



[Jahrtausendturm Magdeburg]

Tseitin or not Tseitin?

The Impact of CNF Transformations on Feature-Model Analyses

SE 2023 (ASE 2022) — February 22–24 — Paderborn

Elias Kuitert¹, Sebastian Krieter², Chico Sundermann², Thomas Thüm², Gunter Saake¹

University of Magdeburg¹, Ulm²



Implementing Configurable Software Systems

A Configurable Graph

```
class Node {  
    #ifdef LABELED  
        std::string label;  
    #endif  
    #ifdef COLORED  
        std::string color;  
    #endif  
};  
  
class Edge {  
    #ifdef DIRECTED  
        Node from, to;  
    #elif UNDIRECTED && HYPER  
        std::set<Node> nodes;  
    #endif  
};
```

Product Line Implementation

(here: C++ with C preprocessor)

Implementing Configurable Software Systems

A Configurable Graph

```
class Node {  
    #ifdef LABELED  
        std::string label;  
    #endif  
    #ifdef COLORED  
        std::string color;  
    #endif  
};  
  
class Edge {  
    #ifdef DIRECTED  
        Node from, to;  
    #elif UNDIRECTED && HYPER  
        std::set<Node> nodes;  
    #endif  
};
```

Product Line Implementation
(here: C++ with C preprocessor)

#define
LABELED
DIRECTED

Configuration

A Labeled Directed Graph

```
class Node {  
    std::string label;  
};  
  
class Edge {  
    Node from, to;  
};
```

Product Implementation

Implementing Configurable Software Systems

A Configurable Graph

```
class Node {  
    #ifdef LABELED  
        std::string label;  
    #endif  
    #ifdef COLORED  
        std::string color;  
    #endif  
};  
  
class Edge {  
    #ifdef DIRECTED  
        Node from, to;  
    #elif UNDIRECTED && HYPER  
        std::set<Node> nodes;  
    #endif  
};
```

Product Line Implementation
(here: C++ with C preprocessor)

→ #define
LABELED
DIRECTED

→ #define
COLORED
UNDIRECTED
HYPER

Configuration

A Labeled Directed Graph

```
class Node {  
    std::string label;  
};  
  
class Edge {  
    Node from, to;  
};
```

A Colored Undirected Hypergraph

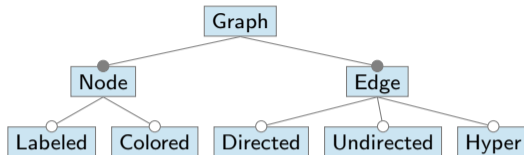
```
class Node {  
    std::string color;  
};  
  
class Edge {  
    std::set<Node> nodes;  
};
```

Product Implementation

Modeling Features and their Dependencies

Feature Models

- tree models **features**
- cross-tree **constraints** model dependencies
- solver-based **analyses** can be used to understand the configuration space better

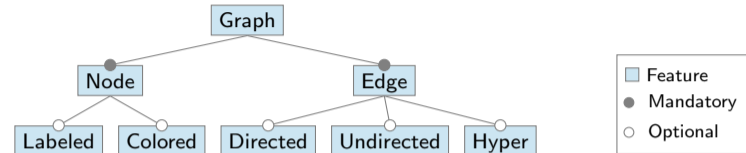


$\neg(\text{Directed} \wedge \text{Undirected})$ **feature conflict**
 $\text{Hyper} \rightarrow \text{Undirected}$ **missing implementation**
 $\text{Directed} \not\leftrightarrow (\text{Undirected} \wedge \text{Hyper})$ **well-defined #ifdef**

Modeling Features and their Dependencies

Feature Models

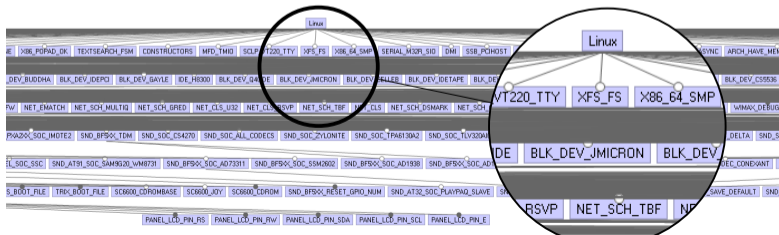
- tree models **features**
- cross-tree **constraints** model dependencies
- solver-based **analyses** can be used to understand the configuration space better



$\neg(\text{Directed} \wedge \text{Undirected})$ **feature conflict**
 $\text{Hyper} \rightarrow \text{Undirected}$ **missing implementation**
 $\text{Directed} \not\leftrightarrow (\text{Undirected} \wedge \text{Hyper})$ **well-defined #ifdef**

The Linux Kernel

- > 13000 features [2018]
- > 10^{700} products [2007]
- 114 dead features [2013]
- 151 reverse dependency bugs [2019]



Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis

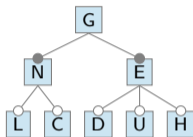


Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis



A Feature Model FM



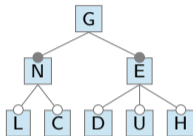
$\neg(D \wedge U)$
 $H \rightarrow U$
 $D \not\leftrightarrow (U \wedge H)$

Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis



A Feature Model FM



$\neg(D \wedge U)$
 $H \rightarrow U$
 $D \nleftrightarrow (U \wedge H)$

$\xrightarrow{\Phi}$

As a Formula $\Phi(FM)$

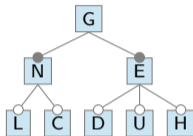
G
 $\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G)$
 $\wedge ((L \vee C) \rightarrow N)$
 $\wedge ((D \vee U \vee H) \rightarrow E)$
 $\wedge \neg(D \wedge U) \wedge (H \rightarrow U)$
 $\wedge (D \nleftrightarrow (U \wedge H))$

Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis



A Feature Model FM



$\neg(D \wedge U)$
 $H \rightarrow U$
 $D \nleftrightarrow (U \wedge H)$

$\xrightarrow{\Phi}$

As a Formula $\Phi(FM)$

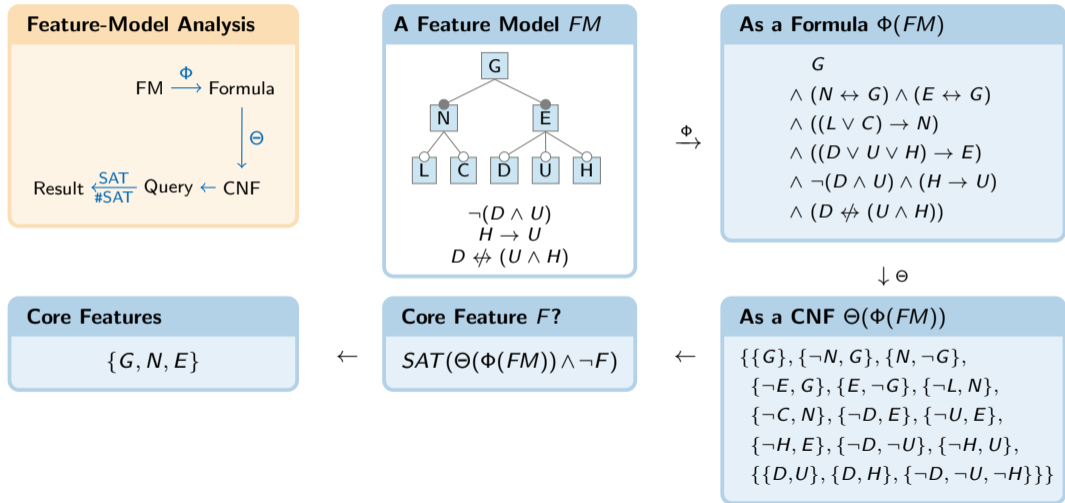
G
 $\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G)$
 $\wedge ((L \vee C) \rightarrow N)$
 $\wedge ((D \vee U \vee H) \rightarrow E)$
 $\wedge \neg(D \wedge U) \wedge (H \rightarrow U)$
 $\wedge (D \nleftrightarrow (U \wedge H))$

$\downarrow \Theta$

As a CNF $\Theta(\Phi(FM))$

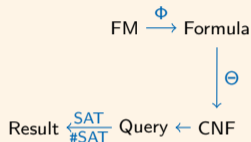
$\{\{G\}, \{\neg N, G\}, \{N, \neg G\},$
 $\{\neg E, G\}, \{E, \neg G\}, \{\neg L, N\},$
 $\{\neg C, N\}, \{\neg D, E\}, \{\neg U, E\},$
 $\{\neg H, E\}, \{\neg D, \neg U\}, \{\neg H, U\},$
 $\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$

Analyzing Feature Models with SAT and #SAT Solvers

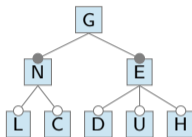


Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis



A Feature Model FM



$$\neg(D \wedge U)$$

$$H \rightarrow U$$

$$D \not\leftrightarrow (U \wedge H)$$

As a Formula $\Phi(FM)$

$$G$$

$$\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G)$$

$$\wedge ((L \vee C) \rightarrow N)$$

$$\wedge ((D \vee U \vee H) \rightarrow E)$$

$$\wedge \neg(D \wedge U) \wedge (H \rightarrow U)$$

$$\wedge (D \not\leftrightarrow (U \wedge H))$$

Φ

Θ

Core Features

$\{G, N, E\}$

Core Feature $F?$

$\text{SAT}(\Theta(\Phi(FM)) \wedge \neg F)$

Feature Model Cardinality

8

Products in $FM?$

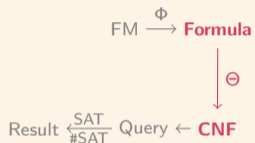
$\text{\#SAT}(\Theta(\Phi(FM)))$

As a CNF $\Theta(\Phi(FM))$

$\{\{G\}, \{\neg N, G\}, \{N, \neg G\},$
 $\{\neg E, G\}, \{E, \neg G\}, \{\neg L, N\},$
 $\{\neg C, N\}, \{\neg D, E\}, \{\neg U, E\},$
 $\{\neg H, E\}, \{\neg D, \neg U\}, \{\neg H, U\},$
 $\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$

Often Overlooked: Conjunctive Normal Form (CNF)

Feature-Model Analysis



From Formula ...

G

$$\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G)$$
$$\wedge ((L \vee C) \rightarrow N)$$
$$\wedge ((D \vee U \vee H) \rightarrow E)$$
$$\wedge \neg(D \wedge U) \wedge (H \rightarrow U)$$
$$\wedge (D \not\leftrightarrow (U \wedge H))$$

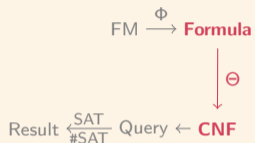
$\downarrow \Theta$

... to CNF

$$\{\{G\}, \{\neg N, G\}, \{N, \neg G\},$$
$$\{\neg E, G\}, \{E, \neg G\}, \{\neg L, N\},$$
$$\{\neg C, N\}, \{\neg D, E\}, \{\neg U, E\},$$
$$\{\neg H, E\}, \{\neg D, \neg U\}, \{\neg H, U\},$$
$$\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$$

Often Overlooked: Conjunctive Normal Form (CNF)

Feature-Model Analysis



Conjunctive Normal Form

- **conjunction** \wedge of **disjunctions** \vee of **literals** $X, \neg X$
- here: a set of **clauses**, which are sets of literals
- used by almost all solvers

From Formula ...

$$G$$
$$\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G)$$
$$\wedge ((L \vee C) \rightarrow N)$$
$$\wedge ((D \vee U \vee H) \rightarrow E)$$
$$\wedge \neg(D \wedge U) \wedge (H \rightarrow U)$$
$$\wedge (D \not\leftrightarrow (U \wedge H))$$

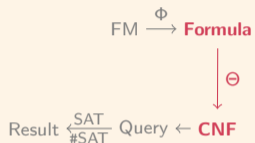
$\downarrow \ominus$

... to CNF

$$\{\{G\}, \{\neg N, G\}, \{N, \neg G\},$$
$$\{\neg E, G\}, \{E, \neg G\}, \{\neg L, N\},$$
$$\{\neg C, N\}, \{\neg D, E\}, \{\neg U, E\},$$
$$\{\neg H, E\}, \{\neg D, \neg U\}, \{\neg H, U\},$$
$$\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$$

Often Overlooked: Conjunctive Normal Form (CNF)

Feature-Model Analysis



Conjunctive Normal Form

- **conjunction** \wedge of **disjunctions** \vee of **literals** $X, \neg X$
- here: a set of **clauses**, which are sets of literals
- used by almost all solvers

From Formula ...

$$G$$
$$\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G)$$
$$\wedge ((L \vee C) \rightarrow N)$$
$$\wedge ((D \vee U \vee H) \rightarrow E)$$
$$\wedge \neg(D \wedge U) \wedge (H \rightarrow U)$$
$$\wedge (D \not\leftrightarrow (U \wedge H))$$

$\downarrow \Theta$

Our Goal: Raise Awareness for CNF Transformations

[ASE'22]

- how to **transform** feature-model formulas **into CNF**?
 \Rightarrow describe and classify CNF transformations
- does this **impact** the work of **practitioners and researchers**?
 \Rightarrow evaluate efficiency and correctness on feature models

... to CNF

$$\{\{G\}, \{\neg N, G\}, \{N, \neg G\},$$
$$\{\neg E, G\}, \{E, \neg G\}, \{\neg L, N\},$$
$$\{\neg C, N\}, \{\neg D, E\}, \{\neg U, E\},$$
$$\{\neg H, E\}, \{\neg D, \neg U\}, \{\neg H, U\},$$
$$\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$$

CNF Transformations

Distributive $\Theta = D$

apply laws of logic (**De Morgan's laws** and **distributivity**)

$$D \not\leftrightarrow (U \wedge H)$$

$$\xrightarrow{D} (D \vee (U \wedge H)) \wedge (\neg D \vee \neg (U \wedge H))$$

$$\xrightarrow{D} \{\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$$

- ✓ **equivalence**
SAT ✓, #SAT = 4
- ✓ easy to implement
- ✗ **exponential** complexity

CNF Transformations

Distributive $\Theta = D$

apply laws of logic (**De Morgan's laws** and **distributivity**)

$$D \not\leftrightarrow (U \wedge H)$$

$$\xrightarrow{D} (D \vee (U \wedge H)) \wedge (\neg D \vee \neg(U \wedge H))$$

$$\xrightarrow{D} \{\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$$

- ✓ **equivalence**
SAT ✓, #SAT = 4
- ✓ easy to implement
- ✗ **exponential** complexity

Tseitin $\Theta = T$ [83]

abbreviate a subformula ϕ with an auxiliary variable $x_\phi \leftrightarrow \phi$

$$D \not\leftrightarrow (U \wedge H)$$

$$\xrightarrow{T} (D \not\leftrightarrow x) \wedge x \leftrightarrow (U \wedge H)$$

$$\xrightarrow{D} \{\{D, x\}, \{\neg D, \neg x\}, \{\neg x, U\}, \{\neg x, H\}, \{\neg U, \neg H, x\}\}$$

- ✓ **quasi-equivalence**
SAT ✓, #SAT = 4
- ✓ **linear** complexity
- ✗ take care of new variables

CNF Transformations

Distributive $\Theta = D$

apply laws of logic (**De Morgan's laws** and **distributivity**)

$$D \not\leftrightarrow (U \wedge H)$$

$$\xrightarrow{D} (D \vee (U \wedge H)) \wedge (\neg D \vee \neg(U \wedge H))$$

$$\xrightarrow{D} \{\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$$

- ✓ **equivalence**
SAT ✓, #SAT = 4
- ✓ easy to implement
- ✗ **exponential** complexity

Tseitin $\Theta = T$ [83]

abbreviate a subformula ϕ with an auxiliary variable $x_\phi \leftrightarrow \phi$

$$D \not\leftrightarrow (U \wedge H)$$

$$\xrightarrow{T} (D \leftrightarrow x) \wedge x \leftrightarrow (U \wedge H)$$

$$\xrightarrow{D} \{\{D, x\}, \{\neg D, \neg x\}, \{\neg x, U\}, \{\neg x, H\}, \{\neg U, \neg H, x\}\}$$

- ✓ **quasi-equivalence**
SAT ✓, #SAT = 4
- ✓ **linear** complexity
- ✗ take care of new variables

Plaisted-Greenbaum $\Theta = PG$ [86]

abbreviate a subformula ϕ with an auxiliary variable $x_\phi \rightarrow \phi$

$$D \not\leftrightarrow (U \wedge H)$$

$$\xrightarrow{PG} (D \leftrightarrow x) \wedge x \rightarrow (U \wedge H)$$

$$\xrightarrow{D} \{\{D, x\}, \{\neg D, \neg x\}, \{\neg x, U\}, \{\neg x, H\}\}$$

- ✓ **equi-assignability** SAT ✓
- ✓ **linear** complexity $< T$
- ✗ **equi-countability** #SAT = 5

Evaluation

Research Questions

RQ 1 **efficiency** of CNF transformations?

RQ 2 CNF transformation → **efficiency** of analyses?

RQ 3 CNF transformation → **correctness** of analyses?

Evaluation

Research Questions

- RQ 1 **efficiency** of CNF transformations?
- RQ 2 CNF transformation → **efficiency** of analyses?
- RQ 3 CNF transformation → **correctness** of analyses?

Experimental Setup

- 22 configurable software systems
- 3 CNF transformation tools
- 23 SAT and #SAT solvers

Evaluation

Research Questions

- RQ 1 **efficiency** of CNF transformations?
- RQ 2 CNF transformation → **efficiency** of analyses?
- RQ 3 CNF transformation → **correctness** of analyses?

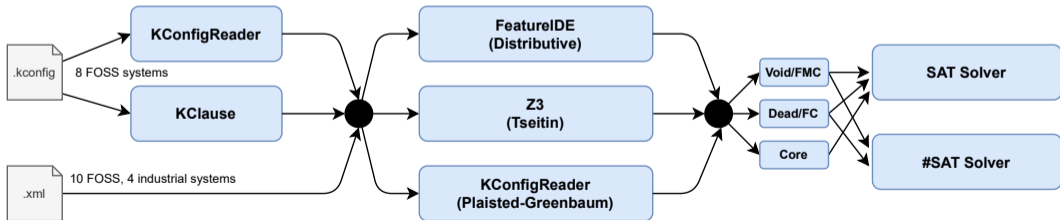
Experimental Setup

- 22 configurable software systems
- 3 CNF transformation tools
- 23 SAT and #SAT solvers

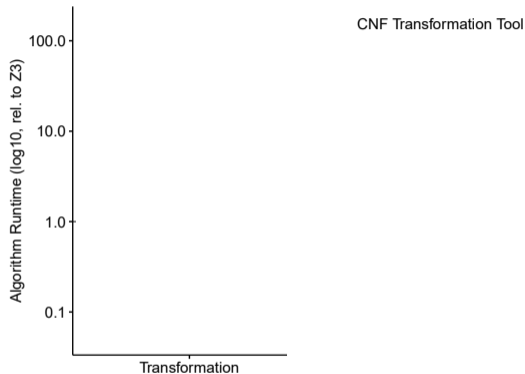
Stage 1: Formula Extraction

Stage 2: CNF Transformation

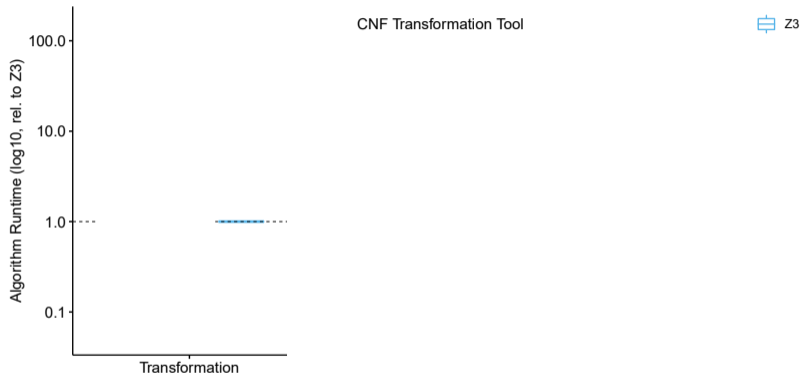
Stage 3: Automated Analysis



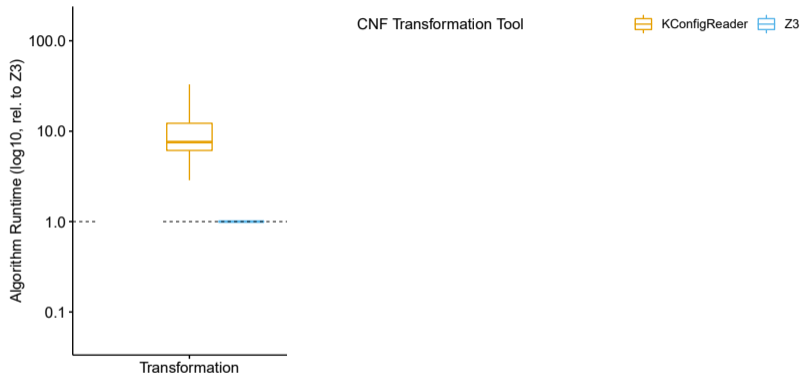
Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



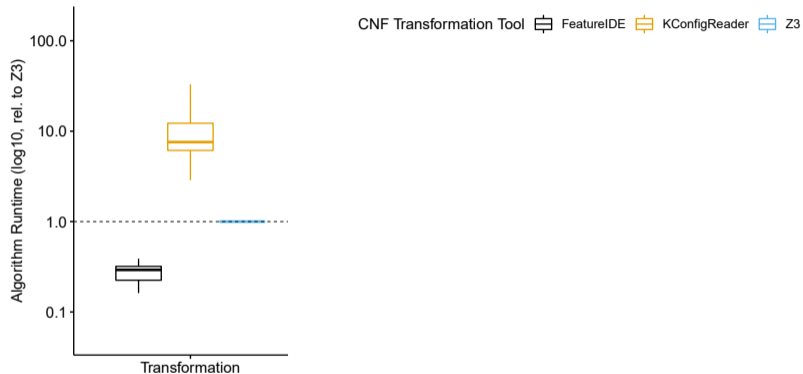
Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



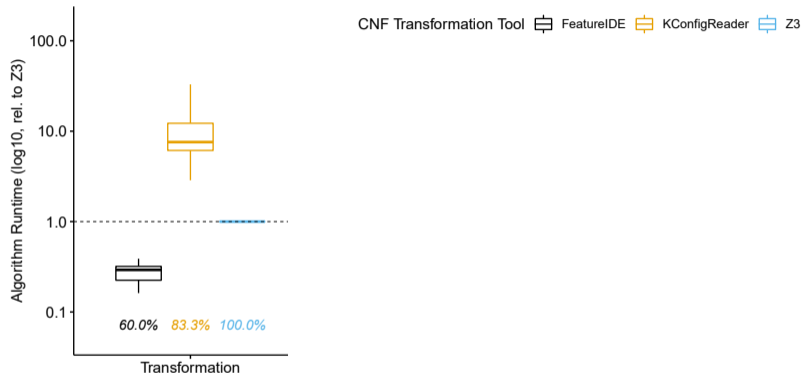
Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)

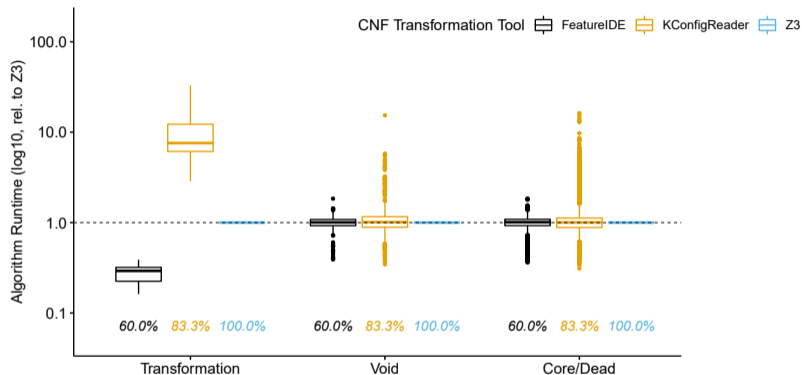


Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



RQ 1: *D* often fails,
tools differ significantly

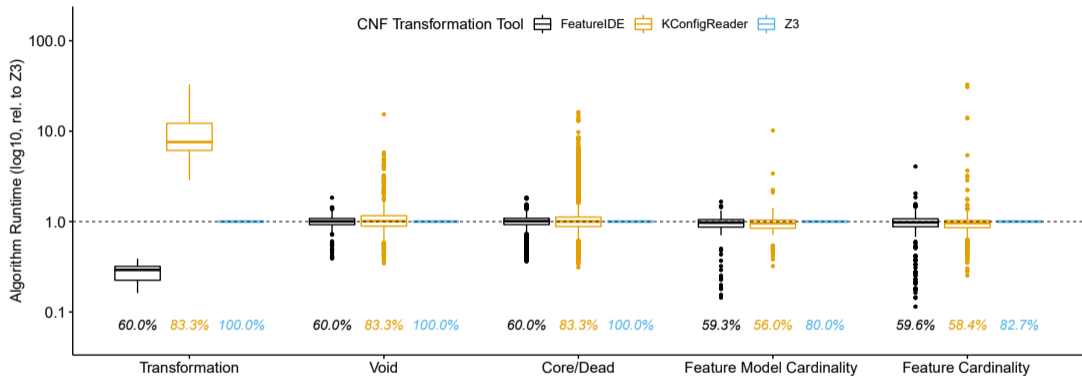
Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



RQ 1: *D* often fails, tools differ significantly

RQ 2 (SAT): almost all calls succeed, solve time varies by factor 0.31–16.27

Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



RQ 1: *D* often fails, tools differ significantly

RQ 2 (SAT): almost all calls succeed, solve time varies by factor 0.31–16.27

RQ 2 (#SAT): 81.6% of calls succeed, solve time varies by factor 0.11–32.7

Correctness of #SAT-Based Analyses (RQ 3)

How Many Valid Configurations in BusyBox 1.35.0?

FeatureIDE (Distributive) says:

```
47842046044873008384 13517649496919484532 17980737275928522342 35800557238486733859
78971326945465595845 72908124465341304467 84732350200161989505 38440744692509401678
9913600000000000000 000000000
```

Tseitin (Z3) says:

```
47842046044873008384 13517649496919484532 17980737275928522342 35800557238486733859
78971326945465595845 72908124465341304467 84732350200161989505 38440744692509401678
9913600000000000000 000000000
```

KConfigReader (Plaisted-Greenbaum) says:

```
15751357446718468213 90135655996554596226 77965648288591932216 37368937605749145888
80850342078354075798 38471914912986177301 71318442740266744344 68038795993960163378
1860761600000000000 000000000 0 ⇒ off by factor 3.292
```

Correctness of #SAT-Based Analyses (RQ 3)

How Many Valid Configurations in BusyBox 1.35.0?

FeatureIDE (Distributive) says:

```
47842046044873008384 13517649496919484532 17980737275928522342 35800557238486733859
78971326945465595845 72908124465341304467 84732350200161989505 38440744692509401678
9913600000000000000 000000000
```

Tsetin (Z3) says:

```
47842046044873008384 13517649496919484532 17980737275928522342 35800557238486733859
78971326945465595845 72908124465341304467 84732350200161989505 38440744692509401678
9913600000000000000 000000000
```

KConfigReader (Plaisted-Greenbaum) says:

```
15751357446718468213 90135655996554596226 77965648288591932216 37368937605749145888
80850342078354075798 38471914912986177301 71318442740266744344 68038795993960163378
1860761600000000000 000000000 0 ⇒ off by factor 3.292
```

RQ 3

- with PG , $\approx 70\%$ of #SAT calls return **incorrect results**
- incorrect by factor ≈ 3 (median)
- incorrect by factor $\approx 10^{77}$ (worst)

Correctness of #SAT-Based Analyses (RQ 3)

How Many Valid Configurations in BusyBox 1.35.0?

FeatureIDE (Distributive) says:

```
47842046044873008384 13517649496919484532 17980737275928522342 35800557238486733859
78971326945465595845 72908124465341304467 84732350200161989505 38440744692509401678
9913600000000000000 000000000
```

Tseitin (Z3) says:

```
47842046044873008384 13517649496919484532 17980737275928522342 35800557238486733859
78971326945465595845 72908124465341304467 84732350200161989505 38440744692509401678
9913600000000000000 000000000
```

KConfigReader (Plaisted-Greenbaum) says:

```
15751357446718468213 90135655996554596226 77965648288591932216 37368937605749145888
80850342078354075798 38471914912986177301 71318442740266744344 68038795993960163378
1860761600000000000 000000000 0 ⇒ off by factor 3.292
```

RQ 3

- with PG , $\approx 70\%$ of #SAT calls return **incorrect results**
- incorrect by factor ≈ 3 (median)
- incorrect by factor $\approx 10^{77}$ (worst)

Our Recommendations

- RQ 1** D for small, T for large models
- RQ 2** largely depends on the model
⇒ future work
- RQ 3** do not use PG for #SAT

Conclusion

The Impact of CNF Transformations on Feature-Model Analyses

Distributive

apply laws of logic

- ✓ **equivalence**
- ✓ easy to implement
- ✗ **exponential** complexity

FeatureIDE

often fails on large models

Tseitin

abbreviate ϕ with $x_\phi \leftrightarrow \phi$

- ✓ **quasi-equivalence**
- ✓ **linear** complexity
- ✗ take care of new variables

Z3

succeeds correctly on all models

Plaisted-Greenbaum

abbreviate ϕ with $x_\phi \rightarrow \phi$

- ✓ **equi-assignability**
- ✓ **linear** complexity
- ✗ **equi-countability**

KConfigReader

often incorrect for #SAT calls

find out more:



github.com/ekuiter/tseitin-or-not-tseitin

