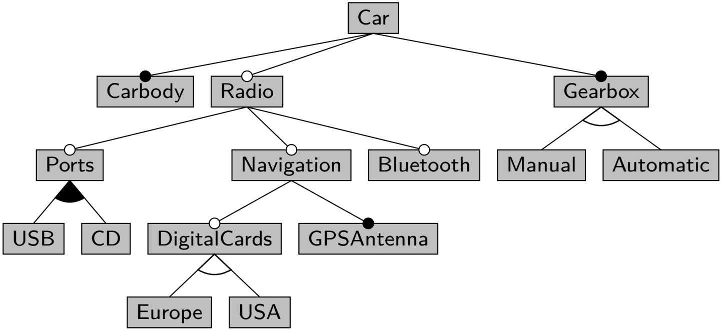
Binary Decision Diagrams in Product-Line Analysis

FOSD Online Meeting 2021 | **Tobias Heß**, Chico Sundermann, Thomas Thüm | 15.04.2021

SP Software Engineering Programming Languages

ulm university universität ulm

Feature Models



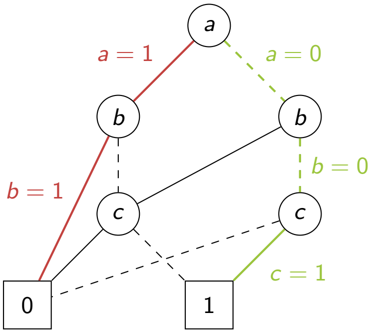
Feature Diagram

$$USA \Rightarrow \neg Manual$$

Constraints

Binary Decision Diagrams (BDD)

Let $f(a, b, c) = (\neg a \vee \neg b) \wedge (\neg a \vee \neg c) \wedge (\neg b \vee \neg c) \vee (a \vee b \vee c)$.



Why should we use BDDs?

$\underbrace{\text{SAT}}_{\text{const.}}$, $\underbrace{\text{Verification,}}_{\text{linear wrt. \#variables}}$ $\underbrace{\text{\#SAT, Commonality}}_{\text{linear wrt. \#nodes}}$

$\boxed{\underbrace{\text{Negation,}}_{\text{const.}}\ \underbrace{\text{Composition, Disjunction}}_{\text{polynomial time \& space wrt. size of BDDs}}}$

Not feasible with CNF or d-DNNF.

Feature Model Differences

Mathieu Acher[1], Patrick Heymans[1,2], Philippe Collet[3], Clément Quinton[2], Philippe Lahire[3], and Philippe Merle[2]

Reasoning about Edits to Feature Models

Thomas Thüm, School of Computer Science, University of Magdeburg, tthuem@st.ovgu.de
Don Batory, Dept. of Computer Science, University of Texas at Austin, batory@cs.utexas.edu
Christian Kästner, School of Computer Science, University of Magdeburg, ckaestne@ovgu.de

$\underbrace{\text{Existential Quantification, QBF}}_{\text{polynomial wrt. size of BDDs}}$

Slicing Feature Models

Mathieu Acher, Philippe Collet, Philippe Lahire, I3S – CNRS UMR 6070, Université Nice Sophia Antipolis, France, {acher,collet,lahire}@i3s.unice.fr
Robert B. France, Computer Science Department, Colorado State University, USA, france@cs.colostate.edu

Why aren't we using BDDs?

Slicing Feature Models

Mathieu Acher, Philippe Collet, Philippe Lahire
I3S – CNRS UMR 6070
Université Nice Sophia Antipolis, France
{acher,collet,lahire}@i3s.unice.fr

Robert B. France
Computer Science Department
Colorado State University, USA
france@cs.colostate.edu

"[..] the size of the BDD and that are known to scale for up to **2,000** features. [..] very large FMs (i.e., with more than 5,000 features) [..] For this order of complexity, **BDDs do not scale**."

Feature Model Differences

Mathieu Acher[1], Patrick Heymans[1,2], Philippe Collet[3], Clément Quinton[2], Philippe Lahire[3], and Philippe Merle[2]

"[BDDs] scale up to **2,000** features."

Propagating Configuration Decisions with Modal Implication Graphs

Sebastian Krieter
ersity of Magdeburg, Germany
University of Applied Sciences

Thomas Thüm
TU Braunschweig, Germany
t.thuem@tu-braunschweig.de

Sandro Schulze
Reimar Schröter
Gunter Saake

"A problem with BDDs is that they typically do not scale for feature models larger than **1,000** features."

"BDDs scale for feature models with $\leq$ 2,000 features"

Efficient Compilation Techniques for Large Scale Feature Models

Marcilio Mendonca[1], Andrzej Wasowski[2], Krzysztof Czarnecki[1] and Donald Cowan[1]
University of Waterloo[1], IT University of Copenhagen[2]
{marcilio,dcowan}@csg.uwaterloo.ca, wasowski@itu.dk, and kczarnec@swen.uwaterloo.ca

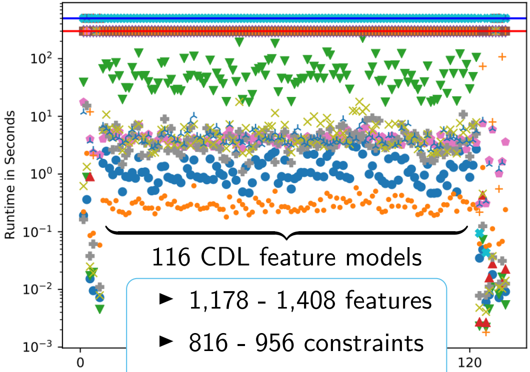
2009

- **Def.:** $ECR(FM) = \dfrac{\text{\# unique features in constraints}}{\text{\# features}}$
- Based on experiments with artificial feature models
  - ECR settings: 10%, 20%, 30%
  - Parameters founded on real-world FMs of the time
  - Largest: *e-Shop*, 326 features, 21 constraints, ECR: 10.4%

And today?

Evaluating #SAT Solvers on Industrial Feature Models

Chico Sundermann, Technische Universität Braunschweig
Thomas Thüm, University of Ulm
Ina Schaefer, Technische Universität Braunschweig



116 CDL feature models

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

0% success rate

And today?

Feature models have “evolved”

- 1-2 orders of magnitude larger
- Often: #features ≈ #constraints
- ECR: 5% - 95%

BDD tooling has not “evolved”

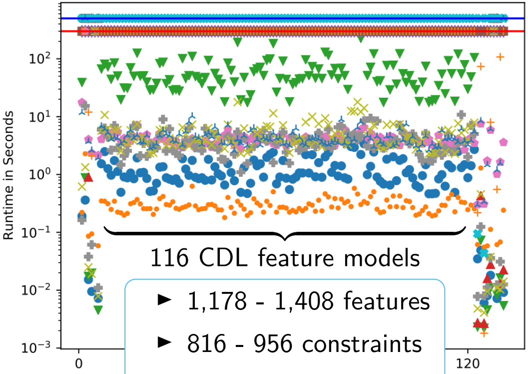
- BuDDy (1996) and CUDD (1995)
- Expert knowledge required
- Not included in current frameworks

Evaluating #SAT Solvers on Industrial Feature Models

Chico Sundermann
Technische Universität Braunschweig

Thomas Thüm
University of Ulm

Ina Schaefer
Technische Universität Braunschweig



116 CDL feature models

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

0% success rate

What can we do?

Variable Ordering

- *Future work*™ in many papers
- linear size $\leftrightarrow$ exponential size

Graph-Based Algorithms for Boolean Function Manipulation

RANDAL E. BRYANT, MEMBER, IEEE

Exploit feature diagram structure

- Feature diagram induces variable ordering $\rightarrow$ **BDD of linear size** (w/o constraints)

Using Extended Logical Primitives for Efficient BDD Building

David Fernandez-Amoros *, Sergio Bra, Ernesto Aranda-Escolástico and Ruben Heradio

Efficient Compilation Techniques for Large Scale Feature Models

Marcilio Mendonca[1], Andrzej Wasowski[2], Krzysztof Czarnecki[1] and Donald Cowan[1]
University of Waterloo[1], IT University of Copenhagen[2]
{marcilio,dcowan}@csg.uwaterloo.ca, wasowski@itu.dk, and kczarnec@swen.uwaterloo.ca

- **Consequence:** Don't use CNF (e.g., DIMACS) as input for BDD construction.

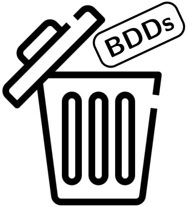
What are we doing?

- 1^{st} stage: ~~Denial~~ **Community, we have a problem!**

Applications of #SAT Solvers on Feature Models

Chico Sundermann University of Ulm, Germany	Michael Nieke TU Braunschweig, Germany	Paul Maximilian Bittner University of Ulm, Germany
Tobias Heß University of Ulm, Germany	Thomas Thüm University of Ulm, Germany	Ina Schaefer TU Braunschweig, Germany

$\Rightarrow$



- 2^{nd} stage: ~~Anger~~ **Address future work!** (e.g., finding good variable orderings)
- 3^{nd} stage: ~~Bargaining~~ **What can we do different?** (e.g., BDD variants like ZDD)

With proper variable ordering, BDDs scale to CDL feature models!

`ddueruem`

A wrapper for BuDDy and CUDD written in Python.

Features

- command line interface

```
./ddueruem.py input.uvl    --preorder force    --dynorder sift-converge
```

`./ddueruem.py input.uvl`: parse input, build BDD, analyse

`--preorder force`: opt. specify variable ordering

`--dynorder sift-converge`: opt. specify variable ordering during build

- downloads & builds BuDDy and CUDD for you
- parses DIMACS or UVL (Unified Variability Language)

Feature requests welcome!

~~Take~~ Stay Home Messages

The Good

BDDs solve many problems in product-line analysis.

The Bad

BDD construction is hard, requires care (variable ordering!), and tooling is insufficient.

The Ugly

Some feature models may force BDDs of exponential size.

Backup Slides

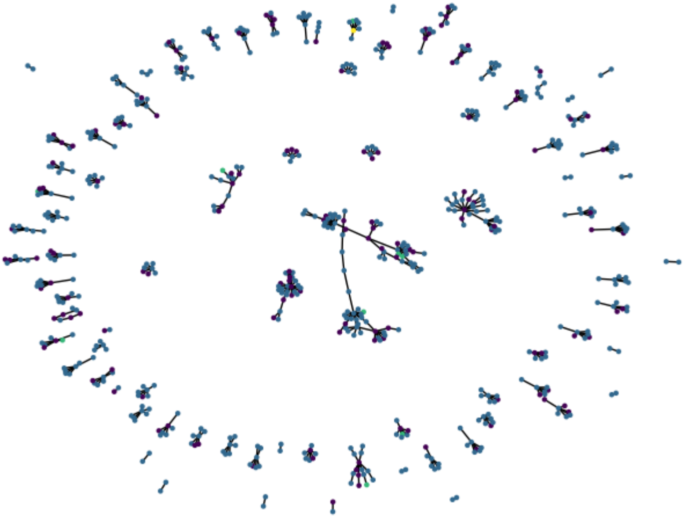
Automotive02_v1 vs. *Automotive02_v4*

	#Features	#CTCs	#Nodes	Max #Nodes	Average Time (s)
Automotive02_v1	14,010	666	70,003	111,000	**20.3**
Automotive02_v2	17,742	914	90,254	90,254	1,275
Automotive02_v3	18,343	1,300	65,022	192,000	1,654
Automotive02_v4	18,616	1,369	76,156	185,000	**1,680**

What makes *Automotive02_v4* **8,000%** harder than *Automotive02_v1*?

- **28%** more features
- **105%** more cross-tree constraints
- features to cross-tree constraints ratio: **4%** vs. **7%**
- ECR: **5.7%** vs. **8%**

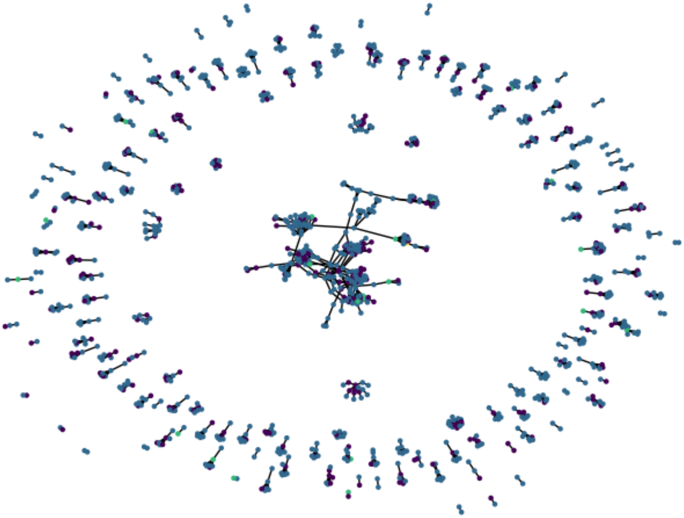
Graph Metrics



Automotive02_v1

- **88** components
- Minimum Size: **2**
- Average Size: **7.2**
- Maximum Size: **43**

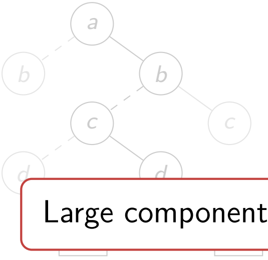
Graph Metrics cont.

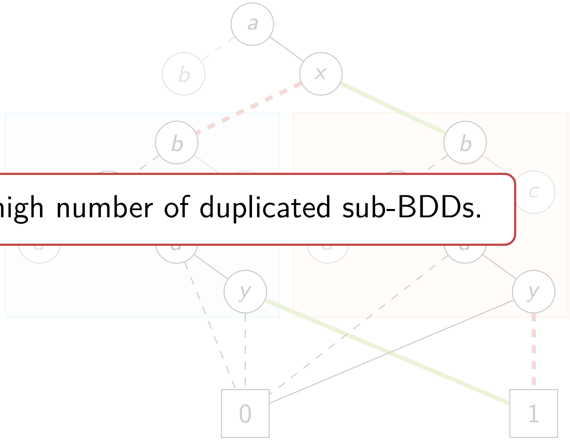


Automotive02_v4

- **167** components
- Minimum Size: **2**
- Average Size: **7.2**
- Maximum Size: **198**

Why are large components a problem?





Large components lead to a high number of duplicated sub-BDDs.

Variable Order: $\pi = (a, x, b, c, y, d)$

Add cross-tree constraint:

$$C = (x \vee y) \wedge (\neg x \vee \neg y)$$

Automotive01: Cross-Tree Constraint Structure