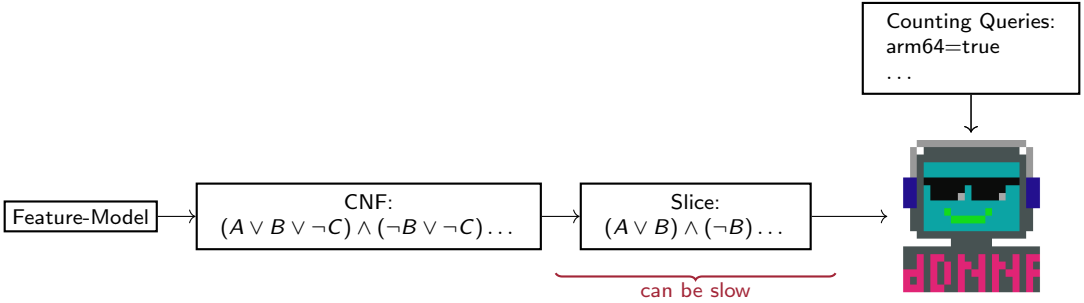
Counting Queries:
arm64=true
...

Feature-Model → CNF: $(A \vee B \vee \neg C) \wedge (\neg B \vee \neg C) \ldots$ → Slice: $(A \vee B) \wedge (\neg B) \ldots$ →

Feature-Model Counting

Problem

How many hardware configurations with X ?



Feature-Model Counting

Problem

How many hardware configurations with X ?

Counting Queries:
arm64=true
...

Feature-Model → CNF:
$(A \vee B \vee \neg C) \wedge (\neg B \vee \neg C) \ldots$ →

Projected d-DNNF Compilation

Compiles F into d-DNNF while slicing a set of variables Y and keep a set of projected variables X

Projected d-DNNF Compilation

Compiles F into d-DNNF while slicing a set of variables Y and keep a set of projected variables X

Related Problem

Projected Model Counting(PMC):

$\Longrightarrow$ Counting all assignments to variables $Var(F) - Y$ that have some extension to a solution in F

What are d-DNNFs?

Any boolean formula that has...

What are d-DNNFs?

Any boolean formula that has...

- Decomposable AND-Nodes

$$(A \vee B) \wedge (A \vee C) \implies \text{Not Decomposable} \times$$

What are d-DNNFs?

Any boolean formula that has...

- Decomposable AND-Nodes

$$(A \vee B) \wedge (A \vee C) \implies \text{Not Decomposable } \times$$
$$(A \vee B) \wedge (D \vee C) \implies \text{Decomposable } \checkmark$$

What are d-DNNFs?

Any boolean formula that has...

- Decomposable AND-Nodes
- Deterministic OR-Nodes
  (If-Then-Else)

$(A \vee A) \implies$ Not Deterministic ✗

What are d-DNNFs?

Any boolean formula that has...

- Decomposable AND-Nodes
- Deterministic OR-Nodes (If-Then-Else)

$$(A \vee A) \implies \text{Not Deterministic} \ \textcircled{\scriptsize\times}$$

$$(A \wedge B) \vee (A \wedge \neg B) \implies \text{Deterministic} \ \checkmark$$

What are d-DNNFs?

Any boolean formula that has...

- Decomposable AND-Nodes
- Deterministic OR-Nodes (If-Then-Else)
- Negations only right before variables

What are d-DNNFs?

Any boolean formula that has...

- Decomposable AND-Nodes
- Deterministic OR-Nodes (If-Then-Else)
- Negations only right before variables

Knowledge Compilation
+
Linear Time Model Counting

What are d-DNNFs?

Any boolean formula that has...

- Decomposable AND-Nodes
- Deterministic OR-Nodes (If-Then-Else)
- Negations only right before variables

d-DNNF Compilation:

Turn CNF into d-DNNF

Knowledge Compilation
+
Linear Time Model Counting

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

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

$$\xrightarrow{a}$$

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

d-DNNF Compilation

DPLL

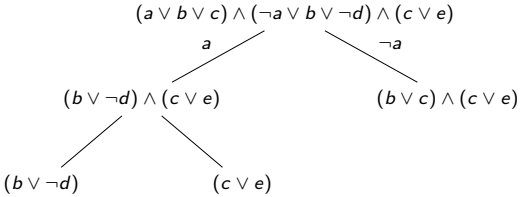
$$(a \vee b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (c \vee e)$$

a $\qquad$ $\neg a$

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

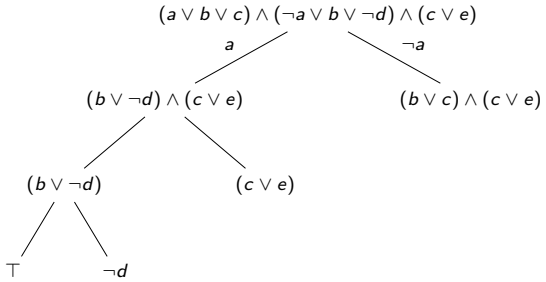
d-DNNF Compilation

DPLL



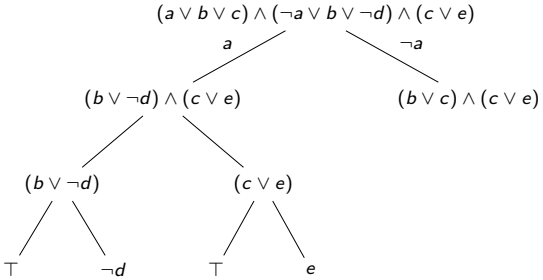
d-DNNF Compilation

DPLL



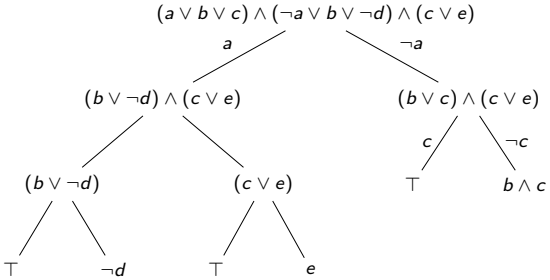
d-DNNF Compilation

DPLL



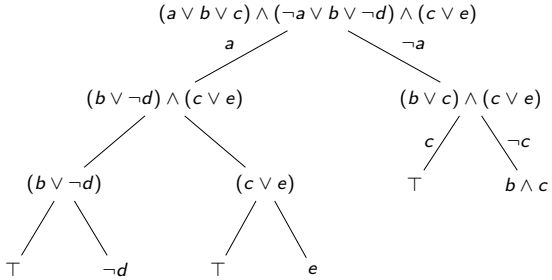
d-DNNF Compilation

DPLL

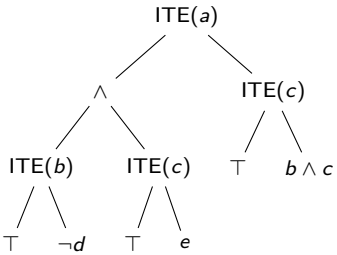


d-DNNF Compilation

DPLL



DNNF



ITE = If Then Else

2. Our Contributions (so far)

Projected d-DNNF Compilation

Concept

Projected variables $= a, b, d$
Sliced variables $= c, e$

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

Projected d-DNNF Compilation

Concept

Projected variables $= a, b, d$
Sliced variables $= c, e$

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

$$a$$

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

Projected d-DNNF Compilation

Concept

Projected variables $= a, b, d$
Sliced variables $= c, e$

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

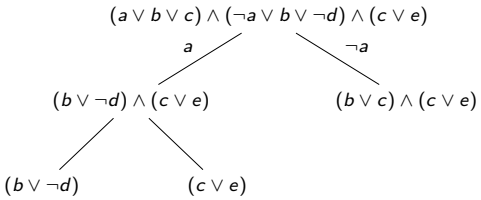
a $\qquad$ $\neg a$

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

Projected d-DNNF Compilation

Concept

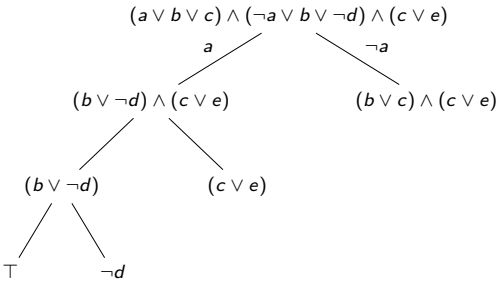
Projected variables $= a, b, d$
Sliced variables $= c, e$



Projected d-DNNF Compilation

Concept

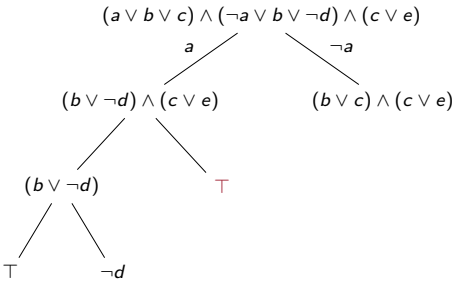
Projected variables $= a, b, d$
Sliced variables $= c, e$



Projected d-DNNF Compilation

Concept

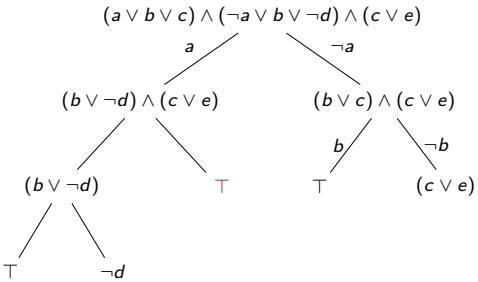
Projected variables $= a, b, d$
Sliced variables $= c, e$



Projected d-DNNF Compilation

Concept

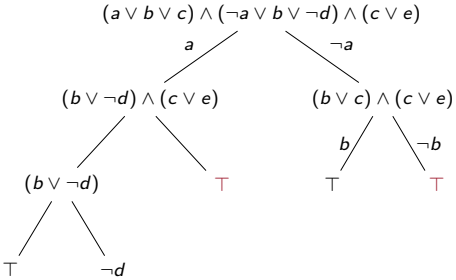
Projected variables $= a, b, d$
Sliced variables $= c, e$



Projected d-DNNF Compilation

Concept

Projected variables $= a, b, d$
Sliced variables $= c, e$



Projected d-DNNF Compilation

Implementation

Based on the D4 d-DNNF compiler

- Replace sets of clauses containing only projected variables with $\top$ or $\bot$
- Always ignore sliced literals
- Restrict variable selection to projected set

Projected d-DNNF Compilation

Implementation

Based on the D4 d-DNNF compiler

- Replace sets of clauses containing only projected variables with $\top$ or $\bot$
- Always ignore sliced literals
- Restrict variable selection to projected set $\implies$ losing D4s variable selection heuristics

Optimizations

Variable Selection

Crucial for performance: Separate clauses to generate decomposable AND-Nodes

Optimizations

Variable Selection

Crucial for performance: Separate clauses to generate decomposable AND-Nodes

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

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

Optimizations

Variable Selection

Crucial for performance: Separate clauses to generate decomposable AND-Nodes

$$(a \lor b \lor c) \land (\neg a \lor b \lor \neg d) \land (c \lor e)$$

$$\xrightarrow{\;d\;} (a \lor b \lor c) \land (\neg a \lor b) \land (c \lor e) \qquad \xrightarrow{\;\neg d\;} (a \lor b \lor c) \land (c \lor e)$$

Optimizations

Variable Selection

Crucial for performance: Separate clauses to generate decomposable AND-Nodes

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

$$\xrightarrow{d} (a \vee b \vee c) \wedge (\neg a \vee b) \wedge (c \vee e) \qquad \xrightarrow{\neg d} (a \vee b \vee c) \wedge (c \vee e)$$

D4 separates clauses using cut sets, *BUT* cuts may contain sliced variables
$\implies$ Adapt D4s variable selection heuristics to favor clean cuts

Optimizations

Variable Selection

Crucial for performance: Separate clauses to generate decomposable AND-Nodes

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

$$\xrightarrow{d} (a \vee b \vee c) \wedge (\neg a \vee b) \wedge (c \vee e) \qquad \xrightarrow{\neg d} (a \vee b \vee c) \wedge (c \vee e)$$

D4 separates clauses using cut sets, *BUT* cuts may contain sliced variables
$\implies$ Adapt D4s variable selection heuristics to favor clean cuts

Preprocessing

Make the problem smaller, various methods adapted from PMC
Problem: preserve true equivalence

Optimizations

Variable Selection

Crucial for performance: Separate clauses to generate decomposable AND-Nodes

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

$$\xrightarrow{d} (a \vee b \vee c) \wedge (\neg a \vee b) \wedge (c \vee e) \qquad \xrightarrow{\neg d} (a \vee b \vee c) \wedge (c \vee e)$$

D4 separates clauses using cut sets, *BUT* cuts may contain sliced variables
$\implies$ Adapt D4s variable selection heuristics to favor clean cuts

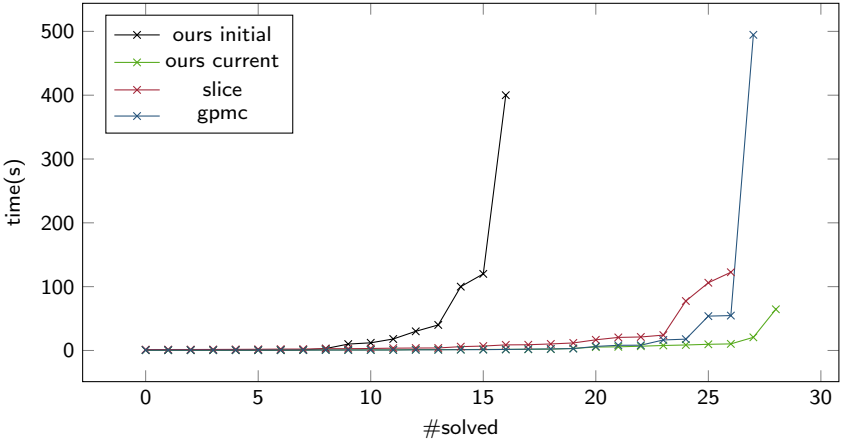
Preprocessing

Make the problem smaller, various methods adapted from PMC
Problem: preserve true equivalence

Technical Stuff

Cache and branch friendly variable and clause renaming/ordering

Results


ours initial
ours current
slice
gpmc
time(s)
500
400
300
200
100
0
0
5
10
15
20
25
30
#solved

Current State

Concept:	Done
Implementation:	Done
Explore optimization:	Almost Done (preprocessing, fine-tuning)
Evaluation:	WIP (more tests needed)
Writing the thesis:	Started

The End

Projected d-DNNF Compilation

Compiles F into d-DNNF while slicing a set of variables Y and keep a set of projected variables X