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

July 4, 2024
Elias Kuiter
University of Magdeburg
1. Introduction
About Me

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

My Research Interests

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

- **P**: Software Project
- **B**: Bachelor Thesis
- **M**: Master Thesis
- thesis language can be English (preferred) or German
- ignore [assigned] topics

You . . .

- are interested in working on current research tools and topics
- successfully completed our product-line lecture

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

\[
\neg (Directed \land Undirected)
\]

Hyper \implies Undirected

Directed $\Leftrightarrow$ (Undirected $\land$ Hyper)
A BDD for Linux?

Thomas Thüm

University of Ulm, Germany

thomas.thuem@uni-ulm.de

The main reason for the scalability challenge is that the variable ordering heavily influences the size of the BDD and they tend to explode for most variable orderings.

What is the holy grail of the product-line community? A binary decision diagram for Linux.

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

- ≈ 20,000 features
- > 10^{1,700} products
- 114 dead features [2013]
- 151 reverse dependency bugs [2019]
Analyzing Feature Models with SAT and \#SAT Solvers

Feature-Model Analysis

\[ \Phi \] \rightarrow \text{Formula} \rightarrow \Theta \rightarrow \text{Result} \rightarrow \text{Query} \rightarrow \text{CNF} \]

FM \rightarrow \text{Formula} \rightarrow \text{Result} \rightarrow \text{Query} \rightarrow \text{CNF}
Analyzing Feature Models with SAT and \#SAT Solvers

Feature-Model Analysis

\[ \Phi \rightarrow \text{Formula} \]

Result

\[ \text{query} \leftarrow \text{CNF} \]

A Feature Model \( FM \)

\[ \neg (D \land U) \]
\[ H \rightarrow U \]
\[ D \Leftrightarrow (U \land H) \]
Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis

A Feature Model $FM$

As a Formula $\Phi(FM)$

<table>
<thead>
<tr>
<th>Feature-Model Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FM</strong> $\rightarrow$ Formula</td>
</tr>
<tr>
<td>$\Theta$</td>
</tr>
<tr>
<td>Query $\leftarrow$ CNF</td>
</tr>
<tr>
<td>Result $\text{SAT}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Core Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Core Features} = {G, N, E}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature Model Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$8$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Products in FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$#\text{SAT}(\Theta(\Phi(FM)))$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>As a CNF $\Theta(\Phi(FM))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${{G}, {\neg N, G}, {N, \neg G}, {\neg E, G}, {E, \neg G}, {\neg L, N}, {\neg C, N}, {\neg D, E}, {\neg U, E}, {\neg H, E}, {(D \land U), \neg (D \land U) \land (H \rightarrow U) \land (D \not\leftrightarrow (U \land H))}}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formula $\Phi(FM)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$ $\land (N \leftrightarrow G) \land (E \leftrightarrow G)$ $\land ((L \lor C) \rightarrow N)$ $\land ((D \lor U \lor H) \rightarrow E)$ $\land \neg (D \land U) \land (H \rightarrow U)$ $\land (D \not\leftrightarrow (U \land H))$</td>
</tr>
</tbody>
</table>
Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis

A Feature Model $FM$

As a Formula $\Phi(FM)$

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

Result

Elias Kuiter
Advanced Topics in Feature-Model Analysis – Thesis Topics and Software Projects – 1. Introduction
Analyzing Feature Models with SAT and \#SAT Solvers

Feature-Model Analysis

\[
\Phi \quad \text{FM} \rightarrow \text{Formula} \\
\Theta \quad \text{Result SAT \#SAT} \\
\text{Query} \rightarrow \text{CNF}
\]

A Feature Model  \( FM \)

\[
\begin{align*}
G & \\
N \leftrightarrow G & \\
E \leftrightarrow G & \\
\neg (D \land U) & \\
H \rightarrow U & \\
D \leftrightarrow (U \land H)
\end{align*}
\]

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

\[
\begin{align*}
G & \\
\land (N \leftrightarrow G) & \land (E \leftrightarrow G) & \\
\land ((L \lor C) \rightarrow N) & \\
\land ((D \lor U \lor H) \rightarrow E) & \\
\land \neg (D \land U) & \land (H \rightarrow U) & \\
\land (D \leftrightarrow (U \land H))
\end{align*}
\]

Core Features

\[ \{G, N, E\} \]

Core Feature  \( F \) ?

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

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

\[
\begin{align*}
\{\{G\}, \{-N, G\}, \{N, \neg G\}, \\
\{-E, G\}, \{E, \neg G\}, \{-L, N\}, \\
\{-C, N\}, \{-D, E\}, \{-U, E\}, \\
\{-H, E\}, \{-D, \neg U\}, \{-H, U\}, \\
\{\{D, U\}, \{D, H\}, \{-D, \neg U, \neg H\}\}
\end{align*}
\]
Analyzing Feature Models with SAT and #SAT Solvers

Feature-Model Analysis

A Feature Model $FM$

As a Formula $\Phi(FM)$

Core Features

{\(G, N, E\)}

Core Feature $F$?

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

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\}\}\}
2. Thesis Topics
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

```c
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

```
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
```
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?
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

- methodology design, reading literature
- challenges: data availability and formats
Evaluating an Extractor for KConfig-Based Feature Models (B)

**Problem**
- A new feature-model extractor, ConfigFix, has recently been published.
- However, it has yet not been compared to existing extractors yet.

**Goal**
- Implement extraction of feature models with ConfigFix in torte.
- Evaluate efficiency and accuracy on a large corpus of feature models.

**Requirements**
- Good in bash programming.
- Work with ConfigFix, which is implemented in C.
Evaluating an Extractor for KConfig-Based Feature Models (B)

Problem
- A new feature-model extractor, ConfigFix, has recently been published.
- However, it has yet not been compared to existing extractors yet.

Goal
- Implement extraction of feature models with ConfigFix in torte.
- Evaluate efficiency and accuracy on a large corpus on feature models.

Requirements
- Good in bash programming.
- Work with ConfigFix, which is implemented in C.
Minimizing CNFs to Isolate Solver Bugs (B)

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

- cf. Böhm et al. 2024
- algorithm design, reading literature
- challenge: generative effects, local minima
Minimizing CNFs to Isolate Solver Bugs (B)

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), re-view 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

• 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? ⚠️

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

What is 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

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

```
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

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
Early Ideas

- evaluate the influence of CNF scrambling (i.e., order of clauses in a CNF) on solver performance
- evaluate the influence of the KConfig extractor binary on the extracted feature-model formula
- laze vs. KConfig: compare a new configuration language with KConfig, maybe implement transformations
- scale Linux feature models up and down (create architecture-unifying and driver-specific models)
- extract solution-space models from KConfig, which encode program variants instead of configurations
- integrate feature-model benchmark into torte to extend it towards non-KConfig models
- compare measures of configurability (e.g., configurations, program variants, distinct binaries)
- . . . (your own topic ideas related to feature-model analysis)

on interest, we can develop these into concrete topics (but not all are suitable for writing a thesis)
Interested?

Contact me: kuiter@ovgu.de

👩‍💻/ekuiter/👩
👩‍💻/ekuiter/🎉