Feature Models

Model variability & commonality.

Standard semantics: \( \phi(\text{car}, \text{body}, \text{engine}, \text{gear}, \ldots) \)
Feature Models

Model variability & commonality.

Standard semantics:

$$\phi(\text{car, body, engine, gear, keyless-entry, power-locks})$$
Feature Models

Model variability & commonality.

Standard semantics: \( \phi(\text{car}, \text{body}, \text{engine}, \text{gear}, \ldots) \)
Feature Models

Model variability & commonality.

Standard semantics: \( \phi(\text{car}, \text{body}, \text{engine}, \text{gear}, \ldots) \)

- keyless-entry \( \rightarrow \) power-locks
- body
- engine
- gear
- keyless-entry
- power-locks
- electric
- gas
- manual
- automatic
Model variability & commonality.
Model variability & commonality.

Standard semantics:

\[ \phi(\text{car, body, engine, gear, ...}) \]
Reverse Engineering Syntax
Reverse Engineering Syntax
Reverse Engineering Syntax
Contents

- Motivation
- Syntax & Semantics (going there)
- Algorithm (going back again)
- Concluding remarks
Motivation
Syntax & Semantics (going there)
Algorithm (going back again)
Concluding remarks
Theoretical Motivation

- To deepen understanding of feature models
- To explore the relation between logics and FMs
- To characterize formulæ that are FMs without leftover constraint
To visualize variability given as systems of constraints.
**Applied Motivation**

- To visualize variability given as systems of constraints.
- To guide the user interactively in visualizing constraints.
Applied Motivation

- To visualize variability given as systems of constraints.
- To guide the user interactively in visualizing constraints.
- To support refactoring tools.
**Applied Motivation**

- To visualize variability given as systems of constraints.
- To guide the user interactively in visualizing constraints.
- To support refactoring tools.
- To support reverse engineering FMs from code.
Contents

- Motivation
- Syntax & Semantics (going there)
- Algorithm (going back again)
- Concluding remarks
Syntax: Going There

Solitary [1..1] → mandatory
Syntax: Going There

Solitary [1..1] $\rightarrow$ mandatory

Solitary [0..1] $\rightarrow$ optional
**Syntax: Going There**

Solitary [1..1] $\rightarrow$ mandatory

Solitary [0..1] $\rightarrow$ optional

Solitary [0..1] $\rightarrow$ grouped
Syntax: Going There

Solitary [1..1] → mandatory

Solitary [0..1] → optional

Solitary [0..1] → grouped

Group [1..1] → xor-group
Syntax: Going There

Solitary [1..1] $\rightarrow$ mandatory

Solitary [0..1] $\rightarrow$ optional

Solitary [0..1] $\rightarrow$ grouped

Group [1..1] $\rightarrow$ xor-group

Group [1..k] $\rightarrow$ or-group
Syntax: Going There

Solitary [1..1] → mandatory

Solitary [0..1] → optional

Solitary [0..1] → grouped

Group [1..1] → xor-group

Group [1..k] → or-group

Left-over constraints
Semantics: Going There
Semantics: Going There

An implication (hyper)graph
Motivation
Syntax & Semantics (going there)
Algorithm (going back again)
Concluding remarks
Why Is It So Hard?

1. Possibly no models corresponding to $\phi$
Why Is It So Hard?

1. Possibly **no** models corresponding to $\phi$

2. Possibly **many** models corresponding to $\phi$

```
\begin{tikzpicture}
  \node (a) at (0,0) {a};
  \node (b) at (-1,-1) {b};
  \node (c) at (1,-1) {c};
  \draw (a) -- (b);
  \draw (a) -- (c);
  \node (d) at (2,0) {a, c};
  \node (e) at (3,-1) {b};
  \node (f) at (4,-1) {c};
  \node (g) at (5,-2) {b};
  \draw (d) -- (e);
  \draw (d) -- (f);
  \draw (f) -- (g);
\end{tikzpicture}
```
Why Is It So Hard?

1. Possibly no models corresponding to $\phi$

2. Possibly many models corresponding to $\phi$

3. Brute-force infeasible
The root of a feature tree

A variable $r$ implied by all the other variables:

for all $i$. $\phi \rightarrow (f_i \rightarrow r)$
The root of a feature tree

A variable $r$ implied by all the other variables:

body $\rightarrow$ car, gas $\rightarrow$ car, . . .
Feature Hierarchy

Property

Test

\[ f \text{ is an ancestor of } g \text{ (descendant)} \]

Implication from descendant \((g)\) to ancestor \((f)\)

\[ g \rightarrow f \]
Feature Hierarchy

Property

Test

f is an ancestor of g (descendant)

Implication from descendant (g) to ancestor (f)

body → car, gas → car, . . .
Feature Hierarchy

Property

Test

$f$ is an ancestor of $g$ (descendant)

Implication from descendant ($g$) to ancestor ($f$)

Direct links: transitive reduction
Mandatory Features

Property

Test

$g$ is a mandatory subfeature of $f$

Biimplication between variables corresponding to $g$ and $f$

$$f \rightarrow g$$
Mandatory Features

Property

Test

\( g \) is a mandatory subfeature of \( f \)

Biimplication between variables corresponding to \( g \) and \( f \)

body \( \rightarrow \) car, car \( \rightarrow \) body, \ldots
Property

Test

And Groups

g is a mandatory subfeature of f

Biimplication between variables corresponding to g and f

And-groups: **cliques** in the graph
Recall that for an or-group:

\[ \phi \rightarrow (f \rightarrow f_1 \lor \cdots \lor f_k) \]

But then also

\[ \phi \rightarrow (f \rightarrow f_1 \lor \cdots \lor f_k \lor g) \]

holds for \( g \) other than \( f_i \).
Or Groups

- Implied disjunction can always be weakened!
- All implied disjunctions = oversized and too-many or-groups
- So detect *minimal* disjunctions
Implied disjunction can always be weakened!

All implied disjunctions = oversized and too-many or-groups

So detect *minimal* disjunctions

\( \{ \overline{f_i} \}_{i=1..k} \) is a prime implicant of \( \overline{f} \)

(see the paper)

Prime implicants are well studied in fault tolerance analysis
Or-Groups

Or-groups of features rooted in $f$

Find prime implicants of $\overline{f}$

---

electric $\land$ gas $\rightarrow$ engine
Or Groups

Or-groups of features rooted in $f$

Find prime implicants of $\overline{f}$

Property Test

manual $\land$ electric $\land$ gas $\rightarrow$ engine
Algorithm

1. if unstatisifiable then quit
2. remove & report dead features
3. compute implication graph & its transitive reduction
4. find and-groups by contracting cliques
5. find all or-groups and xor-groups candidates
Contents

- Motivation
- Syntax & Semantics (going there)
- Algorithm (going back again)
- Concluding remarks
Constructs an overapproximation (add a leftover constraint)

The graph constructed contains maximum information (complete)
Constructs an overapproximation (add a leftover constraint)

The graph constructed contains maximum information (complete)

Implemented using BDDs, algorithm by Coudert&Madre, 1992

Efficient and scalable (computing prime implicants is the bottleneck)
Semantic operations on feature models become logical operations on corresponding formulæ

**Merge:** $(\phi_{FM1} \rightarrow r) \land (\phi_{FM2} \rightarrow r)$

**Difference:** $\phi_{FM1} \land \neg\phi_{FM2}$

...
Future Work

- Generalize the kind of models extracted beyond FODA
- Implement complex refactorings using logical representations
- Experiment with extracting models from code
Successful exercise in semantics
Exhibited links between logical & relational phenomena and FM's
- implication graphs, transitive reduction, cliques, prime implicants

Effective extraction procedure
Implemented

Suggested ideas for future work