Advanced Topics in Feature-Model Analysis
Thesis Topics and Software Projects

April 4, 2024
Elias Kuiter
University of Magdeburg
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

**Contact me:**

kuiter@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

![Diagram](https://via.placeholder.com/150)

\[ \neg (\text{Directed} \land \text{Undirected}) \]

\[ \text{Hyper} \rightarrow \text{Undirected} \]

\[ \text{Directed} \Leftrightarrow (\text{Undirected} \land \text{Hyper}) \]
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
Analyzing Feature Models with SAT and \#SAT Solvers

Feature-Model Analysis

A Feature Model $FM$

$$\begin{align*}
\text{G} & \quad \text{N} & \quad \text{E} \\
\text{L} & \quad \text{C} & \quad \text{D} & \quad \text{U} & \quad \text{H}
\end{align*}$$

$$\neg(D \land U)$$
$$H \rightarrow U$$
$$D \leftrightarrow (U \land H)$$
Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis

- **FM** ➔ **Formula**
  - **Result**
    - **SAT**
    - **Query** ➔ **CNF**

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

- \( G \)
- \( (N \leftrightarrow G) \land (E \leftrightarrow G) \)
- \( ((L \lor C) \rightarrow N) \land ((D \lor U \lor H) \rightarrow E) \)
- \( (D \land U) \land (H \rightarrow U) \)
- \( D \leftrightarrow (U \land H) \)
Analyzing Feature Models with SAT and \#SAT Solvers

Feature-Model Analysis

A Feature Model $FM$

As a Formula $\Phi(FM)$

As a CNF $\Theta(\Phi(FM))$
## Analyzing Feature Models with SAT and \#SAT Solvers

### Feature-Model Analysis

<table>
<thead>
<tr>
<th>Feature-Model Analysis</th>
<th>A Feature Model (FM)</th>
<th>As a Formula (\Phi(FM))</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM (\Phi) (\to) Formula</td>
<td>(\Theta)</td>
<td>(\Phi)</td>
</tr>
<tr>
<td>Result #SAT (\Theta) Query (\to) CNF</td>
<td>(\neg(D \land U)) (H \rightarrow U) (D \leftrightarrow (U \land H))</td>
<td>(G) (\land (N \leftrightarrow G) \land (E \leftrightarrow G)) (\land ((L \lor C) \rightarrow N)) (\land ((D \lor U \lor H) \rightarrow E)) (\land \neg(D \land U) \land (H \rightarrow U)) (\land (D \leftrightarrow (U \land H)))</td>
</tr>
</tbody>
</table>

### Core Features

- \(\{G, N, E\}\)

### Core Feature \(F\)?

- \(\text{SAT}(\Theta(\Phi(FM)) \land \neg F)\)

### 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**

- Formula: $\Phi(FM)$
- Result: $\Theta(\Phi(FM))

**A Feature Model $FM$**

- Core Features: \{G, N, E\}
- Core Feature $F$?
- Feature Model Cardinality: 8
- Products in $FM$?
- \#SAT($\Theta(\Phi(FM))$)

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

- $G$ \land (N \leftrightarrow G) \land (E \leftrightarrow G)
- \land ((L \lor C) \rightarrow N)
- \land ((D \lor U \lor H) \rightarrow E)
- \land \neg (D \land U) \land (H \rightarrow U)
- \land (D \leftrightarrow (U \land H))

**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\}, \{\neg D, \neg U, \neg H\}\}
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

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

```
namespace Root
features
    Root
        optional
            UNWINDER.ORC
            UNWINDER.FRAME.POINTER
            UNWINDER.GUESS
            X86_64
            IO_DELAY.0X80
            IO_DELAY.0XED
            IO_DELAY.UDELAY
            IO_DELAY.NONE
            BRANCH_PROFILE.NONE
            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

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)

**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
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
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
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

```bash
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 :-)

Elias Kuiter
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

```bash
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 :-)

Elias Kuiter
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)
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)
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

<table>
<thead>
<tr>
<th>Directory</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>model_to_xml_featureide</td>
<td>179,8 GB</td>
</tr>
<tr>
<td>kconfig</td>
<td>83,8 GB</td>
</tr>
<tr>
<td>kconfigreader</td>
<td>47,5 GB</td>
</tr>
<tr>
<td>kmax</td>
<td>36,3 GB</td>
</tr>
<tr>
<td>model_to_smt_z3</td>
<td>29,3 GB</td>
</tr>
<tr>
<td>dimacs</td>
<td>27,4 GB</td>
</tr>
<tr>
<td>backbone-dimacs</td>
<td>20,7 GB</td>
</tr>
<tr>
<td>model_to_uvl_featureide</td>
<td>10,8 GB</td>
</tr>
</tbody>
</table>
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

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: kuiter@ovgu.de

🤖/ekuiter/🤖
👩‍💻/ekuiter/👩‍💻