



[Elbe, Magdeburg]

## Tseitin or not Tseitin?

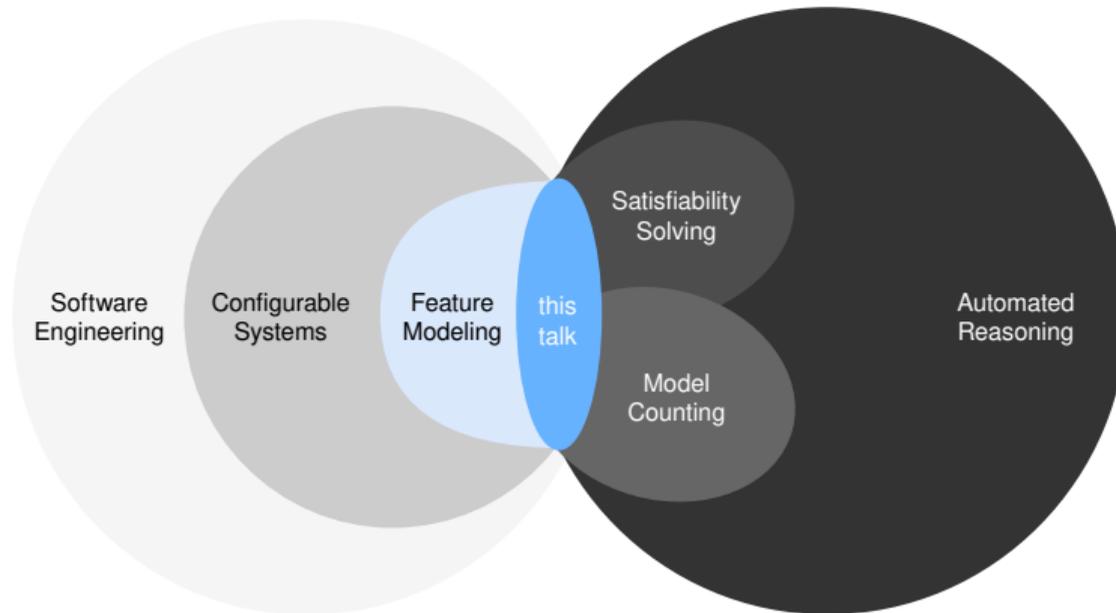
### The Impact of CNF Transformations on Feature-Model Analyses

MCW@SAT 2023 (ASE 2022) — July 4 — Alghero, Italy

Elias Kuitert<sup>1</sup>, Sebastian Krieter<sup>2</sup>, Chico Sundermann<sup>2</sup>, Thomas Thüm<sup>2</sup>, Gunter Saake<sup>1</sup>

University of Magdeburg<sup>1</sup>, Ulm<sup>2</sup>

# Software Engineering Meets Automated Reasoning



# 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

## A Labeled Directed Graph

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

**Configuration**

**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

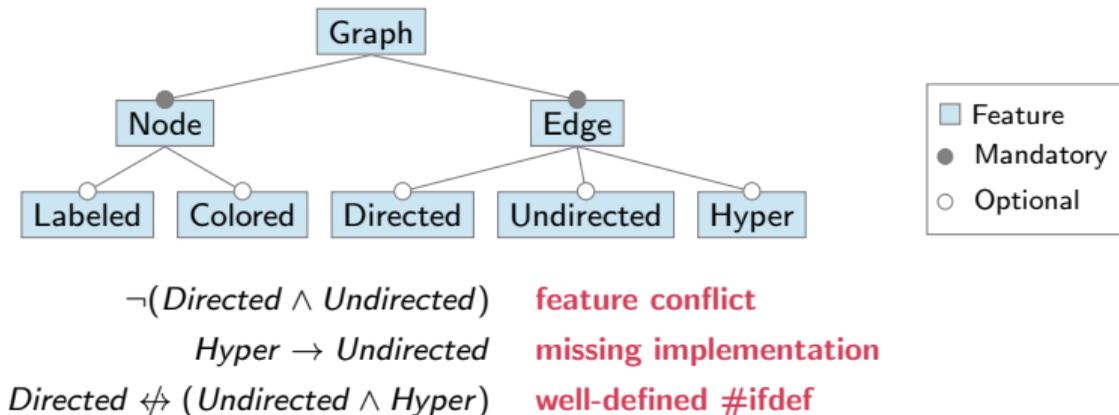
```
class Node {  
  std::string color;  
};  
  
class Edge {  
  std::set<Node> nodes;  
};
```

### Product Implementation

# Modeling Features and their Dependencies

## Feature Models

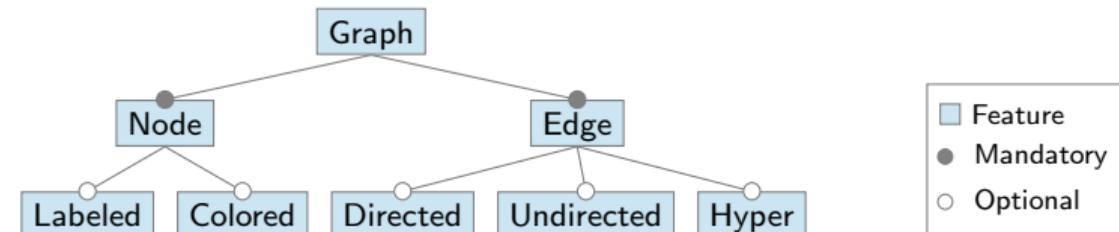
- tree models **features**
- cross-tree **constraints** model dependencies
- solver-based **analyses** can be used to understand the configuration space better



# 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



$\neg(Directed \wedge Undirected)$

**feature conflict**

$Hyper \rightarrow Undirected$

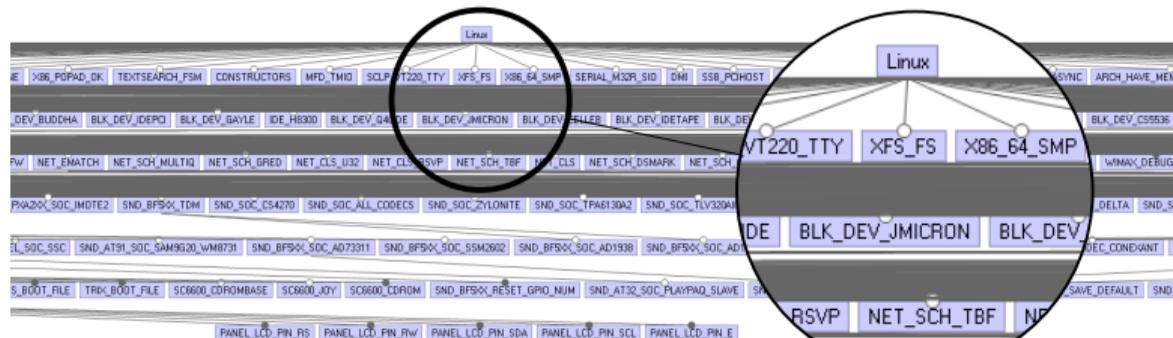
**missing implementation**

$Directed \not\leftrightarrow (Undirected \wedge Hyper)$

**well-defined #ifdef**

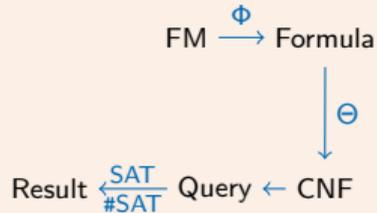
## The Linux Kernel

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



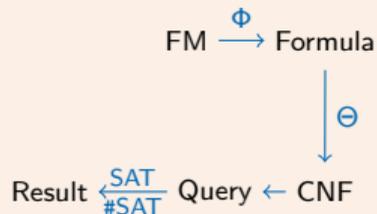
# Analyzing Feature Models with SAT and #SAT Solvers

## Feature-Model Analysis

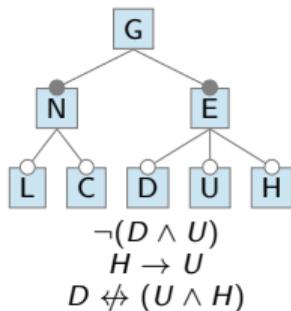


# Analyzing Feature Models with SAT and #SAT Solvers

## Feature-Model Analysis

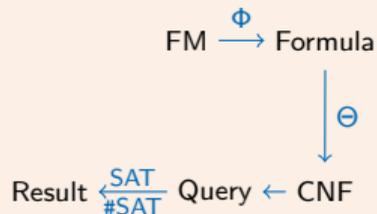


## A Feature Model $FM$

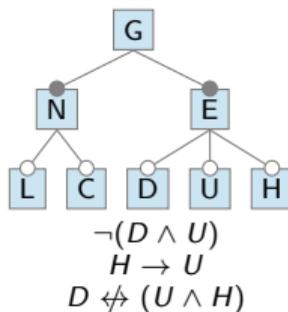


# Analyzing Feature Models with SAT and #SAT Solvers

## Feature-Model Analysis



## A Feature Model $FM$



$\Phi$

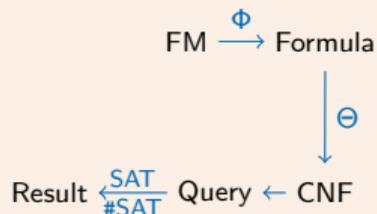
## As a Formula $\Phi(FM)$

Logical formula representation of the Feature Model  $FM$ :

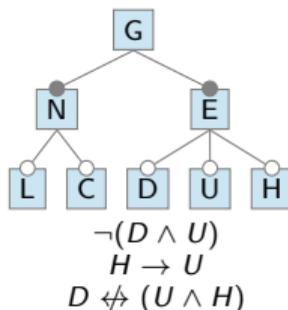
$$\begin{aligned} &G \\ &\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G) \\ &\wedge ((L \vee C) \rightarrow N) \\ &\wedge ((D \vee U \vee H) \rightarrow E) \\ &\wedge \neg(D \wedge U) \wedge (H \rightarrow U) \\ &\wedge (D \not\leftrightarrow (U \wedge H)) \end{aligned}$$

# Analyzing Feature Models with SAT and #SAT Solvers

## Feature-Model Analysis



## A Feature Model $FM$



$\Phi$

## As a Formula $\Phi(FM)$

Logical formula representation of the Feature Model:

$$\begin{aligned} &G \\ &\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G) \\ &\wedge ((L \vee C) \rightarrow N) \\ &\wedge ((D \vee U \vee H) \rightarrow E) \\ &\wedge \neg(D \wedge U) \wedge (H \rightarrow U) \\ &\wedge (D \not\leftrightarrow (U \wedge H)) \end{aligned}$$

$\Theta$

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

Conversion of the formula into Conjunctive Normal Form (CNF):

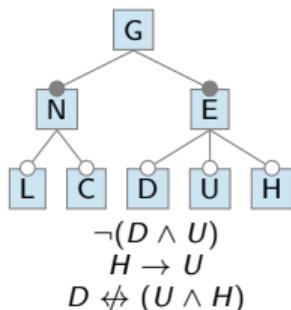
$$\begin{aligned} &\{ \{G\}, \{ \neg N, G \}, \{ N, \neg G \}, \\ &\{ \neg E, G \}, \{ E, \neg G \}, \{ \neg L, N \}, \\ &\{ \neg C, N \}, \{ \neg D, E \}, \{ \neg U, E \}, \\ &\{ \neg H, E \}, \{ \neg D, \neg U \}, \{ \neg H, U \}, \\ &\{ \{ D, U \}, \{ D, H \}, \{ \neg D, \neg U, \neg H \} \} \end{aligned}$$

# Analyzing Feature Models with SAT and #SAT Solvers

## Feature-Model Analysis



## A Feature Model $FM$



## As a Formula $\Phi(FM)$

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

$\downarrow \Theta$

## Core Features

$\{G, N, E\}$

## Core Feature $F$ ?

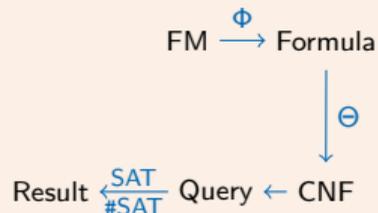
$SAT(\Theta(\Phi(FM)) \wedge \neg F)$

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

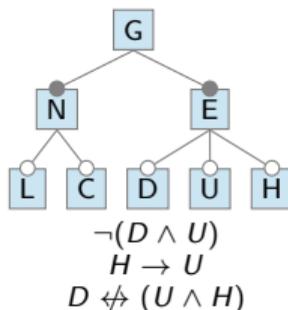
$\{\{G\}, \{\neg N, G\}, \{N, \neg G\},$   
 $\{\neg E, G\}, \{E, \neg G\}, \{\neg L, N\},$   
 $\{\neg C, N\}, \{\neg D, E\}, \{\neg U, E\},$   
 $\{\neg H, E\}, \{\neg D, \neg U\}, \{\neg H, U\},$   
 $\{\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}\}$

# Analyzing Feature Models with SAT and #SAT Solvers

## Feature-Model Analysis



## A Feature Model $FM$



## As a Formula $\Phi(FM)$

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

$\Phi$

$\Theta$

## Core Features

$\{G, N, E\}$

## Core Feature $F$ ?

$SAT(\Theta(\Phi(FM)) \wedge \neg F)$

## Feature Model Cardinality

8

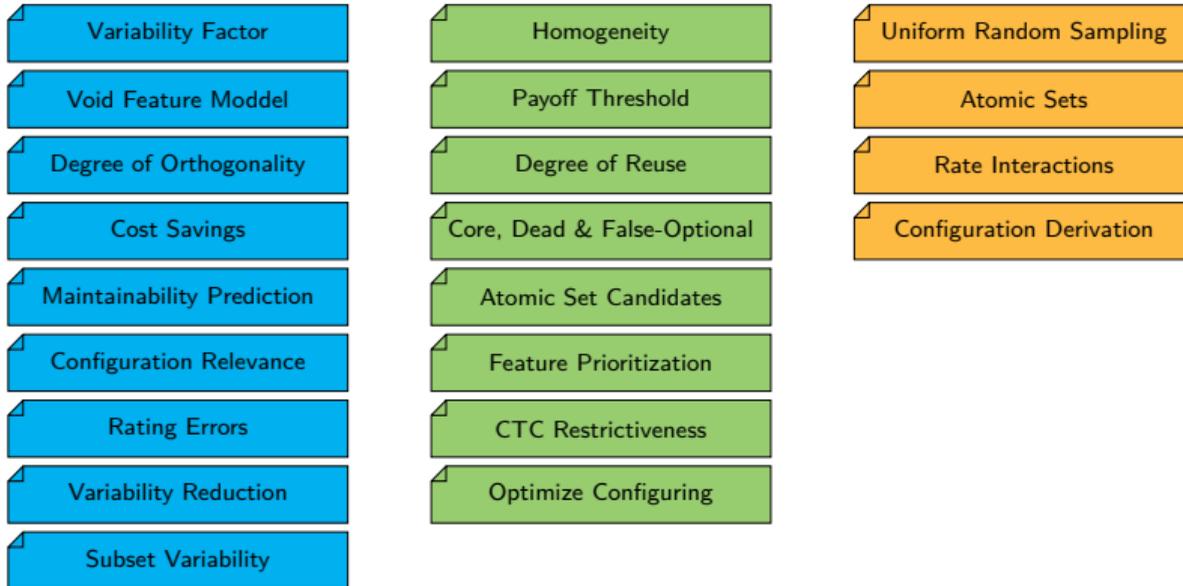
## Products in $FM$ ?

$\#SAT(\Theta(\Phi(FM)))$

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

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

# Feature Modeling Meets Model Counting



# 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 ...

$$\begin{aligned} &G \\ &\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G) \\ &\wedge ((L \vee C) \rightarrow N) \\ &\wedge ((D \vee U \vee H) \rightarrow E) \\ &\wedge \neg(D \wedge U) \wedge (H \rightarrow U) \\ &\wedge (D \not\leftrightarrow (U \wedge H)) \end{aligned}$$

$\downarrow \ominus$

## 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

$$\begin{aligned} &\{\{G\}, \{\neg N, G\}, \{N, \neg G\}, \\ &\{\neg E, G\}, \{E, \neg G\}, \{\neg L, N\}, \\ &\{\neg C, N\}, \{\neg D, E\}, \{\neg U, E\}, \\ &\{\neg H, E\}, \{\neg D, \neg U\}, \{\neg H, U\}, \\ &\{\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\} \end{aligned}$$

# 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  $\phi$  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  $\phi$  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

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

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

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

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

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

- ✓ **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?

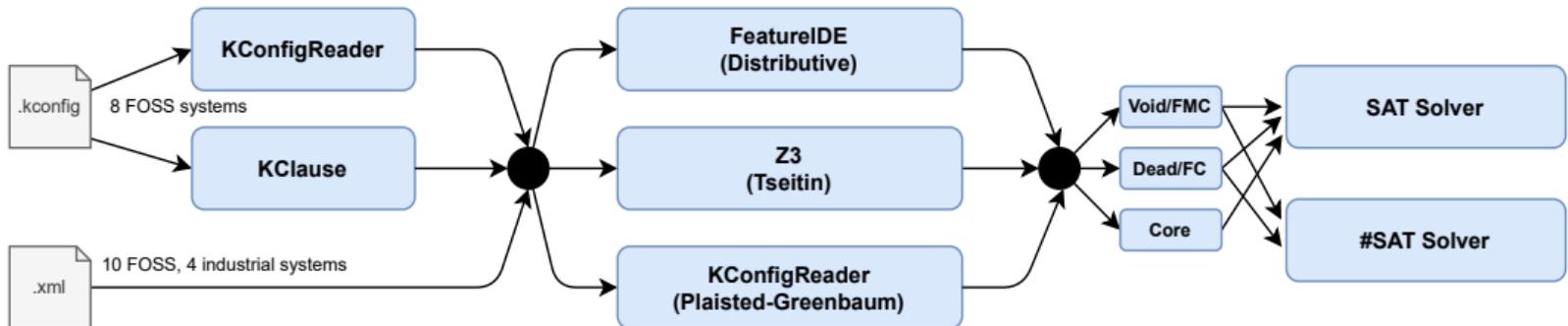
## Experimental Setup

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

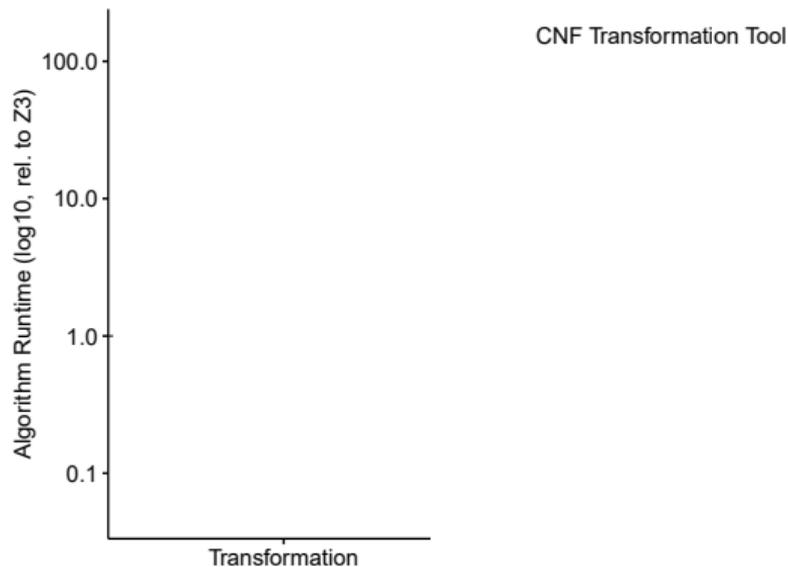
### Stage 1: Formula Extraction

### Stage 2: CNF Transformation

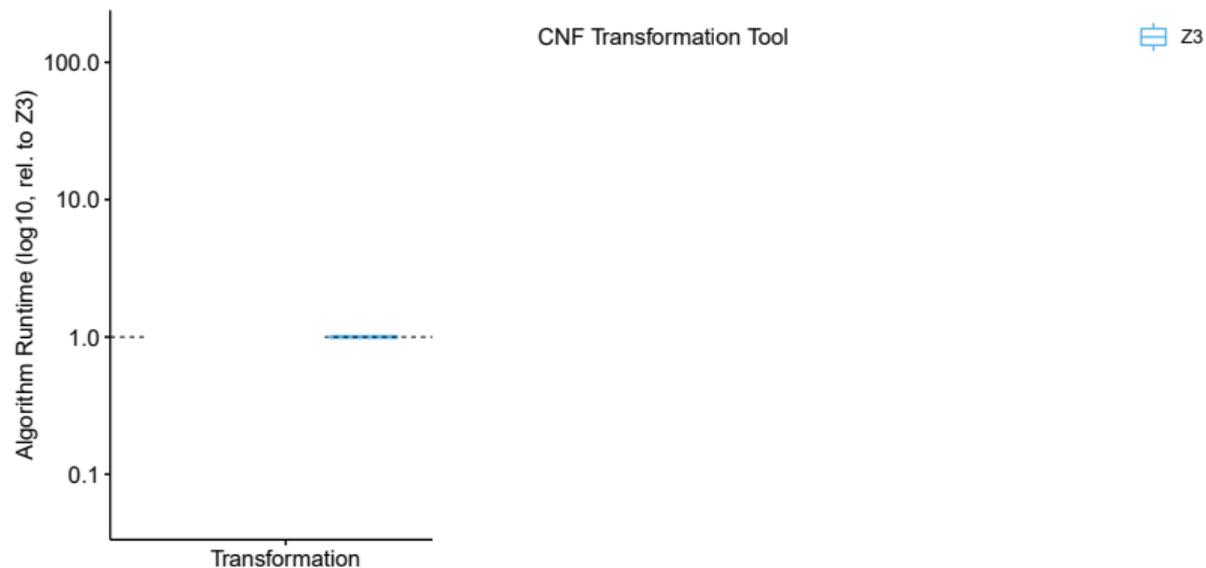
### Stage 3: Automated Analysis



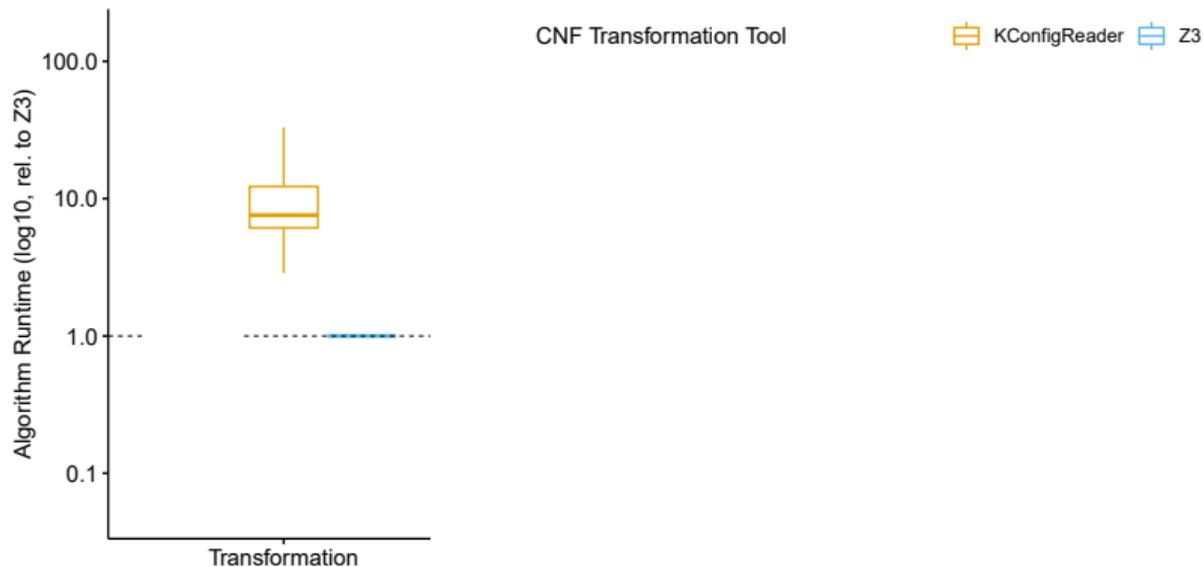
# Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



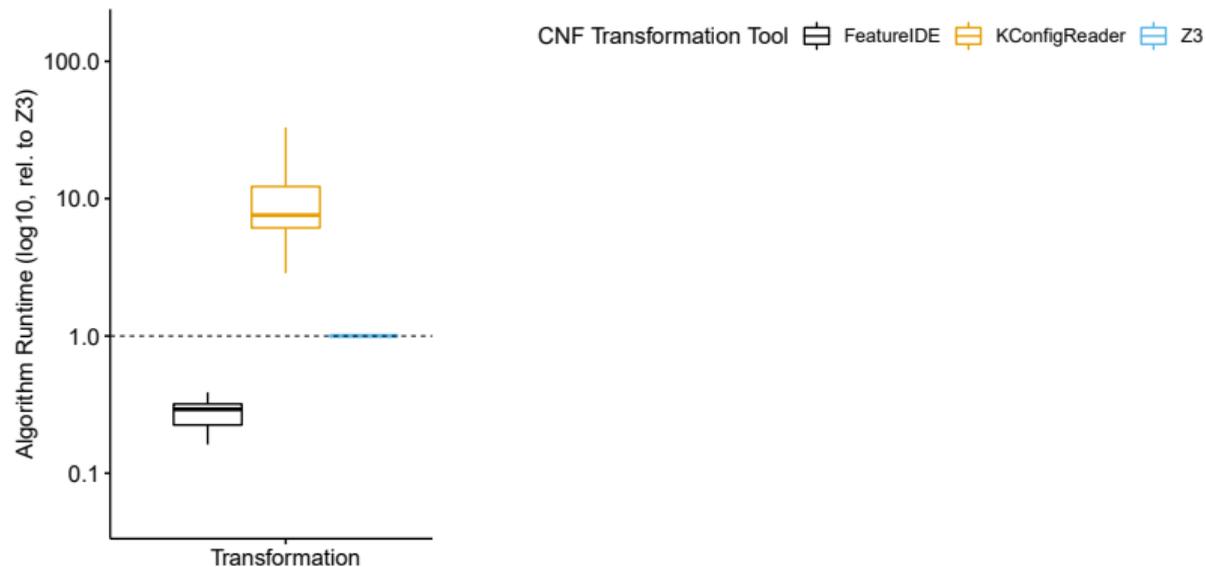
# Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



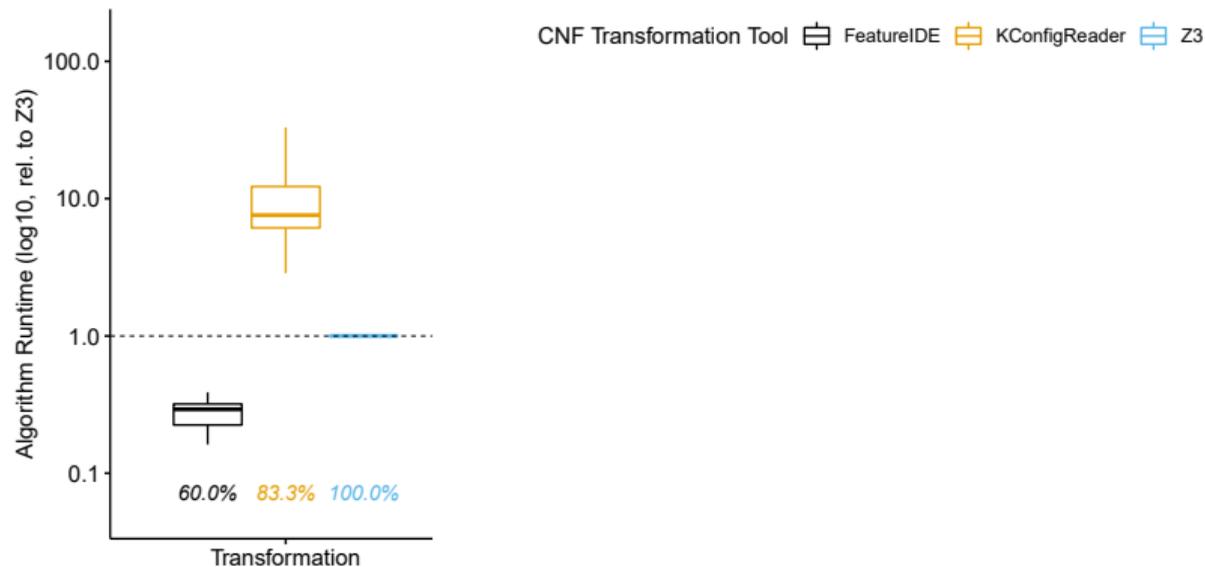
# Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



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

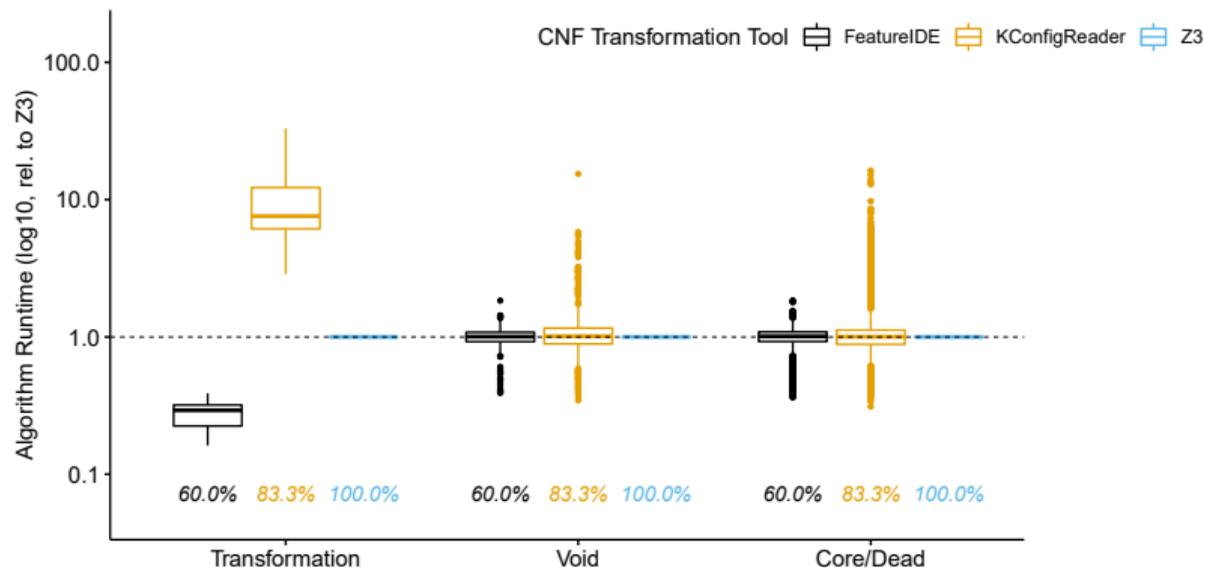


# Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



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

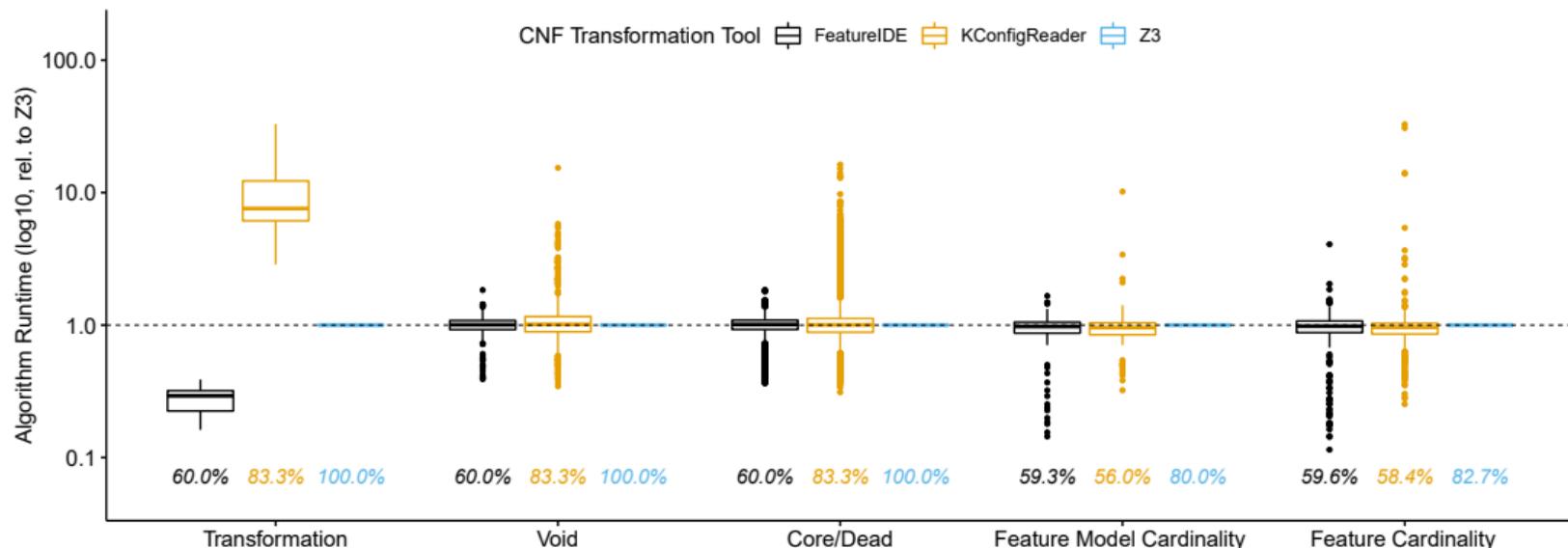
# Efficiency of CNF Transformations (RQ 1) and Analyses (RQ 2)



**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
99136000000000000000 000000000
```

Tsetin (Z3) says:

```
47842046044873008384 13517649496919484532 17980737275928522342 35800557238486733859
78971326945465595845 72908124465341304467 84732350200161989505 38440744692509401678
99136000000000000000 000000000
```

KConfigReader (Plaisted-Greenbaum) says:

```
15751357446718468213 90135655996554596226 77965648288591932216 37368937605749145888
80850342078354075798 38471914912986177301 71318442740266744344 68038795993960163378
18607616000000000000 000000000 0 ⇒ off by factor 3.292
```

## RQ 3

- with *PG*,  $\approx 70\%$  of #SAT calls return **incorrect results**
- incorrect by factor  $\approx 3$  (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
99136000000000000000 000000000
```

Tsetin (Z3) says:

```
47842046044873008384 13517649496919484532 17980737275928522342 35800557238486733859
78971326945465595845 72908124465341304467 84732350200161989505 38440744692509401678
99136000000000000000 000000000
```

KConfigReader (Plaisted-Greenbaum) says:

```
15751357446718468213 90135655996554596226 77965648288591932216 37368937605749145888
80850342078354075798 38471914912986177301 71318442740266744344 68038795993960163378
18607616000000000000 000000000 0 ⇒ off by factor 3.292
```

## RQ 3

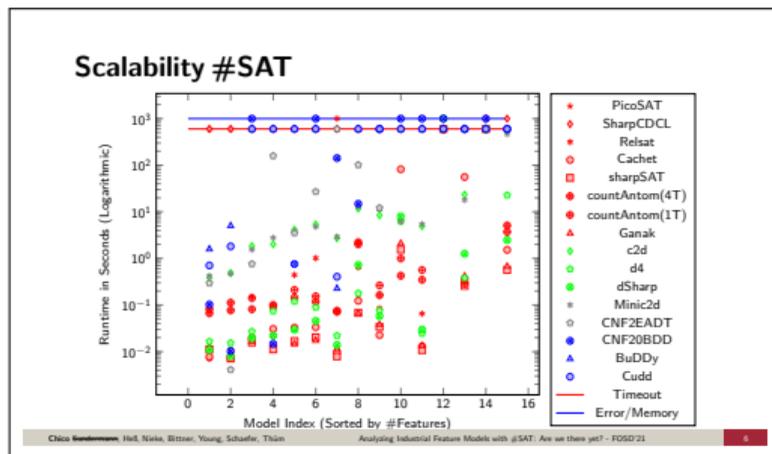
- with  $PG$ ,  $\approx 70\%$  of #SAT calls return **incorrect results**
- incorrect by factor  $\approx 3$  (median)
- incorrect by factor  $\approx 10^{77}$  (worst)

## 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

# Perspective: Model Counting Meets Feature Modeling?

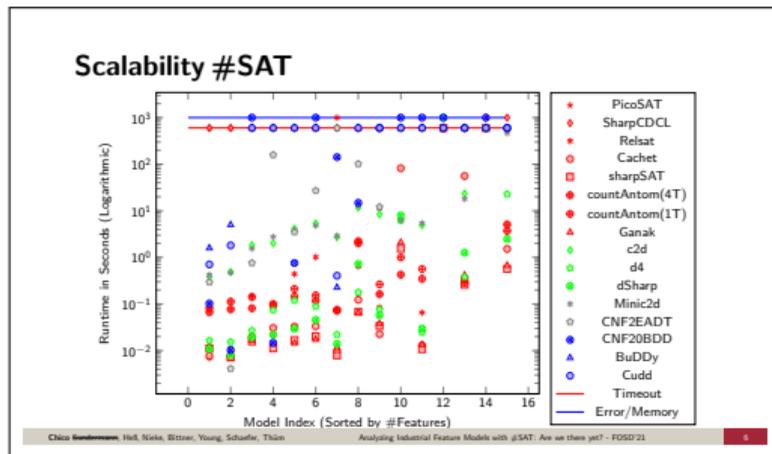
# Perspective: Model Counting Meets Feature Modeling?



## Compiling to d-DNNFs [Chico Sundermann]

works on most models (but not Linux/Automotive)  
15 software systems, 130 models (some industrial)

# Perspective: Model Counting Meets Feature Modeling?



6 Tobias Heß, C. Sundermann, T. Thüm | Binary Decision Diagrams in Product-Line Analysis | FOSD Online Meeting 2021 | 15.04.2021

### And today?

**Feature models have “evolved”**

- ▶ 1-2 orders of magnitude larger
- ▶ Often: #features  $\approx$  #constraints
- ▶ ECR: 5% - 95%

**BDD tooling has not “evolved”**

- ▶ BuDDy (1996) and CUDD (1995)
- ▶ Expert knowledge required
- ▶ Not included in current frameworks

#### Evaluating #SAT Solvers on Industrial Feature Models

Chico Sundermann, Technische Universität Braunschweig | Thüm, Thüm, University of Oldenburg | Heß, Schaefer, Technische Universität Braunschweig

116 CDL feature models

- ▶ 1,178 - 1,408 features
- ▶ 816 - 956 constraints
- ▶ ECR: 80% - 90%

0% success rate

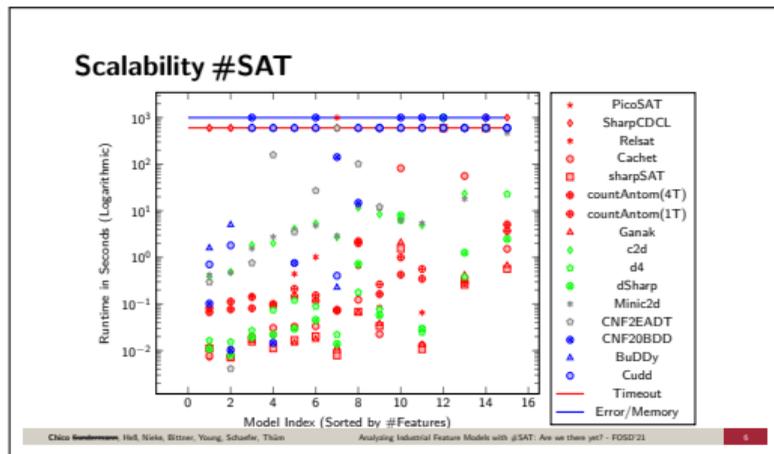
## Compiling to d-DNNFs [Chico Sundermann]

works on most models (but not Linux/Automotive)  
15 software systems, 130 models (some industrial)

## Compiling to BDDs [Tobias Heß]

works on most small models (but not Linux/CDL)  
largely depends on variable ordering

# Perspective: Model Counting Meets Feature Modeling?



6 Tobias Heß, C. Sundermann, T. Thüm | Binary Decision Diagrams in Product-Line Analysis | FOSD Online Meeting 2021 | 15.04.2021

## And today?

**Feature models have “evolved”**

- ▶ 1-2 orders of magnitude larger
- ▶ Often: #features  $\approx$  #constraints
- ▶ ECR: 5% - 95%

**BDD tooling has not “evolved”**

- ▶ BuDDy (1996) and CUDD (1995)
- ▶ Expert knowledge required
- ▶ Not included in current frameworks

**Evaluating #SAT Solvers on Industrial Feature Models**

Chico Sundermann, Tobiashe Universität Braunschweig | Thüm, Thüm, University of Oldenburg | Schaefer, Tobiashe Universität Braunschweig

116 CDL feature models

- ▶ 1,178 - 1,408 features
- ▶ 816 - 956 constraints
- ▶ ECR: 80% - 90%

0% success rate

## Compiling to d-DNNFs [Chico Sundermann]

works on most models (but not Linux/Automotive)  
15 software systems, 130 models (some industrial)

## Compiling to BDDs [Tobias Heß]

works on most small models (but not Linux/CDL)  
largely depends on variable ordering

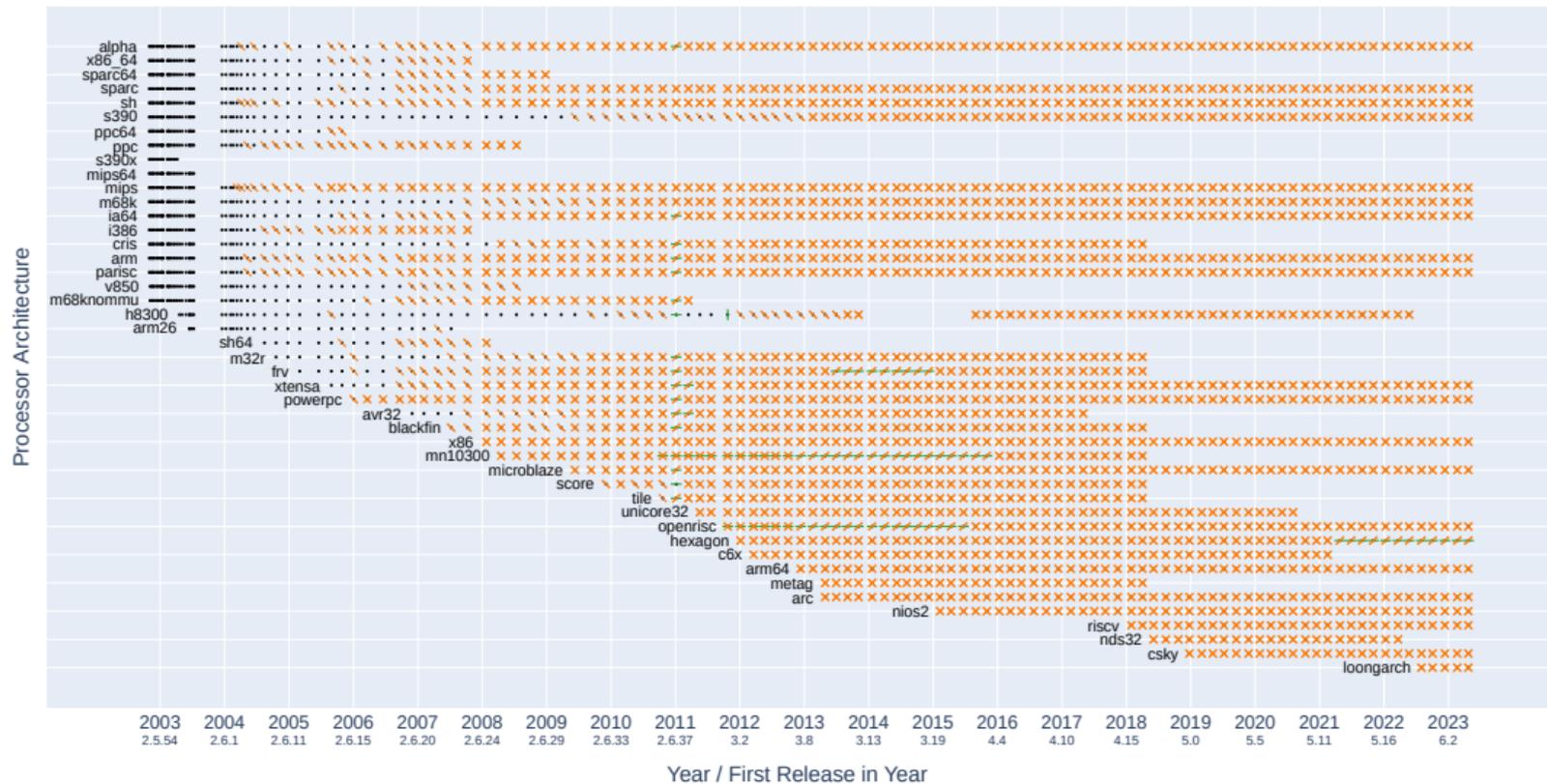
**our opinion:** feature models are interesting, diverse, and potentially hard problem instances for benchmarking (#)SAT and knowledge compilation. maybe they could contribute to the SAT and MC competition as well?

# Perspective: How Hard can Linux be, Really?

# perspective: How far can Linux be, Really?

```
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108] [109] [110] [111] [112] [113] [114] [115] [116] [117] [118] [119] [120] [121] [122] [123] [124] [125] [126] [127] [128] [129] [130] [131] [132] [133] [134] [135] [136] [137] [138] [139] [140] [141] [142] [143] [144] [145] [146] [147] [148] [149] [150] [151] [152] [153] [154] [155] [156] [157] [158] [159] [160] [161] [162] [163] [164] [165] [166] [167] [168] [169] [170] [171] [172] [173] [174] [175] [176] [177] [178] [179] [180] [181] [182] [183] [184] [185] [186] [187] [188] [189] [190] [191] [192] [193] [194] [195] [196] [197] [198] [199] [200] [201] [202] [203] [204] [205] [206] [207] [208] [209] [210] [211] [212] [213] [214] [215] [216] [217] [218] [219] [220] [221] [222] [223] [224] [225] [226] [227] [228] [229] [230] [231] [232] [233] [234] [235] [236] [237] [238] [239] [240] [241] [242] [243] [244] [245] [246] [247] [248] [249] [250] [251] [252] [253] [254] [255] [256] [257] [258] [259] [260] [261] [262] [263] [264] [265] [266] [267] [268] [269] [270] [271] [272] [273] [274] [275] [276] [277] [278] [279] [280] [281] [282] [283] [284] [285] [286] [287] [288] [289] [290] [291] [292] [293] [294] [295] [296] [297] [298] [299] [300] [301] [302] [303] [304] [305] [306] [307] [308] [309] [310] [311] [312] [313] [314] [315] [316] [317] [318] [319] [320] [321] [322] [323] [324] [325] [326] [327] [328] [329] [330] [331] [332] [333] [334] [335] [336] [337] [338] [339] [340] [341] [342] [343] [344] [345] [346] [347] [348] [349] [350] [351] [352] [353] [354] [355] [356] [357] [358] [359] [360] [361] [362] [363] [364] [365] [366] [367] [368] [369] [370] [371] [372] [373] [374] [375] [376] [377] [378] [379] [380] [381] [382] [383] [384] [385] [386] [387] [388] [389] [390] [391] [392] [393] [394] [395] [396] [397] [398] [399] [400] [401] [402] [403] [404] [405] [406] [407] [408] [409] [410] [411] [412] [413] [414] [415] [416] [417] [418] [419] [420] [421] [422] [423] [424] [425] [426] [427] [428] [429] [430] [431] [432] [433] [434] [435] [436] [437] [438] [439] [440] [441] [442] [443] [444] [445] [446] [447] [448] [449] [450] [451] [452] [453] [454] [455] [456] [457] [458] [459] [460] [461] [462] [463] [464] [465] [466] [467] [468] [469] [470] [471] [472] [473] [474] [475] [476] [477] [478] [479] [480] [481] [482] [483] [484] [485] [486] [487] [488] [489] [490] [491] [492] [493] [494] [495] [496] [497] [498] [499] [500] [501] [502] [503] [504] [505] [506] [507] [508] [509] [510] [511] [512] [513] [514] [515] [516] [517] [518] [519] [520] [521] [522] [523] [524] [525] [526] [527] [528] [529] [530] [531] [532] [533] [534] [535] [536] [537] [538] [539] [540] [541] [542] [543] [544] [545] [546] [547] [548] [549] [550] [551] [552] [553] [554] [555] [556] [557] [558] [559] [560] [561] [562] [563] [564] [565] [566] [567] [568] [569] [570] [571] [572] [573] [574] [575] [576] [577] [578] [579] [580] [581] [582] [583] [584] [585] [586] [587] [588] [589] [590] [591] [592] [593] [594] [595] [596] [597] [598] [599] [600] [601] [602] [603] [604] [605] [606] [607] [608] [609] [610] [611] [612] [613] [614] [615] [616] [617] [618] [619] [620] [621] [622] [623] [624] [625] [626] [627] [628] [629] [630] [631] [632] [633] [634] [635] [636] [637] [638] [639] [640] [641] [642] [643] [644] [645] [646] [647] [648] [649] [650] [651] [652] [653] [654] [655] [656] [657] [658] [659] [660] [661] [662] [663] [664] [665] [666] [667] [668] [669] [670] [671] [672] [673] [674] [675] [676] [677] [678] [679] [680] [681] [682] [683] [684] [685] [686] [687] [688] [689] [690] [691] [692] [693] [694] [695] [696] [697] [698] [699] [700] [701] [702] [703] [704] [705] [706] [707] [708] [709] [710] [711] [712] [713] [714] [715] [716] [717] [718] [719] [720] [721] [722] [723] [724] [725] [726] [727] [728] [729] [730] [731] [732] [733] [734] [735] [736] [737] [738] [739] [740] [741] [742] [743] [744] [745] [746] [747] [748] [749] [750] [751] [752] [753] [754] [755] [756] [757] [758] [759] [760] [761] [762] [763] [764] [765] [766] [767] [768] [769] [770] [771] [772] [773] [774] [775] [776] [777] [778] [779] [780] [781] [782] [783] [784] [785] [786] [787] [788] [789] [790] [791] [792] [793] [794] [795] [796] [797] [798] [799] [800] [801] [802] [803] [804] [805] [806] [807] [808] [809] [810] [811] [812] [813] [814] [815] [816] [817] [818] [819] [820] [821] [822] [823] [824] [825] [826] [827] [828] [829] [830] [831] [832] [833] [834] [835] [836] [837] [838] [839] [840] [841] [842] [843] [844] [845] [846] [847] [848] [849] [850] [851] [852] [853] [854] [855] [856] [857] [858] [859] [860] [861] [862] [863] [864] [865] [866] [867] [868] [869] [870] [871] [872] [873] [874] [875] [876] [877] [878] [879] [880] [881] [882] [883] [884] [885] [886] [887] [888] [889] [890] [891] [892] [893] [894] [895] [896] [897] [898] [899] [900] [901] [902] [903] [904] [905] [906] [907] [908] [909] [910] [911] [912] [913] [914] [915] [916] [917] [918] [919] [920] [921] [922] [923] [924] [925] [926] [927] [928] [929] [930] [931] [932] [933] [934] [935] [936] [937] [938] [939] [940] [941] [942] [943] [944] [945] [946] [947] [948] [949] [950] [951] [952] [953] [954] [955] [956] [957] [958] [959] [960] [961] [962] [963] [964] [965] [966] [967] [968] [969] [970] [971] [972] [973] [974] [975] [976] [977] [978] [979] [980] [981] [982] [983] [984] [985] [986] [987] [988] [989] [990] [991] [992] [993] [994] [995] [996] [997] [998] [999] [1000]
```

# Perspective: How Hard can Linux be, Really? [SharpSAT-td+Arjun, d4]



# 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  $\phi$  with  $x_\phi \leftrightarrow \phi$

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

### Z3

succeeds correctly on all models

### Plaisted-Greenbaum

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

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

### KConfigReader

often incorrect for #SAT calls

Compiling to **d-DNNFs**

Compiling to **BDDs**

Counting **Linux** is hard

find out more:



[github.com/ekuiter/tseitin-or-not-tseitin](https://github.com/ekuiter/tseitin-or-not-tseitin)

