Feature Modeling

KV Product Line Engineering (343.354)

Dr. Roberto Lopez-Herrejon
Dr. Rick Rabiser

JOHANNES KEPLER UNIVERSITY LINZ

ISSE

16.3.2016

Feature Modeling

- Feature Modeling
  - Identifying and organizing the common and variable properties of concepts and their relationships
  - Outcome: *feature model*

- Concept
  - Something that is important in a domain
  - Ex. *House* if you domain is house construction

Feature Model

- A feature model is a tree whose
  - Root is a concept
  - Nodes are features
  - Edges establish relationships between features

- Restrictions
  - Only one concept per feature model
  - No cycles
    - because it is a tree

Root Concept

- Concepts are represented as boxes labeled with the name of the concept

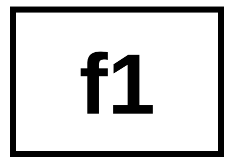
C

- Example

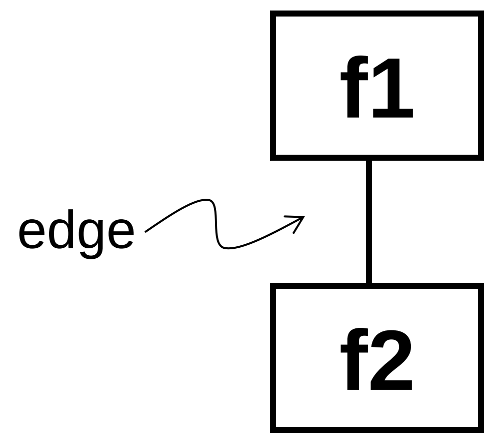
House

Modeling Features

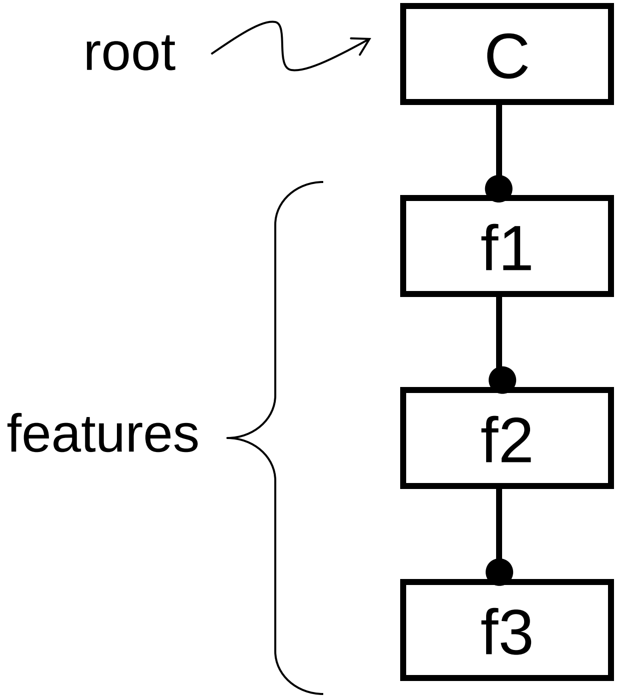
- Features also represented as labeled boxes



- Edges denote relations among features
  - Annotations used for
    - Different types: mandatory, optional
    - Group selectivity



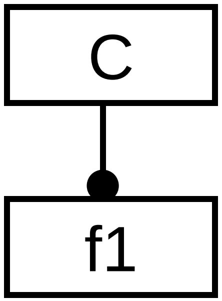
Hierarchical Relations Among Features

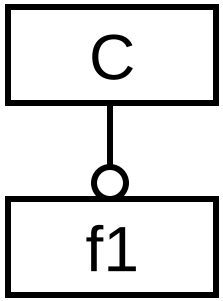


- Concept C is the root
- f1,f2, and f3 are features
- From C
  - direct feature: f1
  - indirect feature: f2, f3
- From f1
  - direct sub-feature: f2
  - indirect sub-feature: f3
- f1 is *parent* of f2
  - f2 is *child* of f1
- f1 is not parent of f3
  - f3 is not child of f1

Types of Features

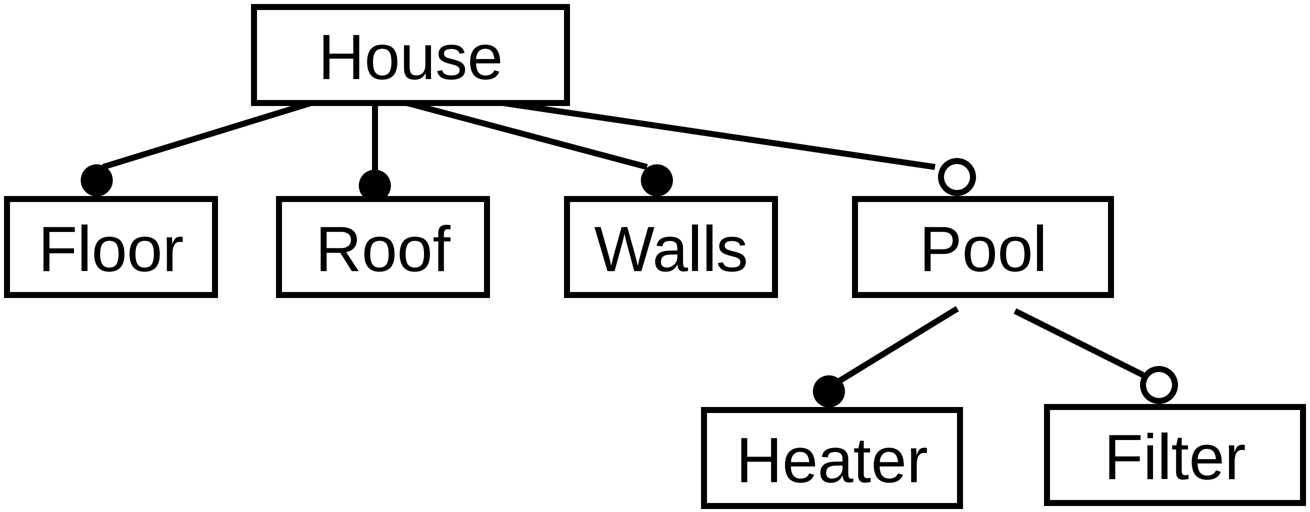
- Mandatory
  - Included if its parent is included
  - Denoted with filled circle
- Optional
  - If the parent is included it may or may not be included
  - Denoted with empty circle





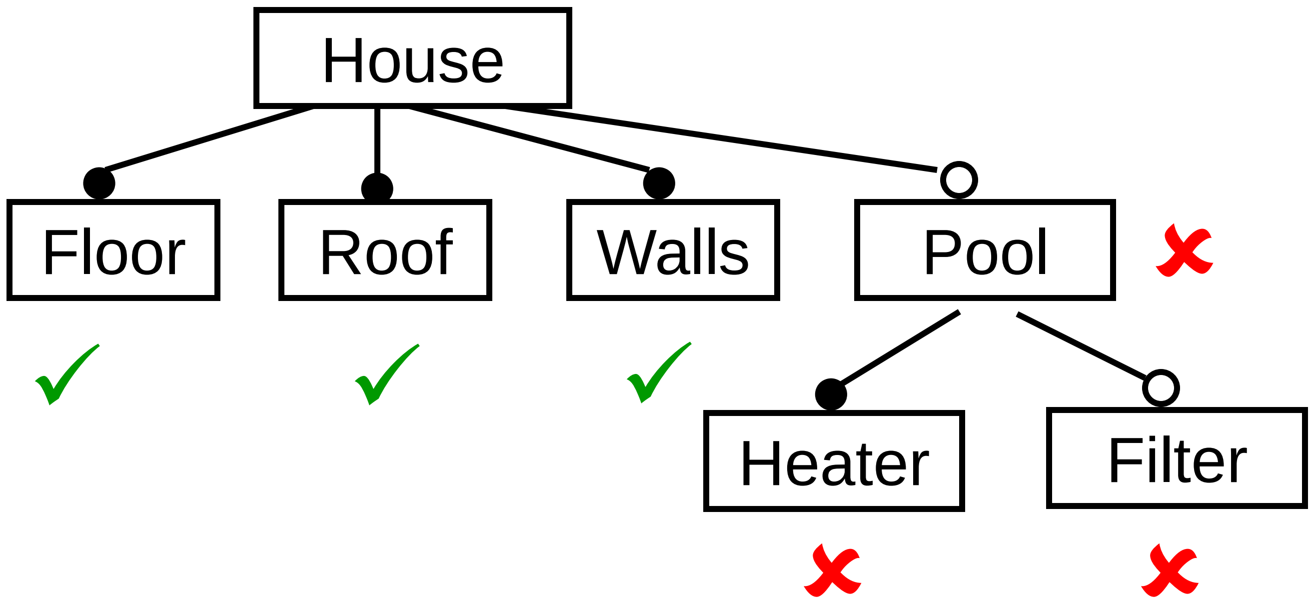
Root concept is always included

Types of Features Example



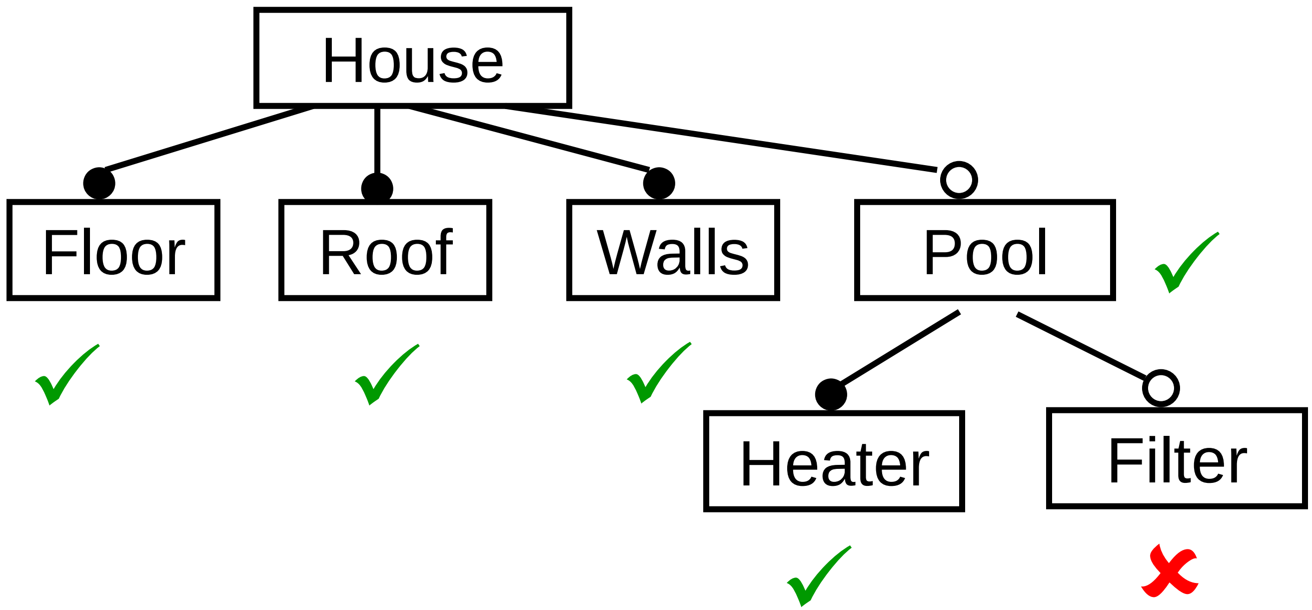
What types of houses can be created?

First Type



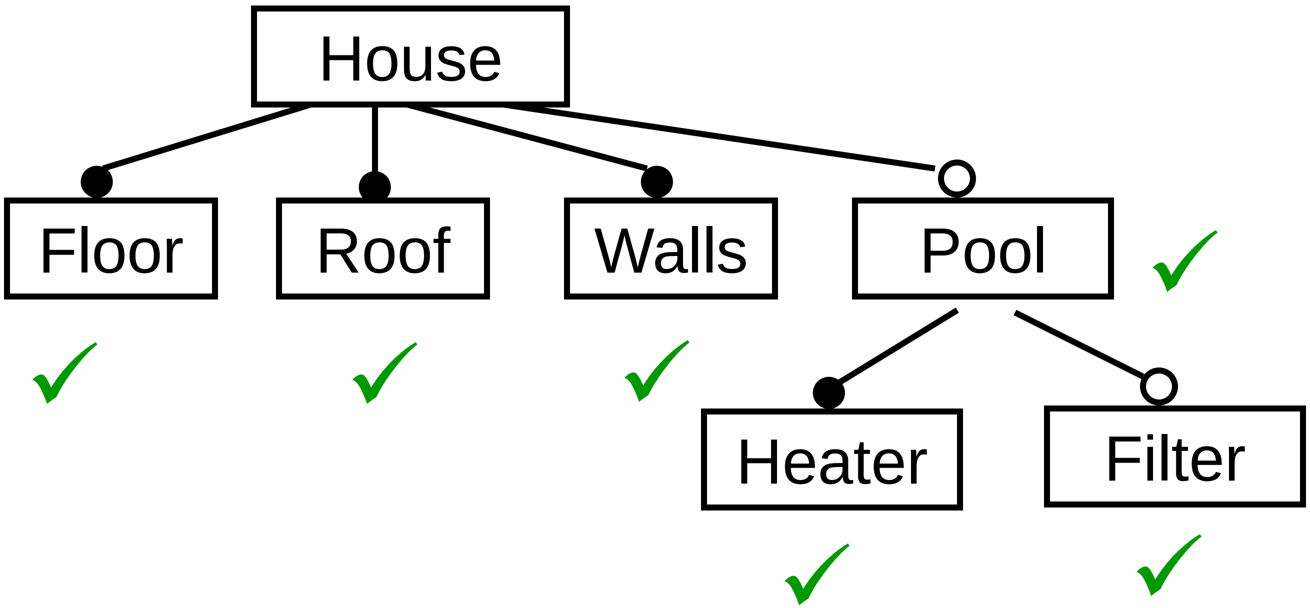
Features = { Floor Roof Walls }

Second Type



Features = { Floor Roof Walls Pool Heater }

Third Type



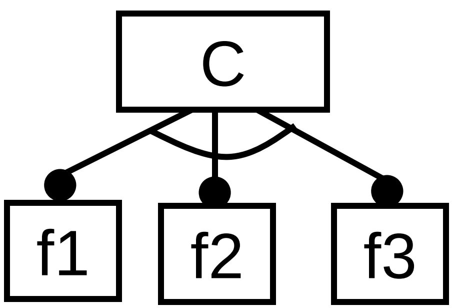
Features = { Floor Roof Walls Pool Heater Filter }

Some Terminology

- A concept instance is set of features that satisfy the constraints of a feature model
  - valid configuration of features
- In this context
  - concept instance = product line member

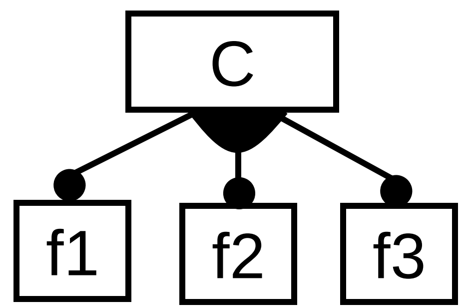
Alternative Features

- Exclusive-or
  - If parent included, select exactly one from the set
  - Denoted with an empty arc



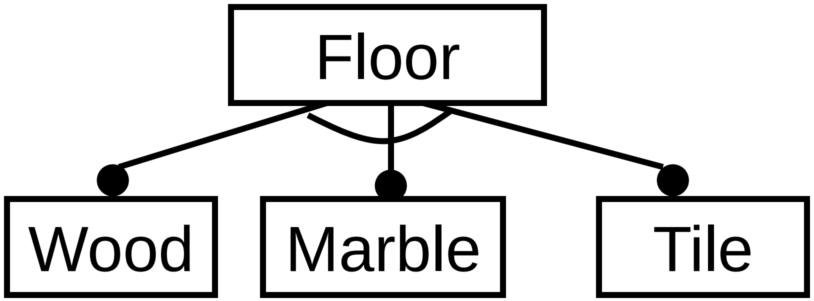
- a.k.a *Alternative-features*

- Inclusive-or
  - If the parent is included, select at least one from the set
  - Denoted with a filled arc



- a.k.a *Or-features*

Exclusive-Or Example



What houses can be created?

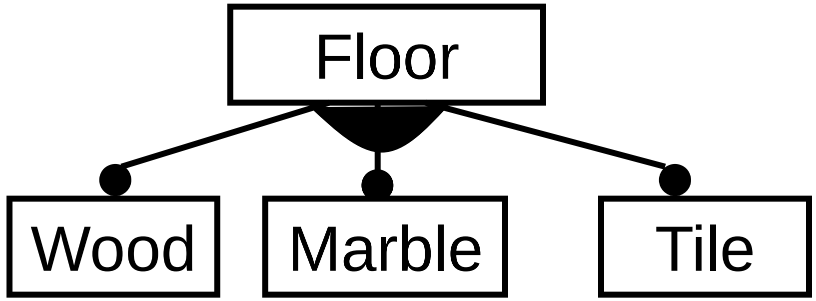
Floor Wood

Floor Marble

Floor Tile

Pick *Exactly* One

Inclusive-Or Example



What houses can be created?

Floor Wood

Floor Marble

Floor Tile

Floor Wood Marble

Floor Wood Tile

Anything else?

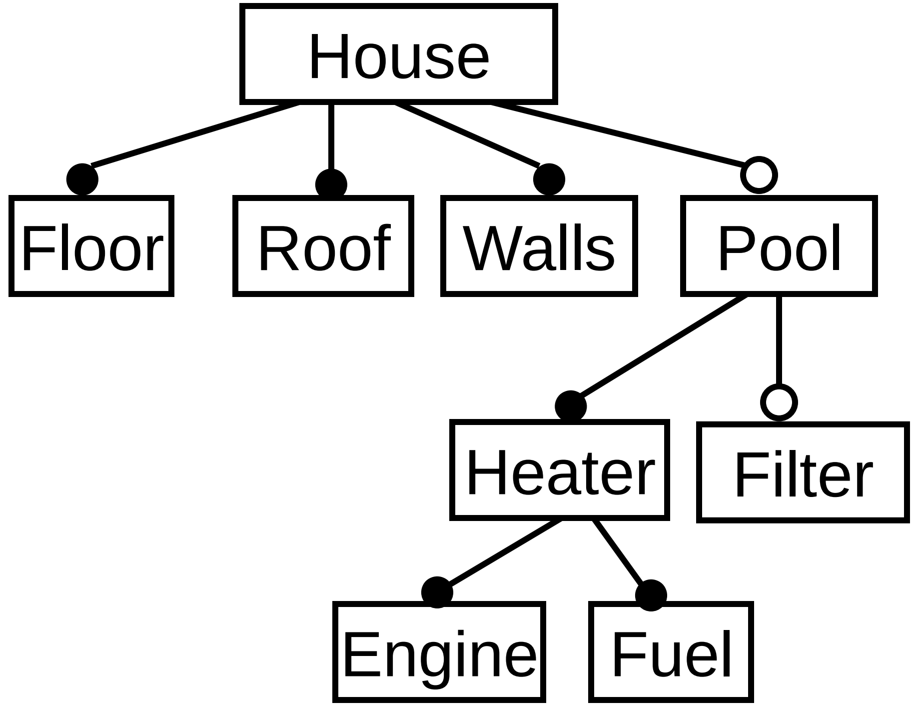
Floor Marble Tile

Floor Wood Marble Tile

Pick *At Least* One

Commonality

- Two levels
  - Concept level: features present in all programs
  - Feature level: the set of sub-features common to all programs that have a particular feature f
- Example
  - Concept: Floor, Roof, Walls
  - Feature Pool: Heater, Engine, Fuel



Feature Model Variability

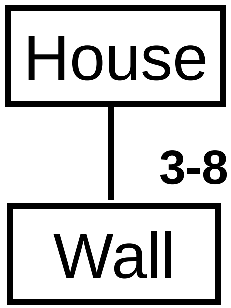
- Variable features are
  - Optional
  - Exclusive-or
  - Inclusive-or

- Variation Points
  - Nodes in the feature diagram that have variable features

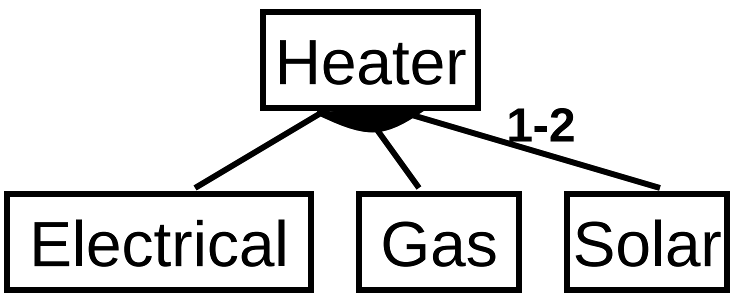
Cardinality

- Cardinality
  - The number of times an element of the feature diagram can occur in a configuration
- Feature cardinality
  - Number of times a feature can occur in a configuration
- Group cardinality
  - Denotes how many variants or choices from a group can occur in a configuration

Cardinality Examples



Our houses are built with
3 to 8 walls



Heater has
1 main energy source
1 backup source

In terms of Cardinality

Feature Type	Cardinality
Mandatory	1-1
Optional	0-1
Exclusive-or with n sub-features	1-1
Exclusive-or with n optional sub-features	0-1
Inclusive-or with n sub-features	1-n

Some Additional Information in Feature Models

- For each feature
  - Semantic description: what this feature means?
  - Rationale: why we need this feature?
  - Stakeholders: who needs this feature?
  - Priorities: what feature is more important?
- For the model
  - Constraints and feature dependencies

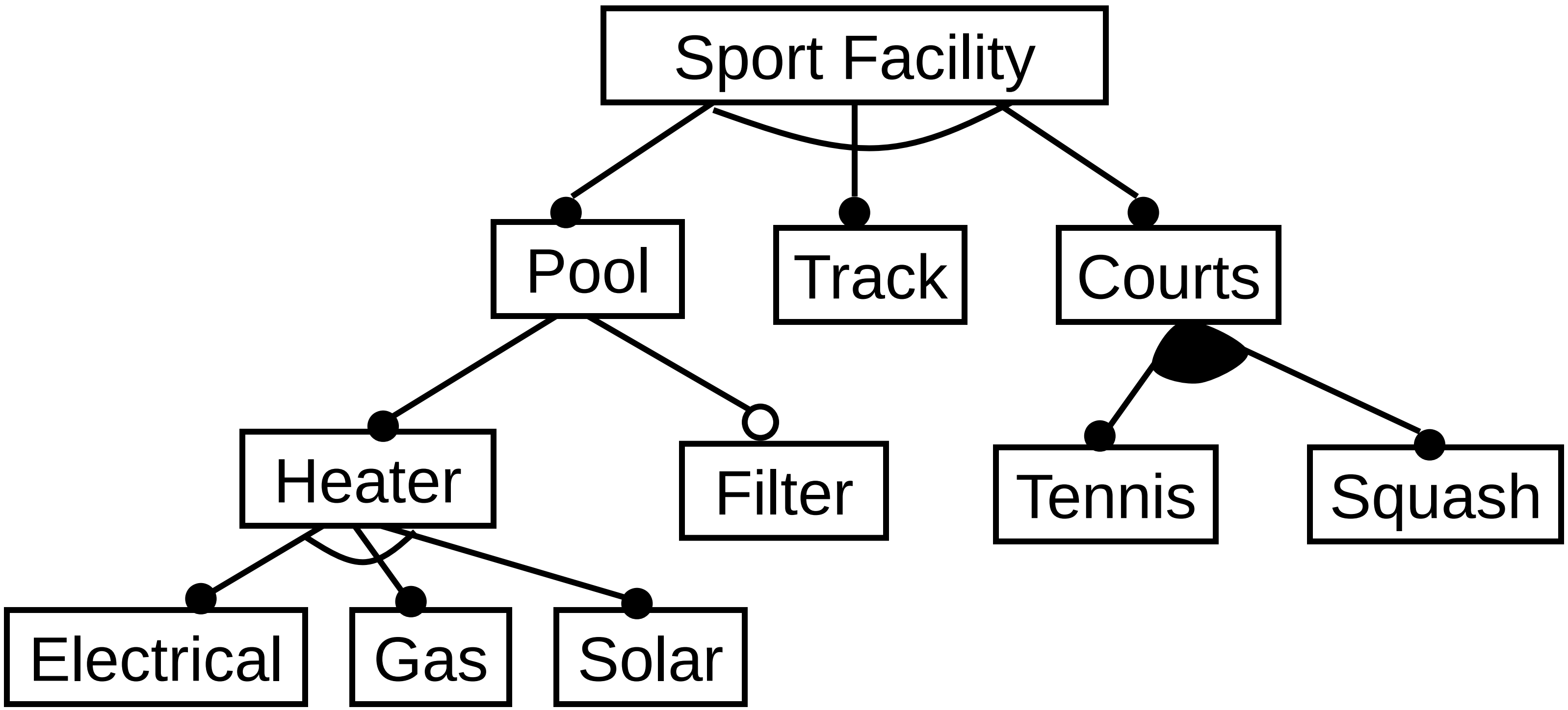
No standard notation – methodology dependent

Mutually Exclusive Features

- Two features are mutually exclusive if all program instances do not have both features simultaneously

- Two cases:
  - Both features are in the same set of exclusive-or features
  - Their ancestors (parent and their parents) are in the same set of exclusive-or features

Mutually Exclusive Features Example



Examples: Electrical and Gas Heater and Courts

Feature Model Analysis

Basic ideas

Propositional Logic

Basics

Propositional Logic (1)

- Definition
  - Formal system on which the formulas represent language propositions
- Two kinds of propositions
  - Atomic
    - p : today is raining
    - q : I bring my umbrella
  - Composed
    - p ^ q : today is raining AND I bring my umbrella
- Propositions can be assigned two values
  - true and false

Propositional Logic (2)

- Other operations to form complex propositional expressions
  - $p \lor q$: today is raining OR I bring my umbrella
  - $\neg p$: today is NOT raining
  - $p \Rightarrow q$: IF today is raining THEN I bring my umbrella
  - $p \Leftrightarrow q$: IF AND ONLY IF today is raining THEN I bring my umbrella
    - equivalent to $(p \Rightarrow q) \wedge (q \Rightarrow p)$

Truth Tables

p	$\neg$p
t	f
f	t

p	q	p $\wedge$ q
t	t	t
t	f	f
f	t	f
f	f	f

p	q	p v q
t	t	t
t	f	t
f	t	t
f	f	f

p	q	p $\Rightarrow$ q
t	t	t
t	f	f
f	t	t
f	f	t

p	q	p $\Leftrightarrow$ q
t	t	t
t	f	f
f	t	f
f	f	t

Some Equivalence Rules

- De Morgan's laws
  - $\neg(p \vee q) \equiv \neg p \wedge \neg q$
  - $\neg(p \wedge q) \equiv \neg p \vee \neg q$
- Distributivity
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
  - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
- Double negation
  - $\neg \neg p \equiv p$

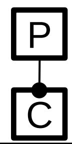
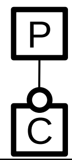
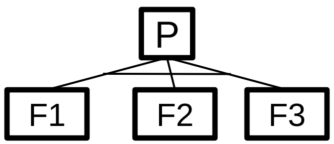
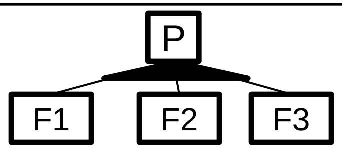
Feature Models and Propositional Logic

What is the connection?

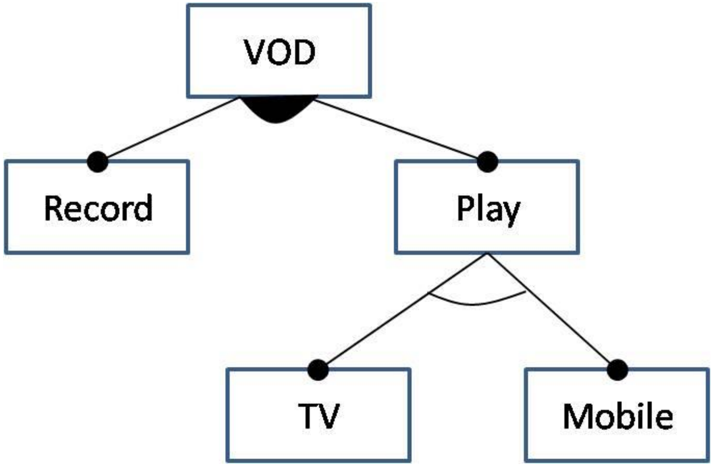
- Feature models are transformed to propositional logic
- Using SAT solver technology we can obtain information for analysis of properties of the feature models

feature model → propositional logic representation → SAT solver → properties analysis

Feature Models to Propositional Logic

Name	**Diagram Notation**	**Propositional Logic**
Mandatory		$P \Leftrightarrow C$
Optional		$C \Rightarrow P$
Exclusive-Or		$(F1 \Leftrightarrow (\neg F2 \land \neg F3 \land P)) \land$ $(F2 \Leftrightarrow (\neg F1 \land \neg F3 \land P)) \land$ $(F3 \Leftrightarrow (\neg F1 \land \neg F2 \land P))$
Inclusive-Or		$P \Leftrightarrow F1 \lor F2 \lor F3$
Requires	Cross feature arrow	$A \Rightarrow B$
Excludes	Cross feature arrow	$A \Rightarrow \neg B \equiv \neg(A \land B)$

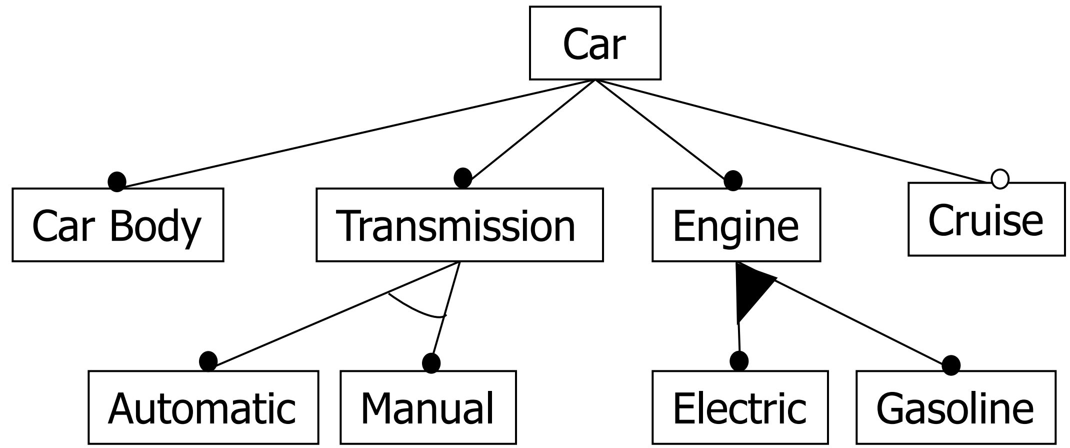
First Example



Product Line Formula PL_f

$$\mathrm{VOD} \Leftrightarrow \mathrm{true} \quad \wedge$$
$$\mathrm{VOD} \Leftrightarrow \mathrm{Record} \vee \mathrm{Play} \quad \wedge$$
$$\mathrm{TV} \Leftrightarrow \neg\mathrm{Mobile} \wedge \mathrm{Play} \quad \wedge$$
$$\mathrm{Mobile} \Leftrightarrow \neg\mathrm{TV} \wedge \mathrm{Play}$$

Second Example



$$PL_f \left\{ \begin{array}{l} \text{Car} \Leftrightarrow \text{true} \quad \wedge \\ \text{Car} \Leftrightarrow \text{CB} \wedge \text{Car} \Leftrightarrow \text{Tr} \wedge \text{Car} \Leftrightarrow \text{Eng} \wedge \text{Cr} \Rightarrow \text{Car} \quad \wedge \\ \text{Auto} \Leftrightarrow \neg\text{Man} \wedge \text{Tr} \quad \wedge \\ \text{Man} \Leftrightarrow \neg\text{Auto} \wedge \text{Tr} \quad \wedge \\ \text{Eng} \Leftrightarrow (\text{Ele} \vee \text{Gas}) \end{array} \right.$$

Satisfiability Problem

- There exists different algorithms and tools that manipulate and analyze propositional formulas
- ***Satisfiability***
  - Is there any combination of true and false values that makes a propositional formula evaluate to true?
- Complex problem
  - NP – non-polynomial time

SAT Solvers

- SAT solvers
  - Are one general approach to address the satisfiability problem
  - Utilize heuristics and other tricks to improve efficiency

- Examples of SAT solvers
  - sat4j
  - picoSAT

Using SAT solvers – Input (1)

- Use a standard format called DIMACS

- Based on the Conjunctive Normal Form (CNF)
  - A cannonical form to which any propositional logic expression can be mapped
  - It consists on a set of clauses, of the form

    $(X_1 \vee X_2 \,..\, \vee X_n)$

    where Xi is a boolean variable, which can also be negated ⌐Xi
  - All the clauses are and-ed

Using SAT solvers – Input (2)

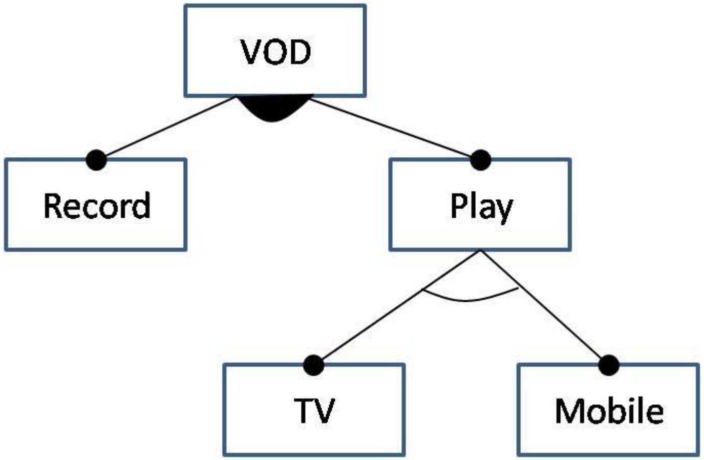
- Example of CNF, transform $p \Leftrightarrow q$

  $p \Leftrightarrow q$ → equivalence

  $(p \Rightarrow q) \wedge (q \Rightarrow p)$ → $(c \Rightarrow d) \equiv (\neg c \vee d)$

  $(\neg p \vee q) \wedge (\neg q \vee p)$

Example – Translation Feature Model to CNF



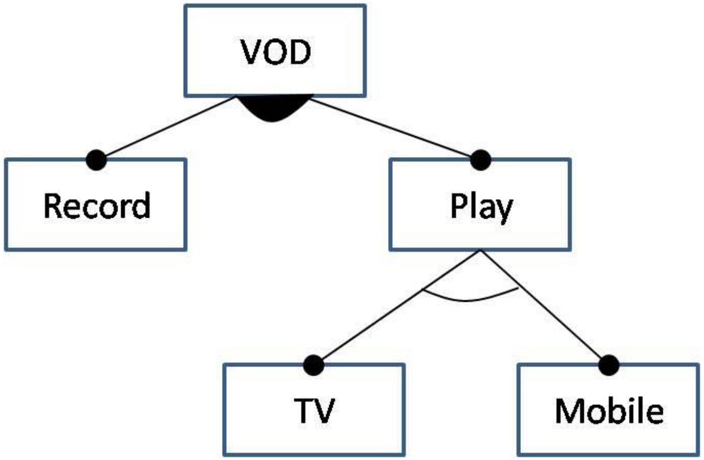
Product Line Formula PL_f

(1) VOD ⇔ true ∧
(2) VOD ⇔ Record ∨ Play ∧
(3) TV ⇔ ¬Mobile ∧ Play ∧
(4) Mobile ⇔ ¬TV ∧ Play

(1) VOD
(2) (¬VOD ∨ Record ∨ Play) ∧ (VOD ∨ ¬Record) ∧ (VOD ∨ ¬Play)
(3) (¬TV ∨ ¬Mobile) ∧ (¬TV ∨ Play) ∧ (TV ∨ Mobile ∨ ¬Play)
(4) (¬Mobile ∨ ¬TV) ∧ (¬Mobile ∨ Play) ∧ (Mobile ∨ TV ∨ ¬Play)

DIMACS Format (1)

- First step, mapping between feature names and integer numbers
  - Mapping is arbitrary but must be consistently used for all the terms
- Example



VOD – 1
Record – 2
Play – 3
TV – 4
Mobile – 5

DIMACS Format (2)

Second step, substitute the feature names by their integer numbers in the CNF expression

VOD – 1
Record – 2
Play – 3
TV – 4
Mobile – 5

```
(1)  VOD
(2)  (¬VOD ∨ Record ∨ Play) ∧ (VOD ∨ ¬Record) ∧ (VOD ∨ ¬Play)
(3)  (¬TV ∨ ¬Mobile) ∧ (¬TV ∨ Play) ∧ (TV ∨ Mobile ∨ ¬Play)
(4)  (¬Mobile ∨ ¬TV) ∧ (¬Mobile ∨ Play) ∧ (Mobile ∨ TV ∨ ¬Play)
```

Mapped to numbers

```
(1)  1
(2)  (¬1 ∨ 2 ∨ 3) ∧ (1 ∨ ¬2) ∧ (1 ∨ ¬3)
(3)  (¬4 ∨ ¬5) ∧ (¬4 ∨ 3) ∧ (4 ∨ 5 ∨ ¬3)
(4)  (¬5∨ ¬4) ∧ (¬5 ∨ 3) ∧ (5 ∨ 4 ∨ ¬3)
```

DIMACS Format (3)

JOHANNES KEPLER UNIVERSITY LINZ

- Third step, each CFN clause is stored in a row
- Symbol And is ommited $\wedge$

(1) 1
(2) $(\neg 1 \vee 2 \vee 3) \wedge (1 \vee \neg 2) \wedge (1 \vee \neg 3)$
(3) $(\neg 4 \vee \neg 5) \wedge (\neg 4 \vee 3) \wedge (4 \vee 5 \vee \neg 3)$
(4) $(\neg 5 \vee \neg 4) \wedge (\neg 5 \vee 3) \wedge (5 \vee 4 \vee \neg 3)$

1 (1)
$\neg 1 \vee 2 \vee 3$
$1 \vee \neg 2$ (2)
$1 \vee \neg 3$
$\neg 4 \vee \neg 5$
$\neg 4 \vee 3$ (3)
$4 \vee 5 \vee \neg 3$
$\neg 5 \vee \neg 4$
$\neg 5 \vee 3$ (4)
$5 \vee 4 \vee \neg 3$

DIMACS Format (4)

- Fourth step, negation signs (¬) are translated as negative signs (-)

1		1
¬1 ∨ 2 ∨ 3		-1 ∨ 2 ∨ 3
1 ∨ ¬2		1 ∨ -2
1 ∨ ¬3	⇒	1 ∨ -3
¬4 ∨ ¬5		-4 ∨ -5
¬4 ∨ 3		-4 ∨ 3
4 ∨ 5 ∨ ¬3		4 ∨ 5 ∨ -3
¬5∨ ¬4		-5 ∨ -4
¬5 ∨ 3		-5 ∨ 3
5 ∨ 4 ∨ ¬3		5 ∨ 4 ∨ -3

DIMACS Format (5)

- Fifth step, disjunction signs are removed

```
1
-1 ∨ 2 ∨ 3
1 ∨ -2
1 ∨ -3
-4 ∨ -5
-4 ∨ 3
4 ∨ 5 ∨ -3
-5 ∨ -4
-5 ∨ 3
5 ∨ 4 ∨ -3
```

```
1
-1  2  3
1  -2
1  -3
-4  -5
-4  3
4  5  -3
-5  -4
-5  3
5  4  -3
```

DIMACS Format (6)

- Sixth step, DIMACS headers are added and clauses ended with 0

```
c coments              comment
p cnf 5 10             Indicates CNF format
1 0
-1  2  3 0
1  -2 0
1  -3  0
-4  -5 0
-4  3 0
4  5  -3 0
-5  -4 0
-5  3 0
5  4  -3 0
```

number of CNF clauses

number of variables

Using SAT solvers – Ouput (1)

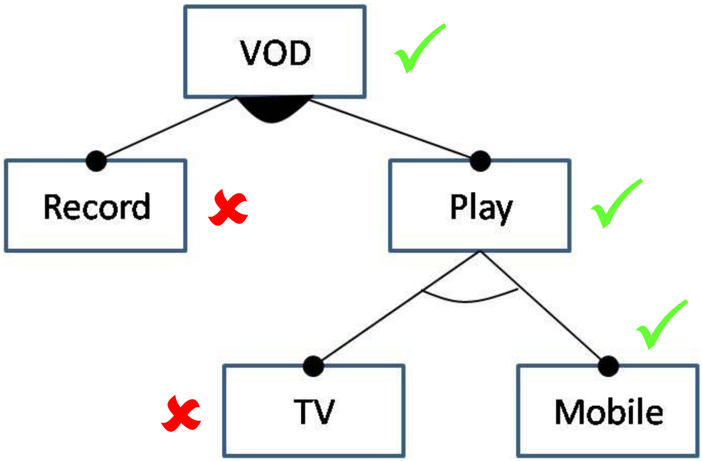
- Basic functionality is finding out if proposition expression is satisfiable or not
  - If yes, returns a configuration Example of CNF, transform p $\Leftrightarrow$ q
- Depending on the SAT solver it may also compute
  - All possible solutions that make the expression satisfiable
  - If not satisfiable, it could provide information on why not

Picosat example

- Tool developed at JKU by Armin Biere group
- Output produced
  - $ ./picosat sple-example.dimacs

    s SATISFIABLE

    v 1 -2 3 -4 5 0



- Interpretation of result
  - VOD selected
  - Record not selected
  - Play selected
  - TV not selected
  - Mobile selected

VOD – 1
Record – 2
Play – 3
TV – 4
Mobile – 5

Catalogue of Feature Model Operations (1)

- Void feature model
  - Checks if there is at least one valid configuration denoted by the feature model
- Valid product
  - Checks if a product configuration is a valid feature combination in the feature model
- Valid partial configuration
  - Check if there is at least one product that satifies the incomplete product definition
- All products
  - Lists all the products denoted by a feature model

Catalogue of Feature Model Operations (2)

- Number of products
  - Counts how many products are denoted by the feature model
- Filter
  - Returns a list of products that satisfy a partial configuration
- Dead feature
  - Check if there is at least one product where a feature is selected
- False optional feature
  - Checks if for an optional feature there are products that have that feature as unselected.
- Core features
  - The list of features that are selected in ALL the products denoted by a feature model

Catalogue of Feature Model Operations (3)

- Variant features
  - The list of features that are selected in some products but not in all
    - Counterpart to core features operation
- Atomic sets
  - A set of features that are always simultaneously selected and unselected in all the products denoted by the feature model

Readings

- Generative Programming. Czarnecki and Eisenecker. Addison-Wesley 2000. Chapter 4.
- K. Kang, et al. Feature-Oriented Domain Analysis (FODA) Feasibility Study. CMU/SEI-90-TR-21, Carnegie Mellon Univ., Pittsburgh, PA, Nov. (1990)
- David Benavides, Sergio Segura, Pablo Trinidad, Antonio Ruíz-Cortés. FAMA: Tooling a Framework for the Automated Analysis of Feature Models. Vamos Workshop, Lero, January 2007.
- M. Antkiewicz, K. Czarnecki. FeaturePlugin: Feature Modeling Plug-in for Eclipse. Workshop Eclipse '04, OOPSLA, 2004.
- David Benavides, Sergio Segura, Antonio Ruíz-Cortés. Automated analysis of feature models 20 years later: A literature review. Journal of Information Systems 35(2010) 615-636.

Weeks AHEAD

- Easter break
  - 23.3 and 30.3.
- Feature-Oriented Software Development (RL)
  - Lecture + take home exercise 2.1

Questions?

- Now or later to rick.rabiser@jku.at | roberto.lopez@jku.at