Feature **D**iagrams & **L**ogic **T**here and **B**ack **A**gain

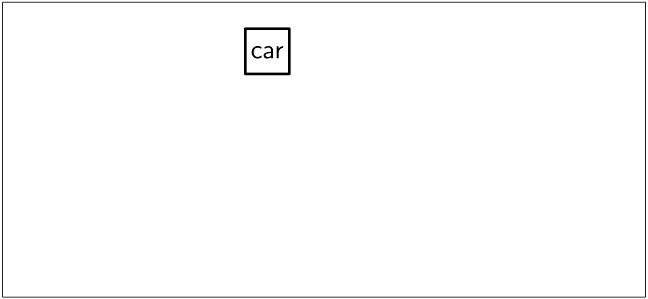
Krzysztof Czarnecki

University of Waterloo

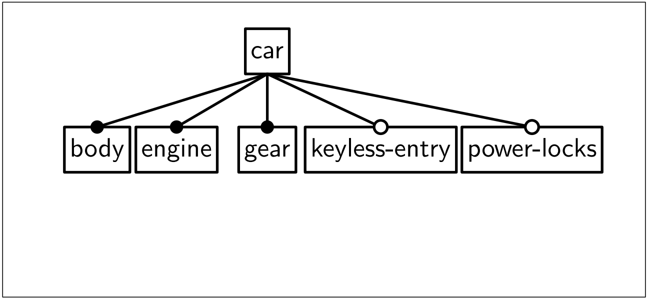
Andrzej Wąsowski

IT University of Copenhagen

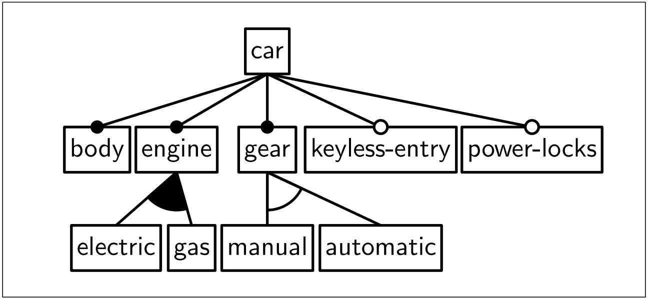
Feature **M**odels



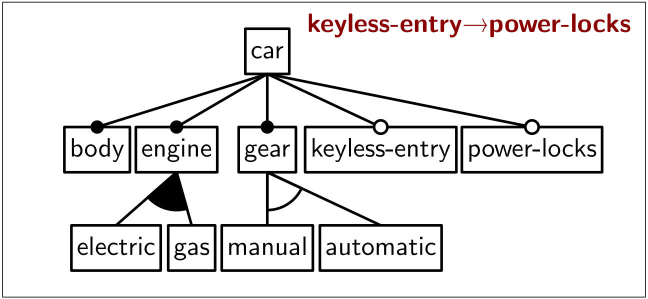
Feature **M**odels



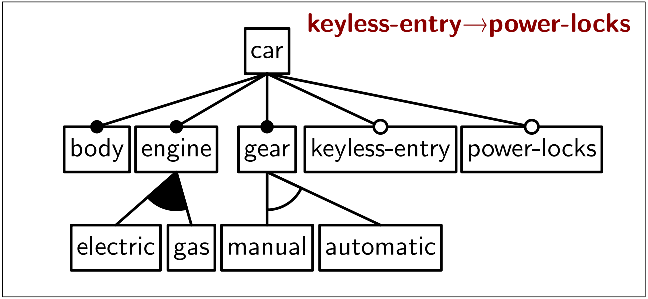
Feature Models



Feature Models

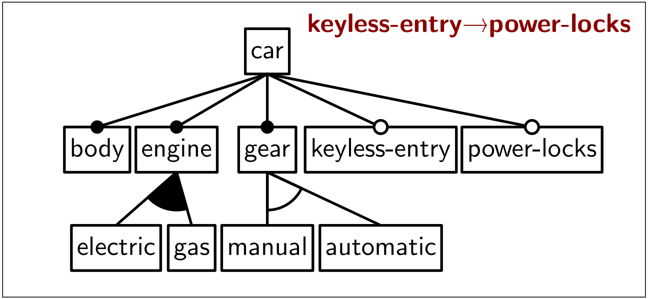


Feature Models



- Model variability & commonality.

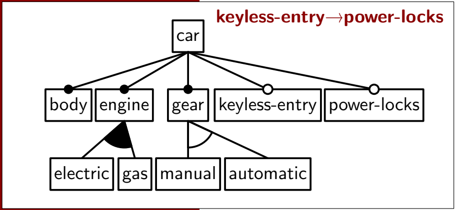
Feature Models



- Model variability & commonality.
- Standard semantics:

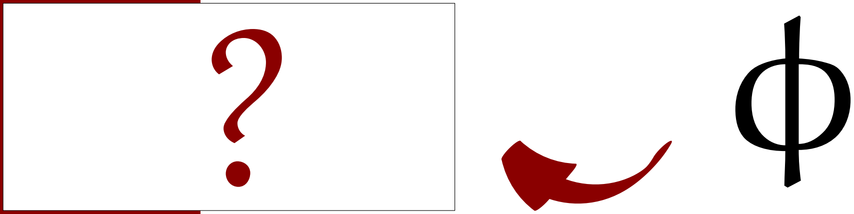
$$\phi(car, body, engine, gear, \dots)$$

Semantics

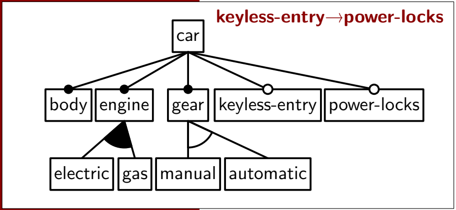


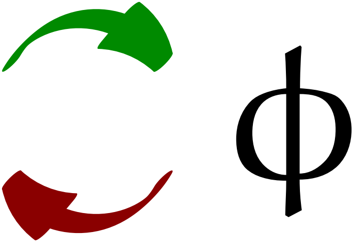
$$\phi$$

Reverse Engineering Syntax

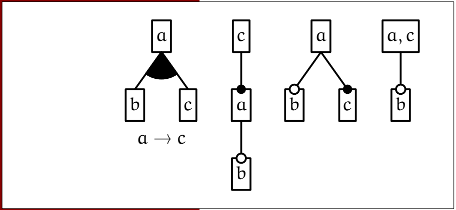


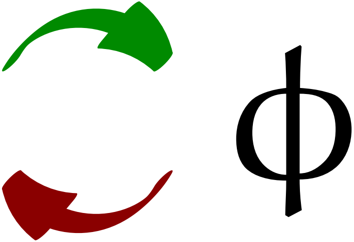
Reverse Engineering Syntax



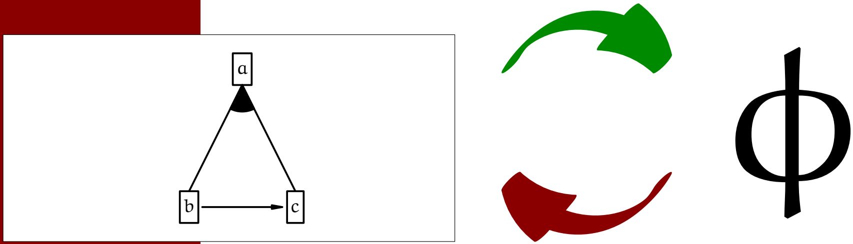


Reverse Engineering Syntax





Reverse **E**ngineering **S**yntax



Contents

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

Contents

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

Theoretical **M**otivation

- To deepen understanding of feature models
- To explore the relation between logics and FMs
- To characterize formulæ that are FMs without leftover constraint

Applied **M**otivation

- To visualize variability given as systems of constraints.

Applied **M**otivation

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

Applied **M**otivation

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

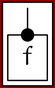
Applied **M**otivation

- 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

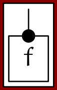
- Motivation
- Syntax & Semantics (going there)
- Algorithm (going back again)
- Concluding remarks

Syntax: **G**oing **T**here

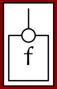


Solitary [1..1] $\rightarrow$ mandatory

Syntax: Going There

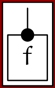


Solitary [1..1] → mandatory

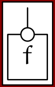


Solitary [0..1] → optional

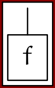
Syntax: **G**oing **T**here



Solitary [1..1] → mandatory

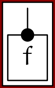


Solitary [0..1] → optional

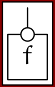


Solitary [0..1] → grouped

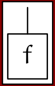
Syntax: **G**oing **T**here



Solitary [1..1] → mandatory



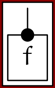
Solitary [0..1] → optional



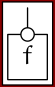
Solitary [0..1] → grouped

Group [1..1] → xor-group

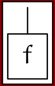
Syntax: **G**oing **T**here



Solitary [1..1] → mandatory



Solitary [0..1] → optional

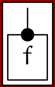


Solitary [0..1] → grouped

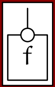
Group [1..1] → xor-group

Group [1..k] → or-group

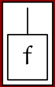
Syntax: Going There



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



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



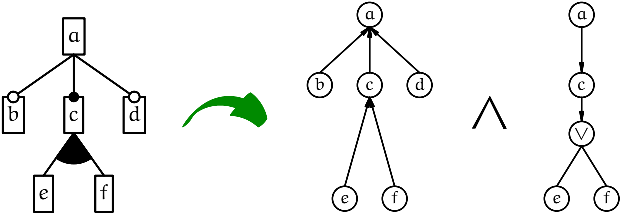
Solitary [0..1] $\rightarrow$ grouped

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

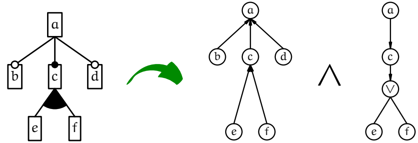
Group [1..k] $\rightarrow$ or-group

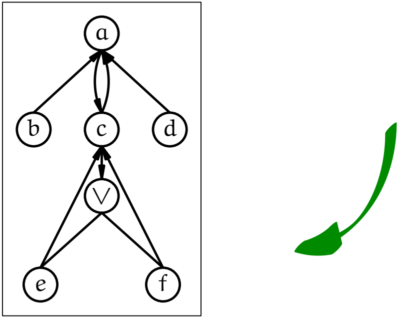
ϕ Left-over constraints

Semantics: Going There



Semantics: Going There





An implication (hyper)graph

Contents

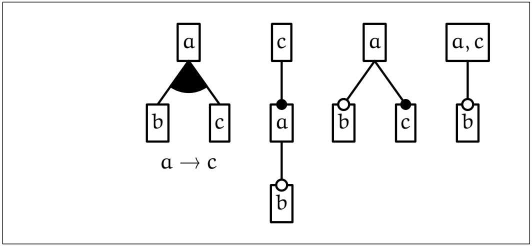
- Motivation
- Syntax & Semantics (going there)
- Algorithm (going back again)
- Concluding remarks

Why **I**s **I**t **S**o **H**ard?

1. Possibly **no** models corresponding to ϕ

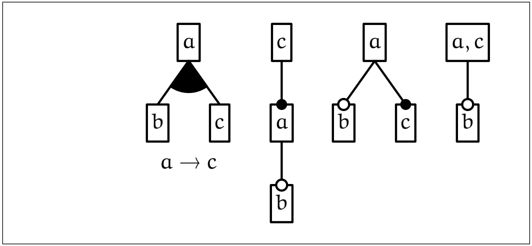
Why Is It So Hard?

1. Possibly **no** models corresponding to ϕ
2. Possibly **many** models corresponding to ϕ



Why Is It So Hard?

1. Possibly **no** models corresponding to ϕ
2. Possibly **many** models corresponding to ϕ



3. Brute-force **infeasible**

Root Feature

Property The root of a feature tree

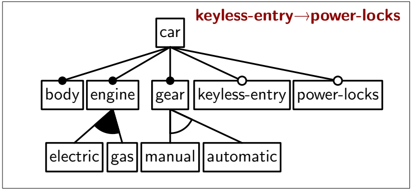
Test A variable r implied by all the other variables:

$$\text{for all } i.\ \phi \to (f_i \to r)$$

Root Feature

Property The root of a feature tree

Test A variable r implied by all the other variables:



$$\text{body} \rightarrow \text{car}, \text{gas} \rightarrow \text{car}, \ldots$$

Feature Hierarchy

Property f is an ancestor of g (descendant)

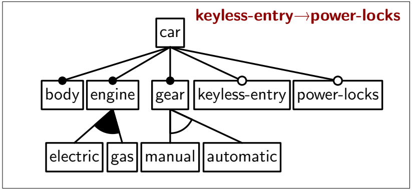
Test Implication from descendant (g) to ancestor (f)

$$\mathrm{g} \longrightarrow \mathrm{f}$$

Feature Hierarchy

Property f is an ancestor of g (descendant)

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

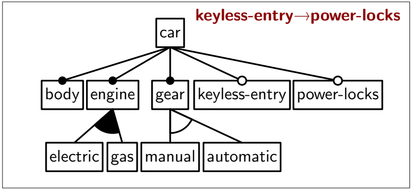


body $\longrightarrow$ car, gas $\longrightarrow$ car, ...

Feature Hierarchy

Property f is an ancestor of g (descendant)

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



Direct links: **transitive reduction**

Mandatory Features

Property g is a mandatory subfeature of f

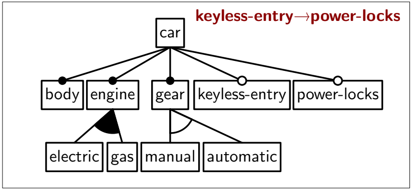
Test Biimplication between variables corresponding to g and f

$$f \rightarrow g$$

Mandatory Features

Property g is a mandatory subfeature of f

Test Biimplication between variables corresponding to g and f

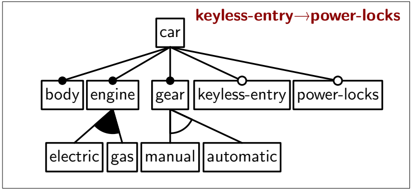


$\text{body} \rightarrow \text{car}, \text{car} \rightarrow \text{body}, \ldots$

And Groups

Property g is a mandatory subfeature of f

Test Biimplication between variables corresponding to g and f



And-groups: **cliques** in the graph

Or **G**roups

- Recall that for an or-group:

$$\phi \rightarrow (f \rightarrow f_1 \vee \cdots \vee f_k)$$

- But then also

$$\phi \rightarrow (f \rightarrow f_1 \vee \cdots \vee f_k \vee g)$$

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

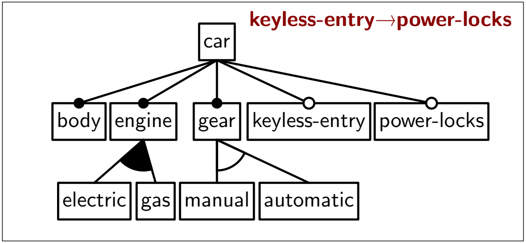
Or Groups

- Implied disjunction can always be weakened!
- All implied disjunctions = oversized and too-many or-groups
- So detect *minimal* disjunctions
- $\{\bar{f}_i\}_{i=1..k}$ is a **prime implicant** of $\bar{f}$ (see the paper)
- Prime implicants are well studied in fault tolerance analysis

Or Groups

Property Or-groups of features rooted in f

Test Find prime implicants of $\overline{f}$

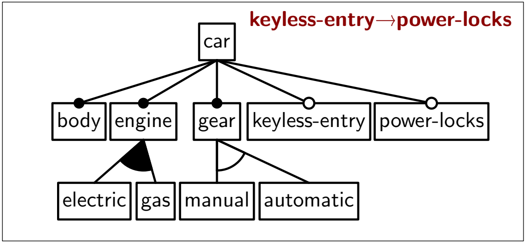


$$\overline{\text{electric}} \wedge \overline{\text{gas}} \rightarrow \overline{\text{engine}}$$

Or Groups

Property Or-groups of features rooted in f

Test Find prime implicants of $\overline{f}$



$$\overline{\text{manual}} \wedge \overline{\text{electric}} \wedge \overline{\text{gas}} \rightarrow \overline{\text{engine}}$$

Algorithm

1. if unstatisfiable 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

Discussion (I)

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

Discussion (I)

- 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)

Discussion (II)

Semantic operations on feature models become logical operations on corresponding formulæ

- Merge: $(\phi_{FM1} \rightarrow r) \wedge (\phi_{FM2} \rightarrow r)$
- Difference: $\phi_{FM1} \wedge \neg \phi_{FM2}$
- ...

Future **W**ork

- Generalize the kind of models extracted beyond FODA
- Implement complex refactorings using logical representations
- Experiment with extracting models from code

Summary

- Successful exercise in semantics
- Exhibited links between logical & relational phenomena and FMs
  - implication graphs, transitive reduction, cliques, prime implicants
- Effective extraction procedure
- Implemented
- Suggested ideas for future work