



A Binary Decision Diagram for Linux?

The Knowledge Compilation Challenge for Variability | Thomas Thüm | October 22nd, 2020



Product Lines are Powered by SAT

Product Lines are Powered by SAT



Product Lines are Powered by SAT

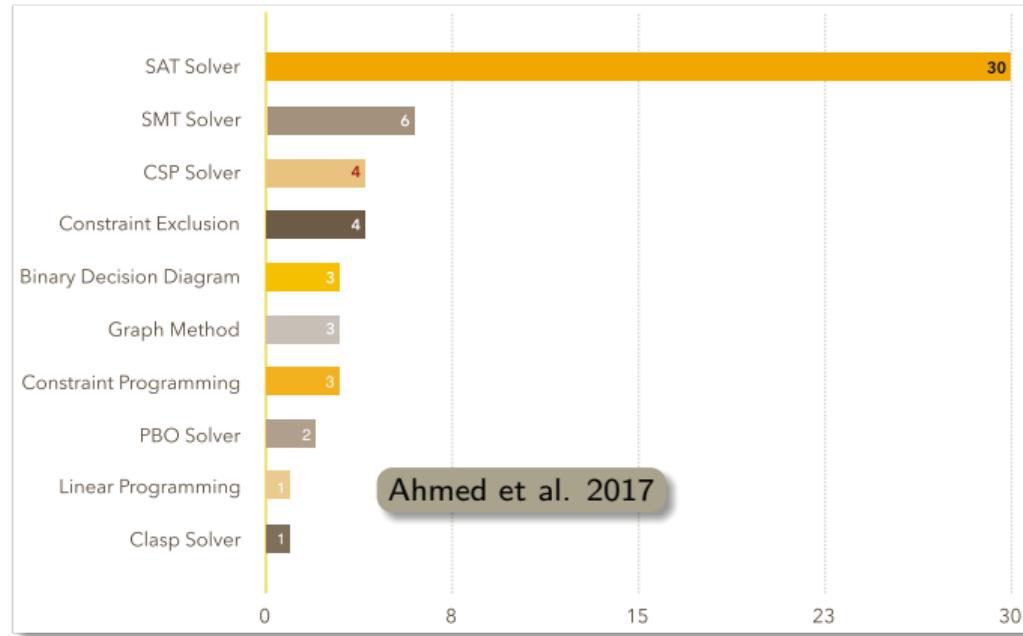


Thomas Thüm

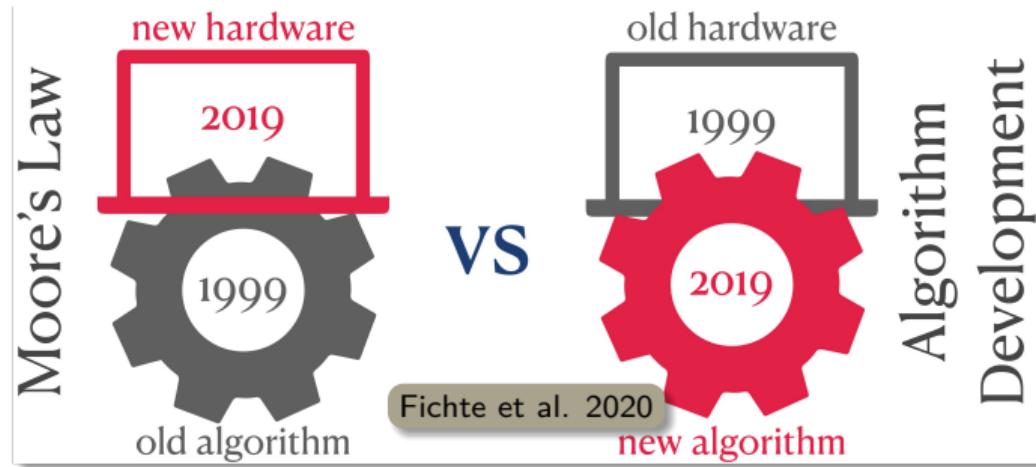
Product Lines are Powered by SAT



Used Solvers (for Interaction Testing)



Time-Leap Challenge for SAT-Solving

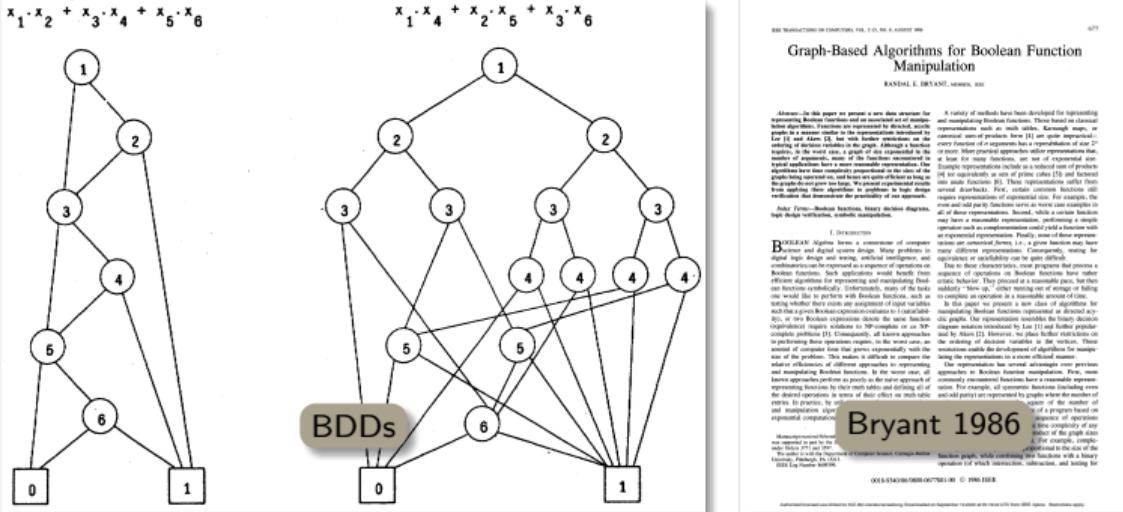


What is Our Holy Grail?

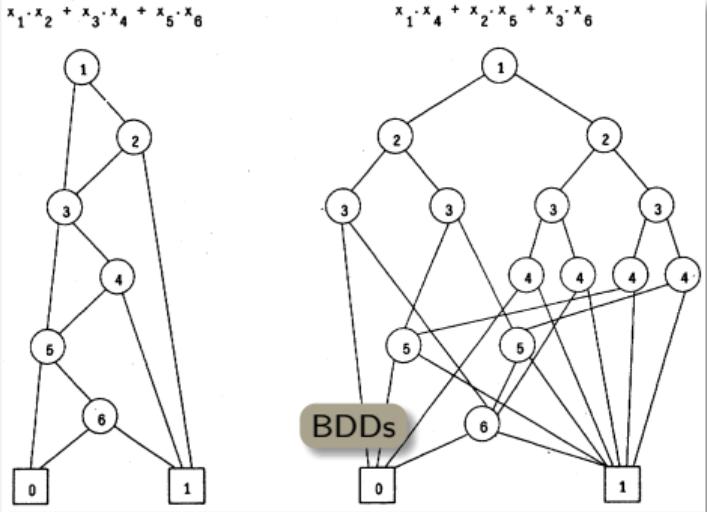
A BDD for Linux.



Why Binary Decision Diagrams?



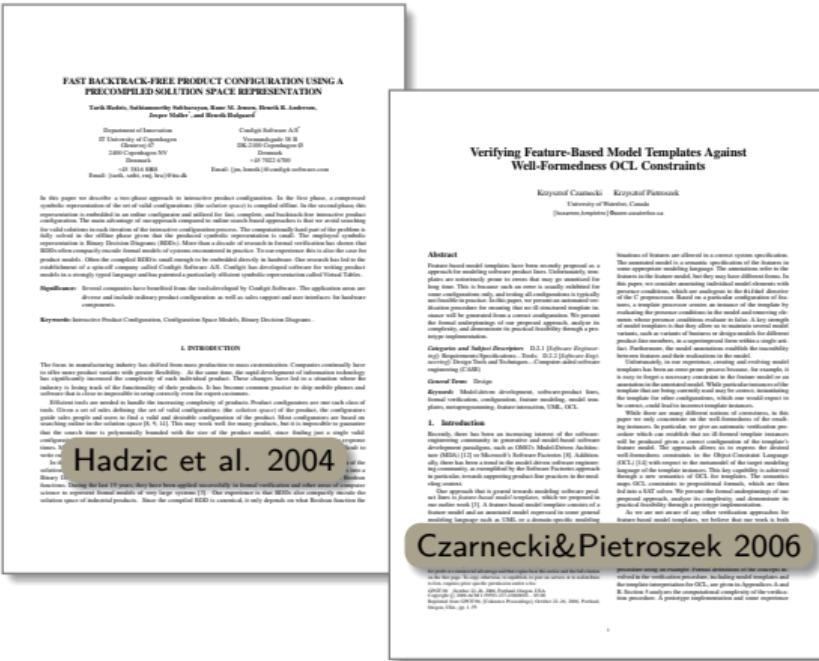
Many Queries on BDDs are fast.



Use of BDDs for Product Lines



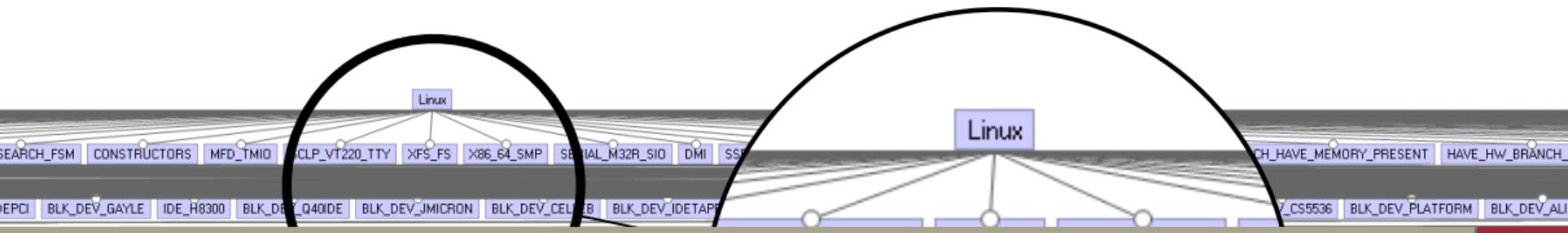
Use of BDDs for Product Lines



Use of BDDs for Product Lines



Why Linux?



Large Product-Lines Beyond Linux

Large Product-Lines Beyond Linux

Lazy Product Discovery in Huge Configuration Spaces

Michael Lienhardt
ONERA - The French Aerospace Lab
France
michael.lienhardt@onera.fr

Fernando Damiani
Università di Torino
Italy
fernando.damiani@polito.it

Einar Broch Johnsen
University of Oslo
Norway
eiracj@ifi.uio.no

Jacopo Mauro
University of Southern Denmark
Denmark
jmauro@cs.sdu.dk

ABSTRACT

Highly-configurable software systems can have thousands of inter-dependent configuration options. In such spaces, discovering a valid product configuration among all possible configurations is time consuming. This paper proposes a method to efficiently discover valid configurations for some selected options for complex and error-prone systems. The method is based on a feature model and a feature model-based search space. It iteratively explores the search space by selecting the configuration options of each subsystem.

The proposed method is evaluated on two benchmarks in large highly-configurable feature models with interdependent features. We consider the number of configurations to generate and complete the search space explore an area of the user-defined configuration space. The results show that lazy product discovery has equalized performance between our method and a state-of-the-art approach. Moreover, our method requires all diagrams to be compiled in another feature model. Furthermore, the method succeeds when more efficient, because-based engines fail to find a valid configuration.

CCS CONCEPTS

Software Product Lines; Configurable Software; Variability Modeling; Feature Model Analysis; Composition; Linear Dilemmas; Feature Interactions; Configuration; modeling and variability; Software libraries and repositories; Software creation and management;

KEYWORDS

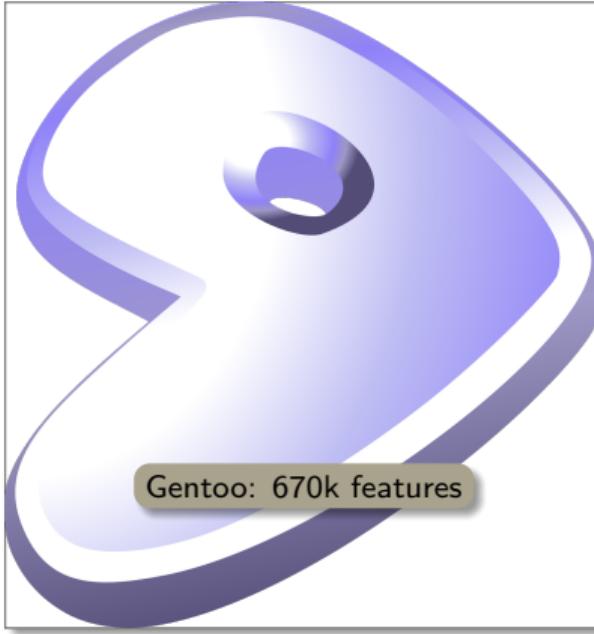
Software Product Lines; Configurable Software; Variability Modeling; Feature Model Analysis; Composition; Linear Dilemmas

Michael Lienhardt, Fernando Damiani, Einar Broch Johnsen, and Jacopo Mauro
Lazy Product Discovery in Huge Configuration Spaces
In Proceedings of the International Conference on Software Engineering (ICSE '20), May 24–28, 2020, Seoul, Republic of Korea. ACM, New York, NY, USA, 11 pages.
<https://doi.org/10.1145/3395311.3395600>

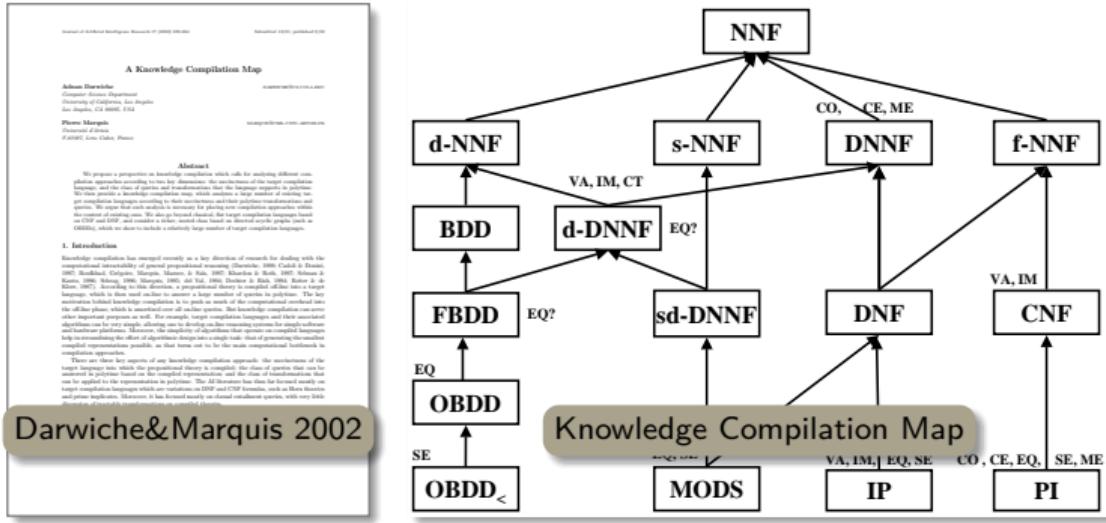
Abstract— Highly-configurable software systems can have thousands of inter-dependent configuration options. In such spaces, discovering a valid product configuration among all possible configurations is time consuming. This paper proposes a method to efficiently discover valid configurations for some selected options for complex and error-prone systems. The method is based on a feature model and a feature model-based search space. It iteratively explores the search space by selecting the configuration options of each subsystem. The proposed method is evaluated on two benchmarks in large highly-configurable feature models with interdependent features. We consider the number of configurations to generate and complete the search space explore an area of the user-defined configuration space. The results show that lazy product discovery has equalized performance between our method and a state-of-the-art approach. Moreover, our method requires all diagrams to be compiled in another feature model. Furthermore, the method succeeds when more efficient, because-based engines fail to find a valid configuration.

Lienhardt et al. 2020

Abstract— Highly-configurable software systems can have thousands of inter-dependent configuration options. In such spaces, discovering a valid product configuration among all possible configurations is time consuming. This paper proposes a method to efficiently discover valid configurations for some selected options for complex and error-prone systems. The method is based on a feature model and a feature model-based search space. It iteratively explores the search space by selecting the configuration options of each subsystem. The proposed method is evaluated on two benchmarks in large highly-configurable feature models with interdependent features. We consider the number of configurations to generate and complete the search space explore an area of the user-defined configuration space. The results show that lazy product discovery has equalized performance between our method and a state-of-the-art approach. Moreover, our method requires all diagrams to be compiled in another feature model. Furthermore, the method succeeds when more efficient, because-based engines fail to find a valid configuration.



Knowledge Compilation Beyond BDDs



Thomas Thüm

What is the Goal of this Challenge?

Make Aware of Scalability Problems

Avoid Redundant Effort (cf. Publication Bias)

What is the Goal of this Challenge?

Make Aware of Scalability Problems

Avoid Redundant Effort (cf. Publication Bias)

Foster Exchange on Solutions

Join Forces

What is the Goal of this Challenge?

Make Aware of Scalability Problems

Avoid Redundant Effort (cf. Publication Bias)

Foster Exchange on Solutions

Join Forces

Promote Knowledge Compilation

Advance State-of-the-Art

What is the Scope of this Challenge?

Knowledge Compilation (e.g., BDDs)

Large-Scale Configuration Spaces (e.g., Linux)

What is the Scope of this Challenge?

Knowledge Compilation (e.g., BDDs)

Large-Scale Configuration Spaces (e.g., Linux)

Partial Models and Older (i.e., Smaller) Versions

Incremental Knowledge Compilation (for Evolution)

What is the Scope of this Challenge?

Knowledge Compilation (e.g., BDDs)

Large-Scale Configuration Spaces (e.g., Linux)

Partial Models and Older (i.e., Smaller) Versions

Incremental Knowledge Compilation (for Evolution)

Under- or Over-Approximations

(Un)successful Attempts

Knowledge Compilation Challenge for Variability

