Analyzing Product Lines and More With Binary Decision Diagrams

SoftVarE: Topics for Theses and Projects | Tobias Heß | 14.07.2021
About Me

Short CV

- 2012 - 2017, B.Sc. Mathematics in Ulm
- 2021 - , PhD candidate @ SoftVarE

Research Interests

- Binary Decision Diagrams and variants
- Knowledge Compilation in general
- Product-Line Analysis (e.g., Sampling)
Background

Product-Line Engineering
Motivation

Customer A development → Product X
Products have overlapping features (e.g., code fragments)

Customer B development → Product Y
Problem: Clone & Own

Product X

fork

Product Y

fork

Product Z

fork

Product Y' maintenance

Product Z' maintenance

Product Y'

cherry pick
Solution Approach: Feature-Driven Development

Synopsis

- Develop features not products.
- Use preprocessors and build systems to compose products from selections of features.
(Software) Product Lines

Terminology

- Configuration := Selection of Features
- Each configuration induces a product.

Challenges

- Features may depend on each other.
- Features may exclude each other
- Features may be optional or mandatory

⇒ Not all configurations are valid.
⇒ Need to model constraints and dependencies.
Feature Models

Feature Diagram

Car
  Carbody
  Radio
    Ports
      USB
      CD
    Navigation
      DigitalCards
      GPSAntenna
    Bluetooth
  Gearbox
    Manual
    Automatic

USA ⇒ ¬Manual
Product-Line Analyses

- Car
  - Carbody
  - Radio
  - Gearbox
    - Ports
      - USB
      - CD
      - DigitalCards
    - Navigation
    - Bluetooth
    - Manual
    - Automatic
  - Europe
  - USA

USA ⇒ ¬Manual
Analysis Process

**Feature Model**

- Carbody
- Radio
- Ports
- USB
- CD
- Navigation
- DigitalCards
- Europe
- USA
- GPSAntenna
- Gearbox
- Manual
- Automatic

**Propositional Formula**

\[(x_1 \lor \neg x_4) \land (\neg x_2 \text{ XOR } x_7) \land \ldots\]

**Conjunctive Normal Form (3CNF)**

\[(x_1 \lor \neg x_4) \land (\neg x_2 \lor x_7) \land (x_2 \lor \neg x_7) \land \ldots\]

**SAT Solver**

- satisfiable?
- NP-complete

**#SAT Solver**

- #satisfiable?
- #P-complete

(Tseytin Transformation)
Problems

- SAT solvers are fast (few seconds at most), but many solver runs are necessary.
- #SAT solvers are orders of magnitude slower (e.g., minutes, hours, or days).
- Other analyses/applications are not expressible by means of CNF.

Approach: Knowledge Compilation

- Some data structures allow for SAT, #SAT, ... to be solved efficiently
- Perform an expensive computation once $\rightarrow$ knowledge compilation artefact
- Profit™
Analysis Process w/ Knowledge Compilation

Feature Model

Car
  - Carbody
  - Radio
  - Ports
    - USB
    - CD
  - Navigation
  - Bluetooth
  - Manual
  - Automatic

USA ⇒ ¬Manual

Propositional Formula

\((x_1 \lor \neg x_4) \land (\neg x_2 \oplus x_7) \land \ldots\)

Tseytin Transformation

Conjunctive Normal Form (3CNF)

\((x_1 \lor \neg x_4) \land (\neg x_2 \lor x_7) \land (x_2 \lor \neg x_7) \land \ldots\)

Binary Decision Diagram

Query

\text{satisfiable?}
\begin{align*}
\text{constant} & \quad \#\text{satisfiable?} \\
\text{linear (wrt. nodes)} & \quad \text{NP-complete or d-DNNF (Chico)}
\end{align*}

Knowledge Compilation
Binary Decision Diagrams (BDDs)

Let \( f(a, b, c) = (\neg a \lor \neg b) \land (\neg a \lor \neg c) \land (\neg b \lor \neg c) \land (a \lor b \lor c) \).
OBDDimal

- Library to construct BDDs.
- Development started in January 2021.
- Developed in Rust.

Design Goals

- Extensible
- Usability (Usable by non-expert users)
- Parallelized
- Performance
**Efficiently Storing BDDs in Locally Lockable Hash Tables**

**Goals**

- Devise, implement, and evaluate a hash-based data structure with fine synchronization.
- Survey the literature for approaches to such a data structure.

**Remarks**

- Knowledge in Rust would be helpful.
- Open topic! Ideas are very welcome.

**Bonus**

- Likely to yield a research paper.
ddueruem(-web)

- **ddueruem**: Wrapper to interface with legacy BDD libraries.
- **ddueruem-web**: ddueruem as a RESTful (web)service with frontend
- Development started in January 2021.
- Developed in Python
  - **ddueruem**: Python + ctypes
  - **ddueruem-web**: Python + Django + Docker + Singularity + git
Frontend Development for ddueruem-web

Goals

- Development of a modern frontend for ddueruem-web using state-of-the-art frameworks (vue.js, react, web assembly, ...)

Remarks

- Open topic! Own ideas (e.g., regarding implementation technologies) are very welcome.
Integration of ddueruem-web Into FeatureIDE [P | B]

Goals

- Connect ddueruem-web to FeatureIDE via the RESTful API.
- Extend ddueruem to compute and cache analyses efficiently.

Remarks

- Decent knowledge in Java is mandatory, knowledge in Python helpful but not required.
- Open topic! Own ideas (e.g., regarding implementation technologies) are very welcome.
Incremental BDD Construction [M|B]

 Goals

 ▶ Devise and prove techniques to update BDDs and construct them incrementally.
 ▶ Implement a proof of concept in ddueruem (or OBDDimal) along with caching techniques.
 ▶ Test and evaluate your implementation.

 Remarks

 ▶ Knowledge in Python / Rust is helpful but not required.
 ▶ Open topic! Own ideas are very welcome.

 Bonus

 ▶ Likely to yield a research paper.
Theses and Projects

Divide-and-Conquer Construction of BDDs
\[ BDD(F, \pi_1 = (x_1, x_2, x_3, x_4, x_5, x_6)) \]

\[ BDD(F, \pi_2 = (x_1, x_3, x_5, x_2, x_4, x_6)) \]
Boolean Formulas as Graphs

- Variable ↔ Node
- Connect variables that share clauses by (weighted) edges.
D’n’C Construction of BDDs Based on Graph Decomposition

Goals

▷ Devise algorithms to order variables and clauses based on graph decomposition.
▷ Devise BDD construction techniques based on graph decomposition.
▷ Implement and evaluate your algorithms.

Remarks

▷ Knowledge in Python is helpful but not required.
▷ Open topic! Own ideas and points of emphasis are very welcome.

Bonus

▷ Likely to yield a research paper.
Theses and Projects

BDDs for Static Program Analysis
Static Code / Program Analysis

Synopsis

- White-box testing technique
- Formal requirements are verified during compilation / ahead of runtime.
- Curly lines in your IDE stem from static code analyses.

Role of BDDs

BDDs are used as constraint storage / datalog and act as backend of the analyzer.
BDDs for Static Program Analysis: Literature and Corpus Study

Goals

- Survey the literature to map the state-of-the-field on BDDs in SPA.
- Survey the literature to map the state of BDD-based tools in SPA.
- Establish a collection of BDD-based analyses.

Remarks

- Open topic! Own ideas and points of emphasis are very welcome.

Bonus

- Likely to yield a research paper.
Interested?

tobias.hess@uni-ulm.de

https://github.com/SoftVarE-Group/OBDDimal

https://github.com/h3ssto/ddueruem

For further information on the topics:

https://www.uni-ulm.de/in/sp/teaching/topics-for-theses-and-projects/