# A BDD for Linux?

## The Knowledge Compilation Challenge for Variability

Thomas Thüm
University of Ulm, Germany
thomas.thuem@uni-ulm.de

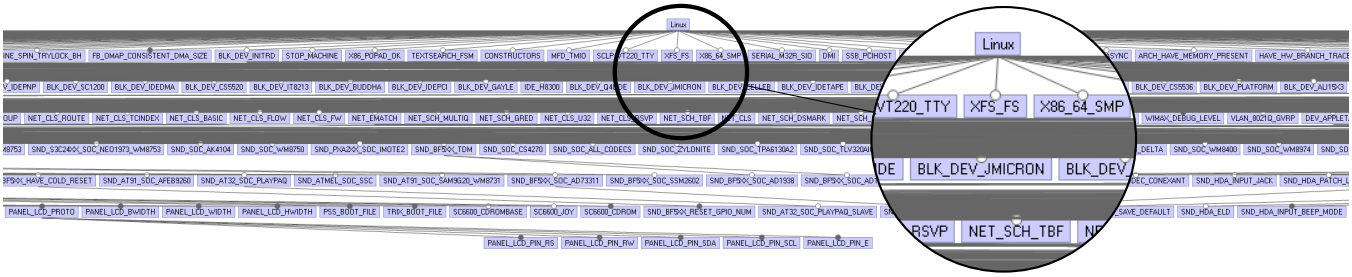


**Figure 1: Excerpt of a feature model for Linux 2.6.33.3 with 6,467 features and 3,545 cross-tree constraints in FeatureIDE.**

## ABSTRACT

What is the number of valid configurations for Linux? How to generate uniform random samples for Linux? Can we create a binary decision diagram for Linux? It seems that the product-line community tries hard to answer such questions for Linux and other configurable systems. However, attempts are often not published due to the publication bias (i.e., unsuccessful attempts are not published). As a consequence, researchers keep trying by potentially spending redundant effort. The goal of this challenge is to guide research on these computationally complex problems and to foster the exchange between researchers and practitioners.

## CCS CONCEPTS

• **Software and its engineering** → **Formal software verification**; **Software testing and debugging**; **Software verification**; **Automated static analysis**; **Consistency**; **Software configuration management and version control systems**; **Preprocessors**; • **Theory of computation** → **Program verification**; **Program analysis**; **Logic and verification**

.

## KEYWORDS

software product line, configurable system, software configuration, product configuration, feature models, decision models, artificial intelligence, satisfiability solving, knownledge compilation, binary decision diagrams


*SPLC '20, October 19–23, 2020, Montreal, QC, Canada*

ACM ISBN 978-1-4503-7569-6/20/10…$15.00
https://doi.org/10.1145/3382025.3414943

**ACM Reference Format:**
Thomas Thüm. 2020. A BDD for Linux?: The Knowledge Compilation Challenge for Variability. In *24th ACM International Systems and Software Product Line Conference (SPLC '20), October 19–23, 2020, Montreal, QC, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3382025.3414943

## 1 MOTIVATION

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

Linux is an operating system with thousands of configuration options (cf. Figure 1). These options cannot be arbitrarily combined, as every option typically comes with constraints with respect to several other options. Constraints are specified in KConfig [21], but can be translated into feature models or propositional logic [11, 32, 34, 37, 40, 49, 57, 65, 68, 71, 77, 81–83]. Whenever we analyze the Linux kernel for errors, ignoring those constraints would lead to false positives. That is, tools would report errors for invalid configurations, which cannot be used to compile a kernel. Hence, these constraints are crucial for any kind of analysis of Linux.

A *binary decision diagram* (short BDD) is data structure representing a propositional formula. While there are multiple representations of propositional formulas, BDDs can have the advantage of reducing NP-complete problems into more tractable problems (aka. *knowledge compilation*) [14, 31]. For instance, checking whether a formula represented as a BDD is satisfiable is an operation with constant effort. While operations on BDDs might scale well, the downside of BDDs is that their construction can be intractable. 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.

*Why do we argue that a BDD for Linux is the holy grail of the product-line community?*

First, it seems to be a challenging task. To the best of our knowledge, no one has been able to create a BDD for Linux so far. In recent work, we tried to create a BDD for hundreds of large feature