# Reasoning about Edits to Feature Models

Thomas Thüm
University of Magdeburg
Germany

Don Batory
University of Texas
USA

Christian Kästner
University of Magdeburg
Germany

ICSE 2009
VANCOUVER

# Customizable Software


Movements in software development

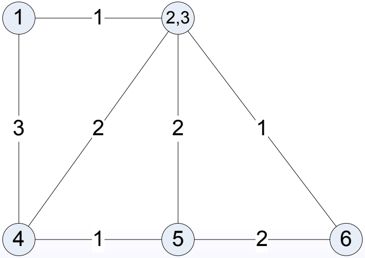
1. All-in-one software suitable for many purposes
2. Customized software for special requirements

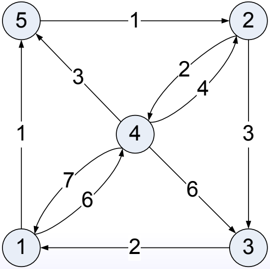
Software product line (SPL)

- Set of software-intensive systems that share a common, managed set of features
- Reuse of development artifacts
- Variability commonly expressed by feature models

# Customizable Graph Library

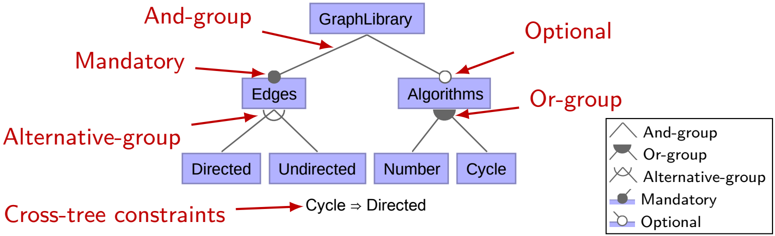
- Undirected and directed graphs
- Does the graph contain a cycle?
- What is the number of edges?
- Find a shortest path between two given nodes
- Customers have different needs



# Feature Models

- Features used to distinguish program variants (Kang et al. 90)
- Feature models specify SPL features and their combinations
- Graphically represented by feature diagrams:

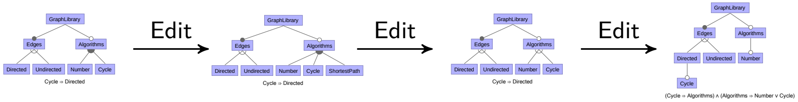



- Valid feature selections (products):

  $\{G, E, D\}$, $\{G, E, U\}$, $\{G, E, D, A, N\}$, $\{G, E, D, A, C\}$,
  $\{G, E, U, A, N\}$, $\{G, E, D, A, N, C\}$
- In practice: hundreds of features, millions of products

# Feature Models Evolve

Software product lines and their feature models evolve over time




Basic edits to feature models:

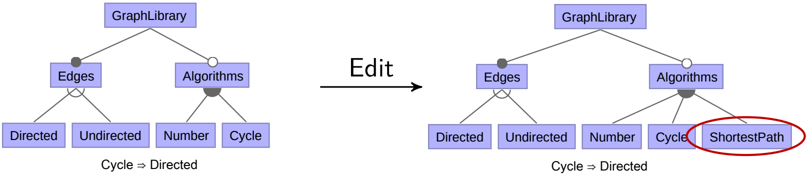
1. Changing a group type
2. Changing optional feature to mandatory or vice versa
3. Adding/removing a feature
4. Adding/removing a constraint
5. Moving a feature

Edit categories:

- Support domain engineers when editing feature models
- Guarantee that no products are added or deleted or both

# Generalization

- Adds new products to an SPL
- Examples: new features, less constraints



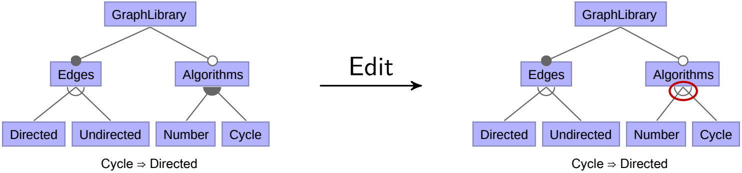

$$\{G,E,D\},\ \{G,E,U\},\ \{G,E,D,A,N\},\ \{G,E,D,A,C\},\ \{G,E,U,A,N\},\ \{G,E,D,A,N,C\}$$

$$\subset$$

$$\{G,E,D\},\ \{G,E,U\},\ \{G,E,D,A,N\},\ \{G,E,D,A,C\},\ \{G,E,D,A,S\},\ \{G,E,U,A,N\},\ \{G,E,U,A,S\},\ \{G,E,U,A,N,C\},\ \{G,E,D,A,N,S\},\ \{G,E,D,A,C,S\},\ \{G,E,U,A,N,S\},\ \{G,E,D,A,N,C,S\}$$

# Specialization

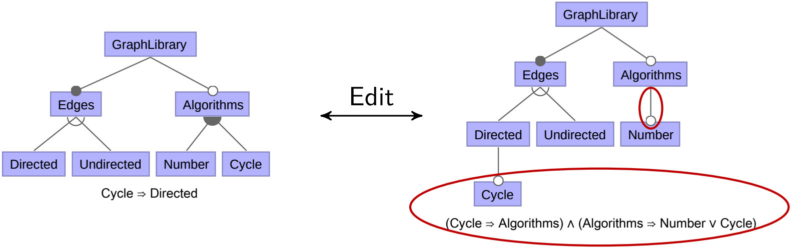
- Removes products from an SPL
- Examples: removed features, additional constraints, staged configuration




$$\{G,E,D\}, \{G,E,U\}, \{G,E,D,A,N\}, \{G,E,D,A,C\}, \{G,E,U,A,N\}, \color{red}{\{G,E,D,A,N,C\}} \supset \{G,E,D\}, \{G,E,U\}, \{G,E,D,A,N\}, \{G,E,D,A,C\}, \{G,E,U,A,N\}$$

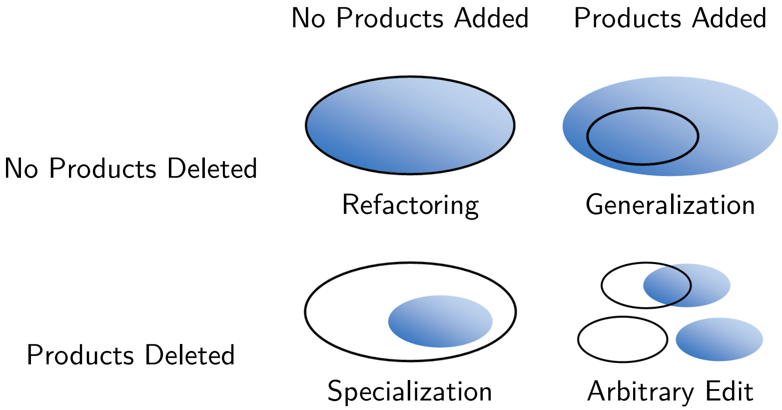
# Refactoring

- Changes the feature model without affecting the SPL
- Useful to improve readability, maintainability, and extensibility
- Examples: moving features, rewriting constraints



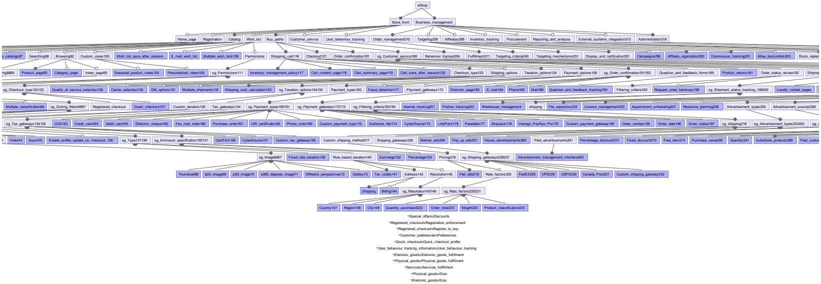

$$
\begin{array}{c}
\{G,E,D\},\ \{G,E,U\},\\
\{G,E,D,A,N\},\ \{G,E,D,A,C\},\\
\{G,E,U,A,N\},\ \{G,E,D,A,N,C\}
\end{array}
=
\begin{array}{c}
\{G,E,D\},\ \{G,E,U\},\\
\{G,E,D,A,N\},\ \{G,E,D,A,C\},\\
\{G,E,U,A,N\},\ \{G,E,D,A,N,C\}
\end{array}
$$

# Categories of Edits



# Our Goal

1. Automatically categorize an edit even for large feature models
2. Calculate examples for added and deleted products if available

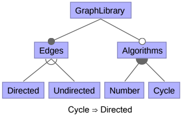



Mendonca, University of Waterloo

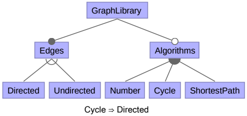
Easy to determine for small feature models, but a matter of scale!

# 1. Enumeration

Determine and compare the set of all products for both software product lines




Edit $\longrightarrow$




$\{G, E, D\}$, $\{G, E, U\}$,
$\{G, E, D, A, N\}$, $\{G, E, D, A, C\}$,
$\{G, E, U, A, N\}$, $\{G, E, D, A, N, C\}$

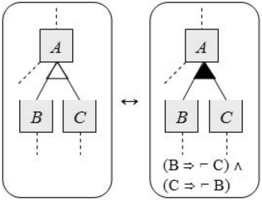
?

$\{G, E, D\}$, $\{G, E, U\}$,
$\{G, E, D, A, N\}$, $\{G, E, D, A, C\}$,
$\{G, E, D, A, S\}$, $\{G, E, U, A, N\}$,
$\{G, E, U, A, S\}$, $\{G, E, U, A, N, C\}$,
$\{G, E, D, A, N, S\}$,
$\{G, E, D, A, C, S\}$,
$\{G, E, U, A, N, S\}$,
$\{G, E, D, A, N, C, S\}$

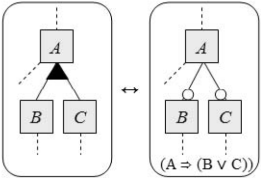
Enumeration does not scale!

# 2. Sound Operations

- Introduced by Alves et al. in 2006
- Catalogue of operations that are known to be a refactoring
- Catalogues for generalization and specialization




(a) *Replace Alternative* Refactoring




(b) *Replace Or* Refactoring

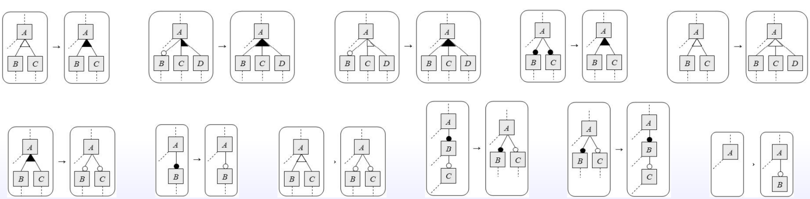
# 2. Sound Operations

Advantages:

- Intuitive: only special operations are allowed

Disadvantages:

- How to compare feature models build from scratch?
- What if we want to use edits not in the operation set, e.g., moving a feature?
- Requires hard-to-use structural editors or path-finder algorithms
- Different catalogues necessary for other variability models

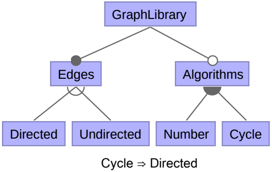


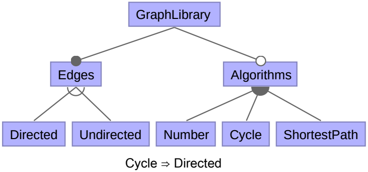
# 3. Reasoning using Propositional Formulas

- Proposed by Sun et al. in 2005, Janota and Kiniri in 2007
- Are all products from feature model $f$ available in $g$?

$$\begin{array}{ccc} \text{Feature model } f & & \text{Feature model } g \\ \downarrow & & \downarrow \\ \text{Propositional formula } P(f) & & \text{Propositional formula } P(g) \\ & \searrow \quad \swarrow & \\ & P(f) \Rightarrow P(g) \text{ is a tautology} & \\ & \downarrow & \\ & P(f) \wedge \neg P(g) \text{ is unsatisfiable} & \\ & \downarrow & \\ & \text{Solver (SAT, BDD, CSP, \ldots)} & \end{array}$$

Standard translation into a propositional formula (Batory 05)




$$
\begin{array}{l}
(G \wedge \\
(E \Rightarrow G) \wedge (A \Rightarrow G) \wedge (G \Rightarrow E) \wedge \\
(E \Rightarrow D \vee U) \wedge (\neg D \vee \neg U) \wedge \\
(A \Rightarrow N \vee C) \wedge (N \Rightarrow A) \wedge \\
(C \Rightarrow A) \wedge \\
(C \Rightarrow D))
\end{array}
\wedge \neg
\begin{array}{l}
(G \wedge \\
(E \Rightarrow G) \wedge (A \Rightarrow G) \wedge (G \Rightarrow E) \wedge \\
(E \Rightarrow D \vee U) \wedge (\neg D \vee \neg U) \wedge \\
(A \Rightarrow N \vee C \vee S) \wedge (N \Rightarrow A) \wedge \\
(C \Rightarrow A) \wedge (S \Rightarrow A) \wedge \\
(C \Rightarrow D))
\end{array}
$$

The propositional formula above is satisfiable if and only if the edit deletes products

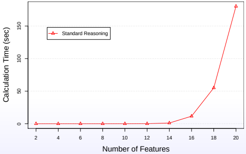
# 3. Reasoning using Propositional Formulas

Advantages:

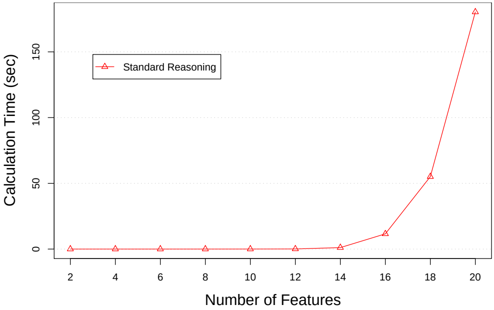
- Cross-tree constraints can be altered arbitrary
- Edits might be unknown
- All edits can be classified
- Applicable to all variability models that can be represented as a propositional formula
- Variability models of different types can be compared

Disadvantages:

- Restricted to feature models with the same feature set
- Performance!



# 3. Reasoning using Propositional Formulas

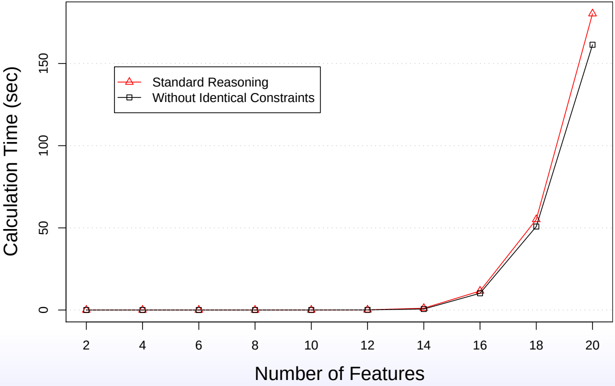



- Off-the-shelf solvers need propositional formulas to be in conjunctive normal form (CNF)
- $P(f) \wedge \neg P(g)$ needs to be converted into CNF
- Converting $P(f)$ is easy because the structure of feature models is CNF-like
- But: the negation of $P(g)$ is very time consuming

# Simplified Reasoning - First Optimization

Removing identical constraints:

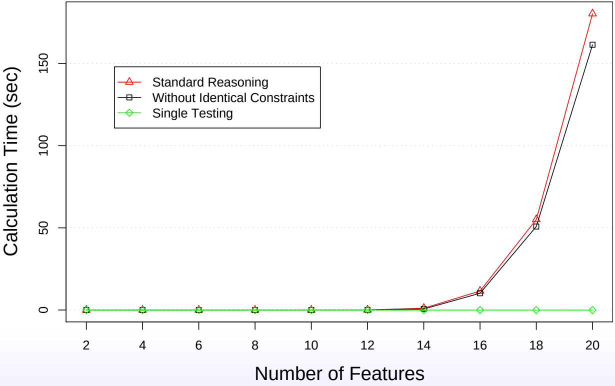
- $P(f) \wedge \neg P(g) \equiv P(f) \wedge \neg p_g$, where $P(g) = p_g \wedge c$ and $c$ are unchanged rules
- The formula to solve is smaller



# Simplified Reasoning - Second Optimization

Single testing:

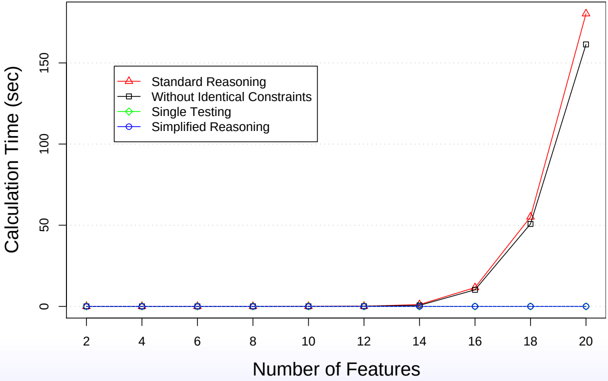
- $P(f) \wedge \neg p_g \equiv P(f) \wedge \neg \bigwedge_{1 \leq i \leq n'} R_i \equiv \bigvee_{1 \leq i \leq n'} (P(f) \wedge \neg R_i)$
- Splitting formula in multiple small SAT problems reduces calculation time



## Simplified Reasoning - Third Optimization

Stop early:

- Stop if a satisfiable formula $P(f) \wedge \neg R_i$ is found
- Saves calculation time when many edits were applied



# Simplified Reasoning - Extensions

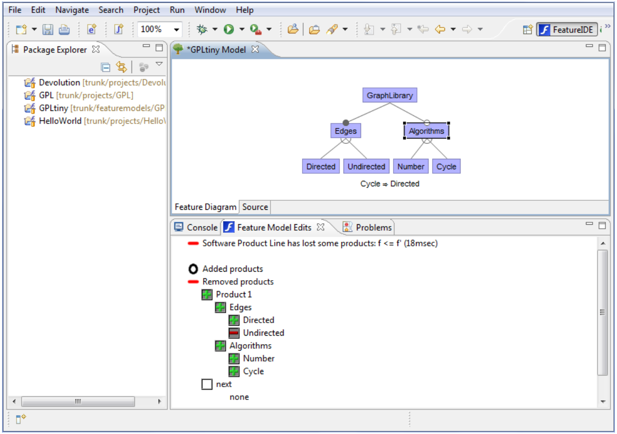
Special handling for added/removed features

- Added features are not selectable in the old feature model version
- Removed features are not selectable in the new feature model version

Special handling for abstract features

- Abstract features are features that do not change the implementation
- Example: removing an abstract feature is a refactoring
- See the paper for details

# Implemented in FeatureIDE

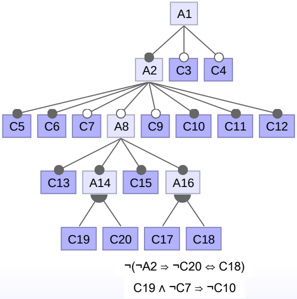



Formal tool demonstration tomorrow at 4:30pm

Available open source at `http://www.fosd.de/featureide`

# Does Our Approach Scale?

- Do our optimizations allow reasoning about large feature models within reasonable time?
- Unable to aquire sufficient amount of large feature models
- We generated 2000 feature models resembling feature models from practice

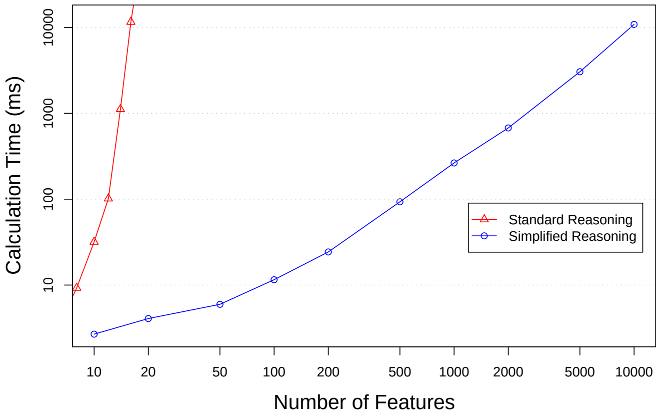



Feature diagram probabilities:
* And-group: 50%
* Or-group: 25%
* Alternative-group: 25%
* Optional child: 50%
* Maximum number of children: 10

Cross-tree constraints:
* 10% from number of features
* Between 2 and 5 variables

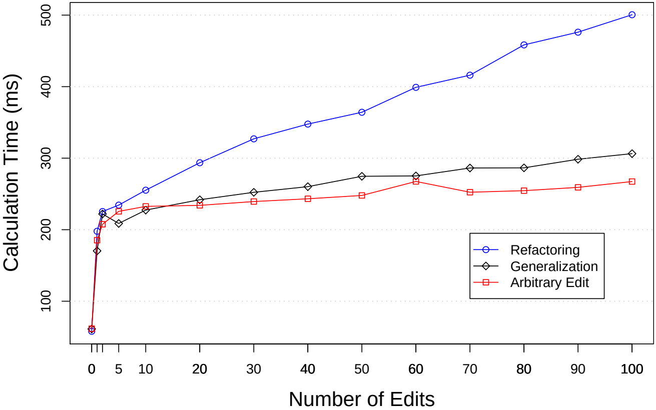
All 2000 generated feature models are avaible on FeatureIDE's website for comparative studies.

# Scalability - Number of Features




- We applied 10 edits to each feature model
- Calculation time to classify edits increases almost linearly

# Scalability - Number of Edits




- Edits applied to feature models with 1000 features
- Nearly constant time, independent of amount of edits

# Threats to Validity

Internal validity

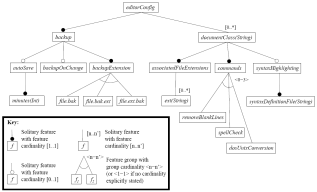
- Influence of computing power
  - Windows XP lab PC with 2.4GHz and 2GB RAM
- Calculation time may depend on certain shapes of feature models or edits
  - Known feature models statistically resembled
  - Edits taken from prior work

External validity

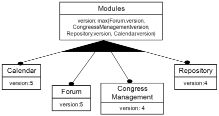
- Generated feature models may not represent industrial models
  - Survey of feature models
  - Generated feature models by others lead to similar results
- Edits might be incomplete or untypical in practice
  - No significant difference for independently generated models

# Related Work

Cardinality-based feature models (Czarnecki et al. 2005)



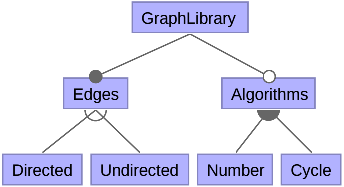

Extended feature models (Benavides et al. 2005)




Other variability models

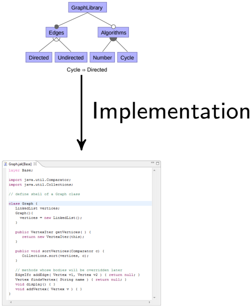
Automated analyses of feature models

- Count derivable products / detect void feature model
- Check if a combination is valid
- Detect dead features (not contained in any product)
- ...




Cycle $\Rightarrow$ Directed

# Related Work




Check consistency of feature model and implementation (Safe composition, type checking)

- Metzger et al. 2007
- Thaker et al. 2007
- Kästner and Apel 2008

# Conclusion & Future Work

Conclusion

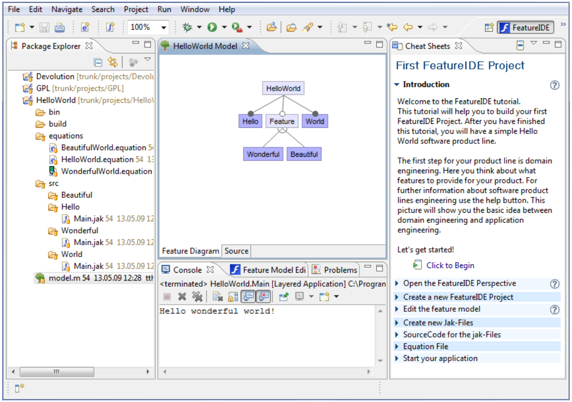
- Edits on feature models can be categorized in refactorings, generalizations, specializations and arbitrary edits
- Reasoning with propositional formula calculates the category of an edit
- Our optimizations scale this approach for feature models
  - with more than 1000 features
  - with more than 100 edits on a feature model

Future work

- Generalizing analysis to other variability models
- Finer distinctions of categories
- Compact visual representation of all added and removed products

# Thank You

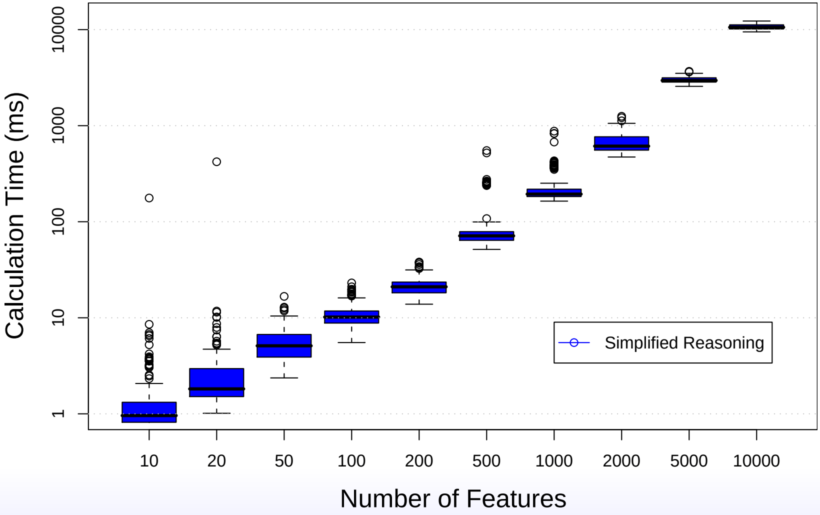
Sponsored in part by:



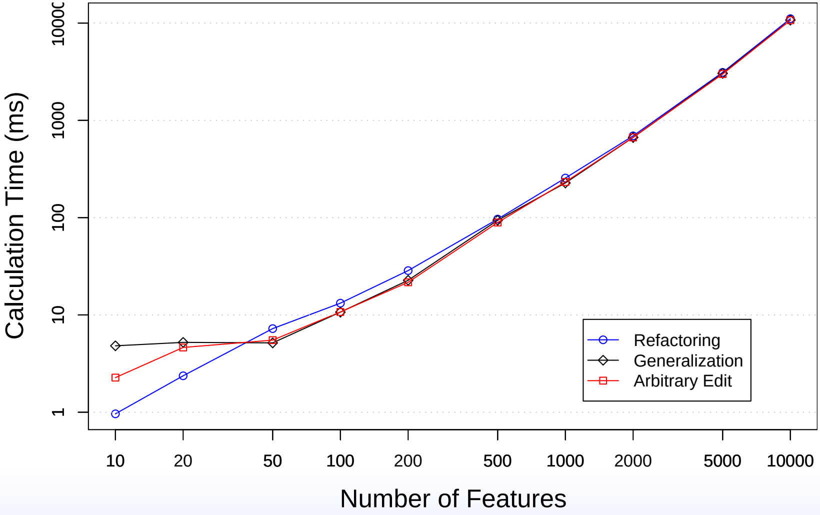

FeatureIDE: `http://www.fosd.de/featureide`

E-Mail: tthuem@st.ovgu.de

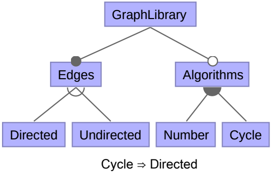
# Scalability - Distribution

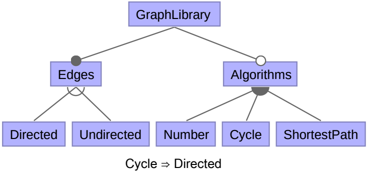


# Scalability - Categories



# Problem with Different Feature Sets



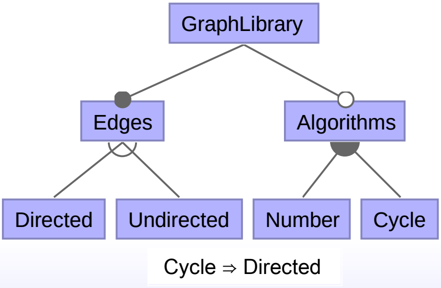

$$
\begin{array}{l}
(G \wedge \\
(E \Rightarrow G) \wedge (A \Rightarrow G) \wedge (G \Rightarrow E) \wedge \\
(E \Rightarrow D \vee U) \wedge (\neg D \vee \neg U) \wedge \\
(A \Rightarrow N \vee C) \wedge (N \Rightarrow A) \wedge \\
(C \Rightarrow A) \wedge \\
(C \Rightarrow D))
\end{array}
\quad \wedge\neg \quad
\begin{array}{l}
(G \wedge \\
(E \Rightarrow G) \wedge (A \Rightarrow G) \wedge (G \Rightarrow E) \wedge \\
(E \Rightarrow D \vee U) \wedge (\neg D \vee \neg U) \wedge \\
(A \Rightarrow N \vee C \vee S) \wedge (N \Rightarrow A) \wedge \\
(C \Rightarrow A) \wedge (S \Rightarrow A) \wedge \\
(C \Rightarrow D))
\end{array}
$$

The propositional formula above is satisfiable if and only if the edit deletes products

- Unfortunately, it is satisfiable: $\{G, E, D, S\}$
- The idea is to add a constraint $\neg S$ to the left formula

# Basic and Compound Edits

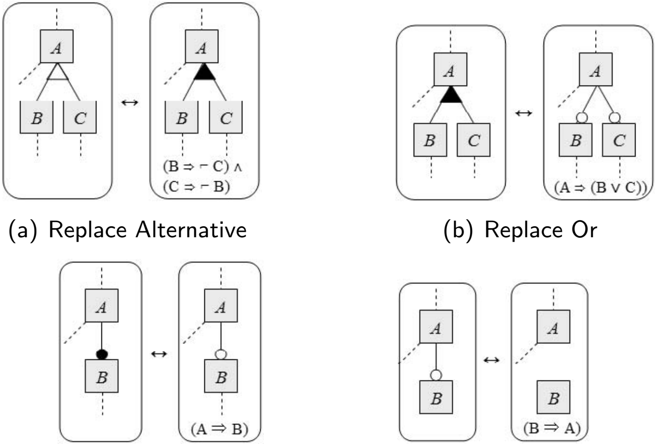
1. Changing a group type, e.g., an Or- to an Alternative-group
2. Changing optional feature to mandatory or vice versa
3. Adding/removing a feature without children
4. Adding/removing a constraint
5. Moving a feature including child features if existent




Compound edit examples:

- Removing a feature with children
- Altering a constraint

# Sound Operations - Refactorings



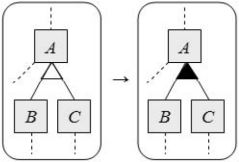

(a) Replace Alternative

(b) Replace Or

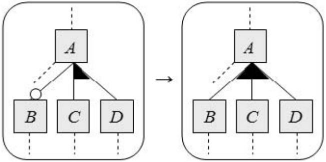
(c) Replace Mandatory

(d) Replace Optional

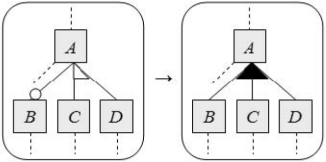
# Sound Operations - Generalizations



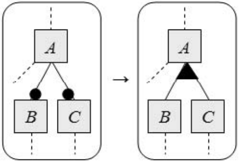

(a) Alternative to Or




(b) Collapse Optional and Or

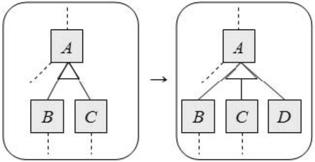



(c) Collapse Optional and Alternative

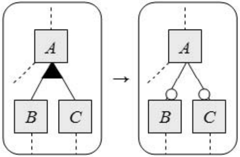



(d) Add Or between Mandatory

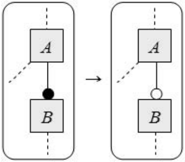
# Sound Operations - Generalizations



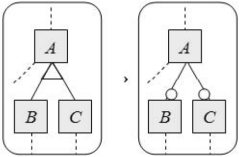

(e) Add New Alternative




(f) Or to Optional

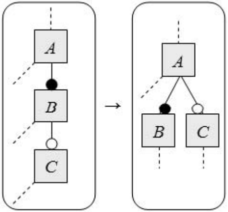



(g) Mandatory to Optional

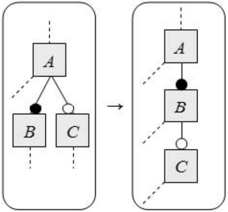



(h) Alternative to Optional

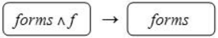
# Sound Operations - Generalizations



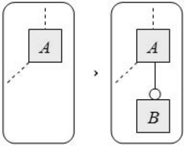

(i) Pull Up Node




(j) Push Down Node




(k) Remove Formula




(l) Add Optional Node