



[Herrenkrug-Eisenbahnbrücke Magdeburg]

Advanced Topics in Feature-Model Analysis

Thesis Topics and Software Projects

April 4, 2024

Elias Kuiter¹

University of Magdeburg¹

1. Introduction

About Me

Short CV

- **2020**: M.Sc. Computer Science in Magdeburg
- **since 2021**: PhD student in Magdeburg supervised by Gunter Saake (Magdeburg) and Thomas Thüm (Ulm)

Research Interests

- Feature-Model Extraction, Transformation, and Analysis
- Satisfiability Solving, Formal Methods, Applied Category Theory

- **P**: Software Project
- **B**: Bachelor Thesis
- **M**: Master Thesis

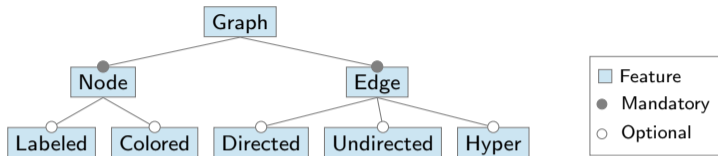


Contact me:
kuitner@ovgu.de

Modeling Features and their Dependencies

Feature Models

- tree models **features**
- cross-tree **constraints** model dependencies
- solver-based **analyses** for investigating the configuration space



$$\neg(\textit{Directed} \wedge \textit{Undirected})$$
$$\textit{Hyper} \rightarrow \textit{Undirected}$$
$$\textit{Directed} \not\leftrightarrow (\textit{Undirected} \wedge \textit{Hyper})$$

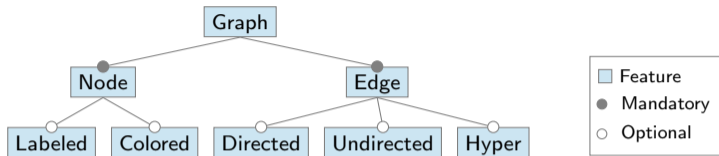
Modeling Features and their Dependencies

Feature Models

- tree models **features**
- cross-tree **constraints** model dependencies
- solver-based **analyses** for investigating the configuration space

The Linux Kernel

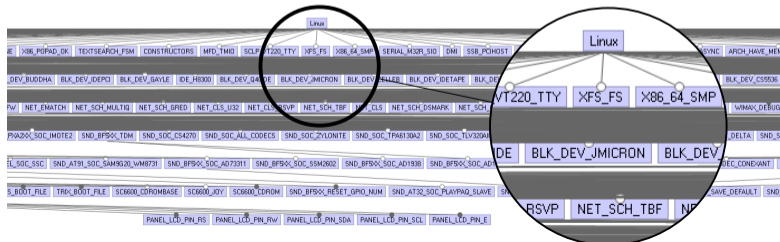
- > 13000 features [2018]
- > 10^{700} products [2007]
- 114 dead features [2013]
- 151 reverse dependency bugs [2019]



$\neg(\text{Directed} \wedge \text{Undirected})$

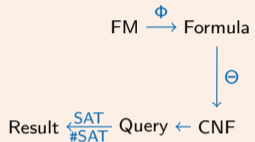
$\text{Hyper} \rightarrow \text{Undirected}$

$\text{Directed} \not\leftrightarrow (\text{Undirected} \wedge \text{Hyper})$



Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis

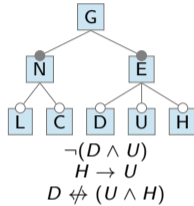


Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis



A Feature Model FM

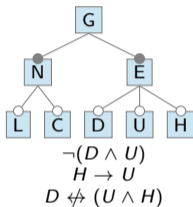


Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis



A Feature Model FM



$\xrightarrow{\Phi}$

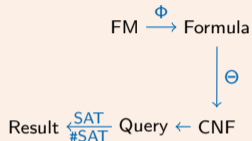
As a Formula $\Phi(FM)$

Logical formula $\Phi(FM)$:

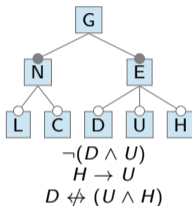
$$\begin{aligned} &G \\ &\wedge (N \leftrightarrow G) \wedge (E \leftrightarrow G) \\ &\wedge ((L \vee C) \rightarrow N) \\ &\wedge ((D \vee U \vee H) \rightarrow E) \\ &\wedge \neg(D \wedge U) \wedge (H \rightarrow U) \\ &\wedge (D \nleftrightarrow (U \wedge H)) \end{aligned}$$

Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis



A Feature Model FM



Φ

As a Formula $\Phi(FM)$

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

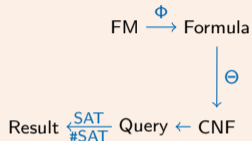
Θ

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

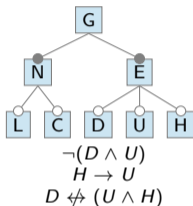
$\{\{G\}, \{\neg N, G\}, \{N, \neg G\},$
 $\{\neg E, G\}, \{E, \neg G\}, \{\neg L, N\},$
 $\{\neg C, N\}, \{\neg D, E\}, \{\neg U, E\},$
 $\{\neg H, E\}, \{\neg D, \neg U\}, \{\neg H, U\},$
 $\{\{D, U\}, \{D, H\}, \{\neg D, \neg U, \neg H\}\}$

Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis



A Feature Model FM



As a Formula $\Phi(FM)$

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

$\Phi \rightarrow$

$\downarrow \Theta$

Core Features

$\{G, N, E\}$

Core Feature $F?$

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

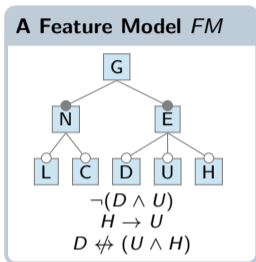
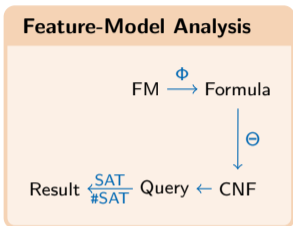
\leftarrow

\leftarrow

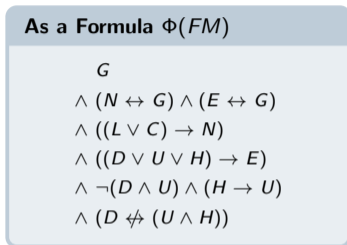
As a CNF $\Theta(\Phi(FM))$

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

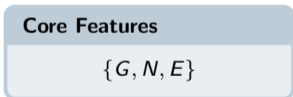
Analyzing Feature Models with SAT and #SAT Solvers



$\xrightarrow{\Phi}$



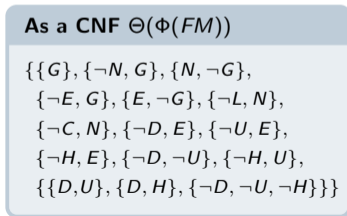
$\downarrow \Theta$



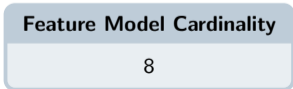
\leftarrow



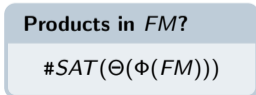
\leftarrow



\leftarrow



\leftarrow



2. Thesis Topics

Extracting Feature Hierarchies for KConfig-Based Feature Models (B/M)

Problem

- feature-model extractors for KConfig mostly ignore the **feature hierarchy**
- tooling for extracting hierarchies is now defunct, identification of **feature parents** in Kconfig is yet under-researched

```
1 namespace Root
2
3 features
4   Root
5     optional
6       UNWINDER_ORC
7       UNWINDER_FRAME_POINTER
8       UNWINDER_GUESS
9       X86_64
10      IO_DELAY_0X80
11      IO_DELAY_0XED
12      IO_DELAY_UDELAY
13      IO_DELAY_NONE
14      BRANCH_PROFILE_NONE
15      PROFILE_ANNOTATED_BRANCHES
```

Extracting Feature Hierarchies for KConfig-Based Feature Models (B/M)

Problem

- feature-model extractors for KConfig mostly ignore the **feature hierarchy**
- tooling for extracting hierarchies is now defunct, identification of **feature parents** in Kconfig is yet under-researched

```
1 namespace Root
2
3 features
4   Root
5     optional
6       UNWINDER_ORC
7       UNWINDER_FRAME_POINTER
8       UNWINDER_GUESS
9       X86_64
10      IO_DELAY_0X80
11      IO_DELAY_0XED
12      IO_DELAY_UDELAY
13      IO_DELAY_NONE
14      BRANCH_PROFILE_NONE
15      PROFILE_ANNOTATED_BRANCHES
```

Goal

- extract a feature hierarchy from KConfig specifications + evaluate accuracy
- and/or: reverse-engineer hierarchy from formula + compare with KConfig hierarchy

Requirements

- interested in research
- adjusting KConfig parser written in C
- adjust or implement a tool for reverse-engineering
- c.f. [Yaman 2023](#), [Yaman et al. 2024](#)

Feature-Model Analysis with SAT Solvers: A Journey Through Time (B/M) [assigned]

Problem

- feature models grow more complex over time
- automated reasoning tools (e.g., SAT solvers) get more efficient over time
- **but**: which development is faster? can SAT solvers actually keep up?



[Photo: Laurent Simon]

Feature-Model Analysis with SAT Solvers: A Journey Through Time (B/M) [assigned]

Problem

- feature models grow more complex over time
- automated reasoning tools (e.g., SAT solvers) get more efficient over time
- **but**: which development is faster? can SAT solvers actually keep up?

Goal

- collect best SAT solvers of the last 20 years
- collect feature models from the last 20 years
- run selected feature-model analyses with solver from year **X** on model of year **X**
- evaluate evolution of SAT solving performance (cf. Moore's law)
- see time travel challenge

Requirements

- interested in research
- methodology design, reading literature
- challenges: data availability and formats



[Photo: Laurent Simon]

Minimizing CNFs to Isolate Solver Bugs (B/M)

Problem

- CNFs of real-world feature models sometimes uncover **bugs** even in production-grade (#)SAT and SMT solvers
- e.g., in countAntom, sharpSAT/dSharp, Z3, clausy, FeatJAR
- during development and maintenance of such solvers, reducing problematic CNFs to a **minimum non-working example** can facilitate finding the causes of bugs, reporting them, and preventing future regressions
- however, this process is currently a manual task and time-consuming

Minimizing CNFs to Isolate Solver Bugs (B/M)

Problem

- CNFs of real-world feature models sometimes uncover **bugs** even in production-grade (#)SAT and SMT solvers
- e.g., in countAntom, sharpSAT/dSharp, Z3, clausy, FeatJAR
- during development and maintenance of such solvers, reducing problematic CNFs to a **minimum non-working example** can facilitate finding the causes of bugs, reporting them, and preventing future regressions
- however, this process is currently a manual task and time-consuming

Goal

- identify fault oracles (e.g., solver crashes), review reduction strategies (e.g., removing clauses one-by-one, bisection, backtracking to avoid a local minimum)
- implement a (semi-)automatic tool that repeatedly reduces clauses and literals in a faulty CNF until it is minimal
- evaluate performance and compare with global minimum (e.g., obtained manually)

Requirements

- interested in research, cf. [Böhm et al. 2024](#)
- algorithm design, reading literature
- challenge: generative effects, local minima

3. Software Projects

torte: Towards Fully Automated Feature-Model Experiments (P)

What is torte? 🍰

[github.com/ekuiter/torte]

- a declarative workbench for **reproducible** feature-model analysis experiments
- can extract, transform, and analyze feature models in a **fully automated** fashion
- draft, execute distribute, and adapt experiments (without clone-and-own)

A Simple Experiment: Counting BusyBox

```
experiment-subjects() {  
  add-busybox-kconfig-history --from 1_36.0 --to 1_36.1  
}  
experiment-stages() {  
  clone-systems  
  extract-kconfig-models  
  transform-models-into-dimacs  
  solve-model-count --timeout 10  
}
```

torte: Towards Fully Automated Feature-Model Experiments (P)

What is torte? 🍰

[github.com/ekuiter/torte]

- a declarative workbench for **reproducible** feature-model analysis experiments
- can extract, transform, and analyze feature models in a **fully automated** fashion
- draft, execute distribute, and adapt experiments (without clone-and-own)

A Simple Experiment: Counting BusyBox

```
experiment-subjects() {  
  add-busybox-kconfig-history --from 1_36.0 --to 1_36.1  
}  
experiment-stages() {  
  clone-systems  
  extract-kconfig-models  
  transform-models-into-dimacs  
  solve-model-count --timeout 10  
}
```

Goal

fix problems and implement new features from roadmap (issue #1)
⇒ enabling new use cases for torte

Requirements

- experience with Bash programming
- some experience with Docker
- willing to write clean code in Bash :-)

A Dashboard for Evolving Variability in Open-Source Systems (P)

Problem

- torte fully automates feature-model analysis
- can be used to analyze latest Linux kernel
- **but**: no user-friendly frontend exists yet

```
Öffnen  output.csv  Speichern
~\output\output_model-count
1 | dimacs-analyzer,dimacs-analyzer-time,dimacs-file,model-count
2 | emse-2023/countAnton,83629444,model_to_dimacs_featjar/kconfigreader/busybox/1.36.0,dimacs,
  754529145685077161557716966594558504127857914743044705157113622432926440841155805396071301653384095
3 | emse-2023/countAnton,60857801,model_to_dimacs_featjar/kmax/busybox/1.36.0,dimacs,
  143525790776422158276923001612623097740999787112481410091723362193802901118443264498528883505493014
4 | emse-2023/countAnton,84166596,model_to_dimacs_featureide/kconfigreader/busybox/1.36.0,dimacs,
  150905829137015432311543381318911700825571582948608941031422724486585288168231161079214260330676819
5 | emse-2023/countAnton,61793399,model_to_dimacs_featureide/kmax/busybox/1.36.0,dimacs,
  287051581552844316553846003225246195481999574224962820183446724387605802236886528997057767010986029
6 | emse-2023/countAnton,83645787,model_to_dimacs_kconfigreader/kconfigreader/busybox/1.36.0,dimacs,
  1422823310754414533398116964722424815821249402881936235232847886188591804063078592611253209415743
7 | emse-2023/countAnton,62364843,model_to_dimacs_kconfigreader/kmax/busybox/1.36.0,dimacs,
  945800176099629162731546931797173541865684159242083175759087829770536913024990653444218683263689621
8 | emse-2023/countAnton,77666641,smt_to_dimacs_z3/kconfigreader/busybox/1.36.0,dimacs,
  150905829137015432311543381318911700825571582948608941031422724486585288168231161079214260330676819
9 | emse-2023/countAnton,73514286,smt_to_dimacs_z3/kmax/busybox/1.36.0,dimacs,
  287051581552844316553846003225246195481999574224962820183446724387605802236886528997057767010986029
10 | emse-2023/d4,36636806,model_to_dimacs_featjar/kconfigreader/busybox/1.36.0,dimacs,
  754529145685077161557716966594558504127857914743044705157113622432926440841155805396071301653384095
11 | emse-2023/d4,30611432,model_to_dimacs_featjar/kmax/busybox/1.36.0,dimacs,
  143525790776422158276923001612623097740999787112481410091723362193802901118443264498528883505493014
12 | emse-2023/d4,37654188,model_to_dimacs_featureide/kconfigreader/busybox/1.36.0,dimacs,
  150905829137015432311543381318911700825571582948608941031422724486585288168231161079214260330676819
13 | emse-2023/d4,30280732,model_to_dimacs_featureide/kmax/busybox/1.36.0,dimacs,
  287051581552844316553846003225246195481999574224962820183446724387605802236886528997057767010986029
14 | emse-2023/d4,39223110,model_to_dimacs_kconfigreader/kconfigreader/busybox/1.36.0,dimacs,
  142282331075441453339811696472242481582124940288193623523328478861885918040630785926811253209415743
15 | emse-2023/d4,35471010,model_to_dimacs_kconfigreader/kmax/busybox/1.36.0,dimacs,
  143525790776422158276923001612623097740999787112481410091723362193802901118443264498528883505493014

CSV  Tabulatorbreite: 4  Z. 1, Sp. 1  INS
```

A Dashboard for Evolving Variability in Open-Source Systems (P)

Problem

- torte fully automates feature-model analysis
- can be used to analyze latest Linux kernel
- **but**: no user-friendly frontend exists yet

Goal

- develop a web frontend for torte
 - find appropriate visualizations
- ⇒ quick visualization of current state of variability

Requirements

- experience with frontend development (e.g., HTML/CSS, React/Vue/Dash, ...)
- no backend experience needed (assuming a static CSV file over AJAX)



```
Öffnen  output.csv  Speichern
~\output\tsche_model-count
1 |himacs-analyzer,dinacs-analyzer-time,dinacs-file,model-count
2 |ense-2023/countAnton,83629444,model_to_dinacs featjar/kconfigreader/busybox/1_36_0.dinacs,
754529145685077161557716966594558504127857914743044705157113622432926440841155805396071301653384095
3 |ense-2023/countAnton,60857801,model_to_dinacs featjar/kmax/busybox/1_36_0.dinacs,
143525790776422158276923001612623097740999787112481410091723362193802901118443264498528883505493014
4 |ense-2023/countAnton,84166596,model_to_dinacs featureide/kconfigreader/busybox/1_36_0.dinacs,
150905829137015432311543381318911700825571582948608941031422724486585288168231161079214260330676819
5 |ense-2023/countAnton,61793399,model_to_dinacs featureide/kmax/busybox/1_36_0.dinacs,
287051581552844316553846003225246195481999574224962820183446724387605802236886528997057767010986029
6 |ense-2023/countAnton,83645787,model_to_dinacs kconfigreader/kconfigreader/busybox/1_36_0.dinacs,
142282331075441453398116964722424815821249402881936235232847886188591804063078592611253209415743
7 |ense-2023/countAnton,62364843,model_to_dinacs kconfigreader/kmax/busybox/1_36_0.dinacs,
94580017609962916273154693179717354186568415924208317575908782977053691382499653444218683263689621
8 |ense-2023/countAnton,77666641,smt_to_dinacs z3/kconfigreader/busybox/1_36_0.dinacs,
150905829137015432311543381318911700825571582948608941031422724486585288168231161079214260330676819
9 |ense-2023/countAnton,73514286,smt_to_dinacs z3/kmax/busybox/1_36_0.dinacs,
287051581552844316553846003225246195481999574224962820183446724387605802236886528997057767010986029
10 |ense-2023/d4,36636806,model_to_dinacs featjar/kconfigreader/busybox/1_36_0.dinacs,
754529145685077161557716966594558504127857914743044705157113622432926440841155805396071301653384095
11 |ense-2023/d4,30611432,model_to_dinacs featjar/kmax/busybox/1_36_0.dinacs,
143525790776422158276923001612623097740999787112481410091723362193802901118443264498528883505493014
12 |ense-2023/d4,37654188,model_to_dinacs featureide/kconfigreader/busybox/1_36_0.dinacs,
150905829137015432311543381318911700825571582948608941031422724486585288168231161079214260330676819
13 |ense-2023/d4,30280732,model_to_dinacs featureide/kmax/busybox/1_36_0.dinacs,
287051581552844316553846003225246195481999574224962820183446724387605802236886528997057767010986029
14 |ense-2023/d4,39223110,model_to_dinacs kconfigreader/kconfigreader/busybox/1_36_0.dinacs,
142282331075441453398116964722424815821249402881936235232847886188591804063078592611253209415743
15 |ense-2023/d4,35471010,model_to_dinacs kconfigreader/kmax/busybox/1_36_0.dinacs,
CSV  Tabulatorbreite: 4  Z. 1, Sp. 1  INS
```

On-Demand Extraction of KConfig-Based Feature Models (P)

Problem

- torte fully automates feature-model analysis
- can be used to analyze latest Linux kernel
- **but**: replication packages are huge and not up-to-date, on-demand extraction is missing

model_to_xml_featureide	179,8 GB
kconfig	83,8 GB
kconfigreader	47,5 GB
kmax	36,3 GB
model_to_smt_z3	29,3 GB
dimacs	27,4 GB
backbone-dimacs	20,7 GB
model_to_uvl_featureide	10,8 GB

On-Demand Extraction of KConfig-Based Feature Models (P)

Problem

- torte fully automates feature-model analysis
- can be used to analyze latest Linux kernel
- **but**: replication packages are huge and not up-to-date, on-demand extraction is missing

model_to_xml_featureide	179,8 GB
kconfig	83,8 GB
kconfigreader	47,5 GB
kmax	36,3 GB
model_to_smt_z3	29,3 GB
dimacs	27,4 GB
backbone-dimacs	20,7 GB
model_to_uvl_featureide	10,8 GB

Goal

- develop a server backend for torte
 - design an appropriate job architecture
 - strengthen against RCE
- ⇒ quick “self-help” for common extraction needs



Requirements


- experience with backend development (e.g., Docker, job processing, PHP/Node.js, ...)
- willing to write a simple HTML frontend

Interested?



Contact me: kuitert@ovgu.de

/ekuitert/

/ekuitert/