

Modeling and managing variability of software intensive systems

Variability Models (Part 2)

Mathieu Acher

Maître de Conférences

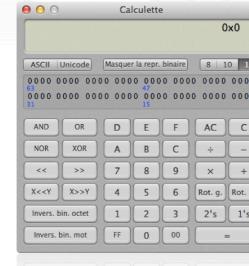
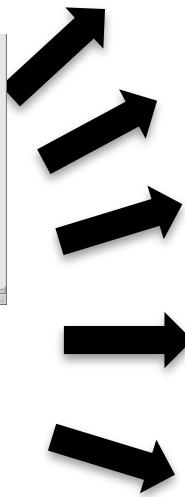
mathieu.acher@irisa.fr

Previously



```
Linux Kernel Configuration
```

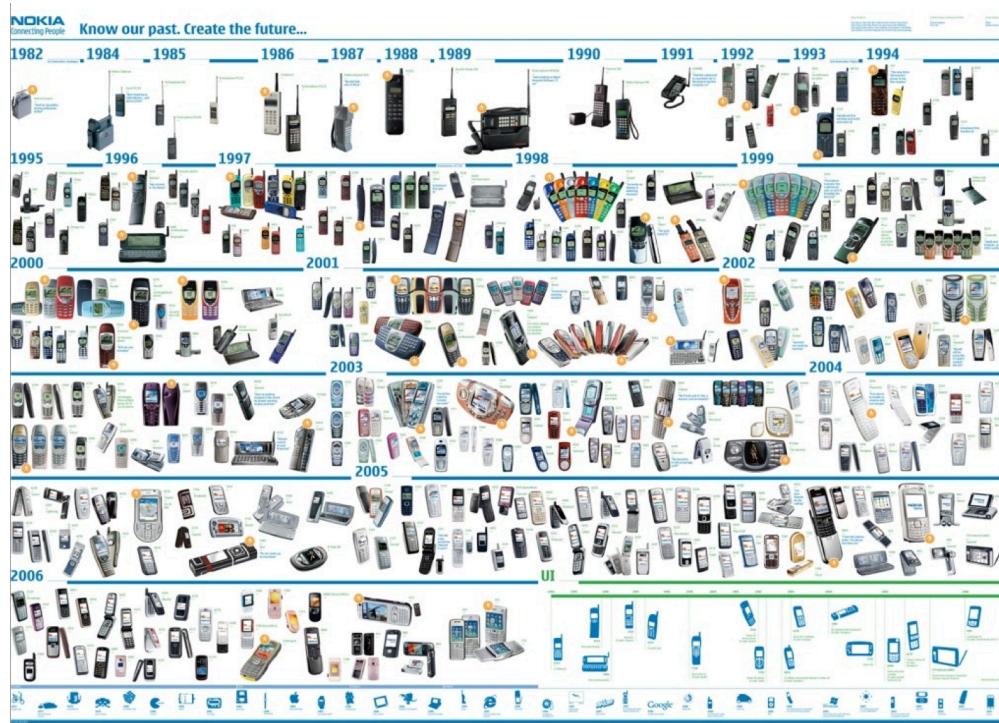
Linux Kernel Configuration window showing various kernel options.



Linux Kernel



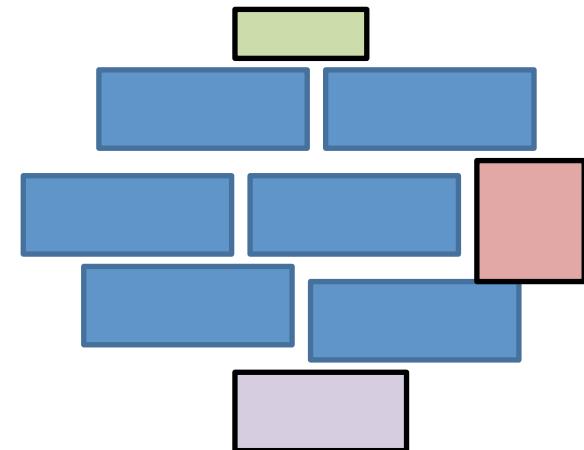
2000 features



Software Product Line Engineering

Factoring out **commonalities**

for **Reuse** [Krueger et al., 1992] [Jacobson et al., 1997]



Managing **variabilities**

for Software **Mass Customization** [Bass et al., 1998] [Krueger et al., 2001], [Pohl et al., 2005]



Variability Management

Common features

print – connect with computer...

Variable features

fax – scan – USB port...

Product-specific features

serial port



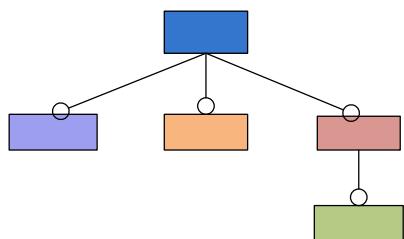
Variability

“the ability of a system to be efficiently extended, changed, customized or configured for use in a particular context”

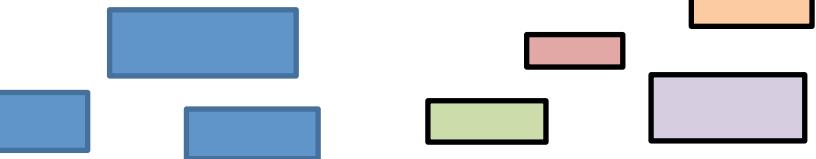
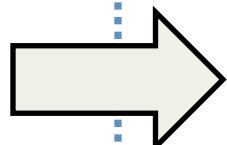
Mikael Svahnberg, Jilles van Gurp, and Jan Bosch (2005)



Domain engineering (development for reuse)



Variability Model

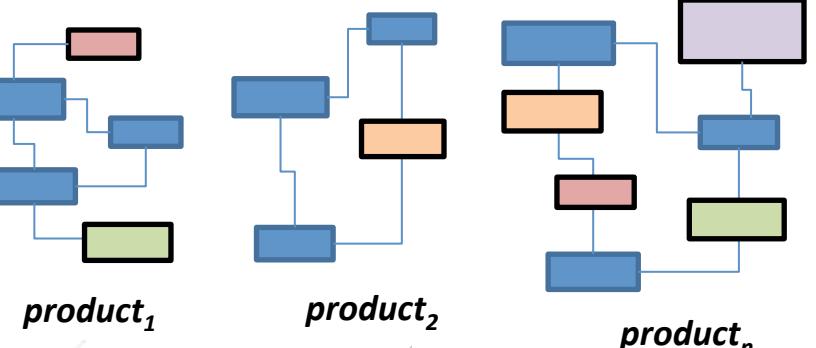
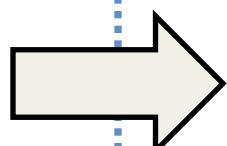
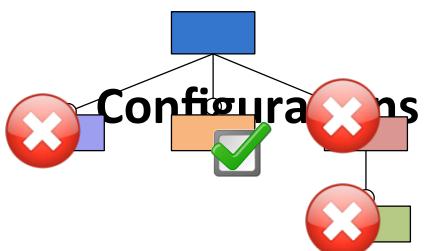


Common assets

Variants

Reusable Assets

(e.g., models or source code)

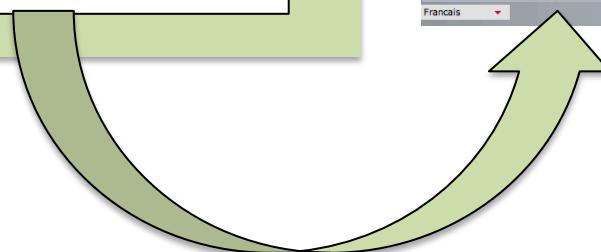


Application engineering (development with reuse)

Modeling Variability: Why Should I Care?

Variability Models

- ▼ CarEquipment
 - ▼ ● Healthing
 - ▼ A
 - AirConditioningFrontAndRear
 - AirConditioning
 - ▼ ● Comfort
 - AutomaticHeadLights
 - ▼ ● DrivingAndSafety
 - FrontFogLights
- ▼ Constraints
 - AutomaticHeadLights ⇒ FrontFogLights



The screenshot shows a car configuration interface for an Audi R8 Spyder. At the top, there are three car models: a blue hatchback, a grey sedan, and a white hatchback. Below them is a view of the interior of the white car. To the right, a price summary table is displayed:

Modèle	Prix total
Audi R8 Spyder 5.2 FSI quattro R tronic	185.899,35 EUR
Prix de base	170.490,00 EUR
Équipements optionnels	15.409,35 EUR

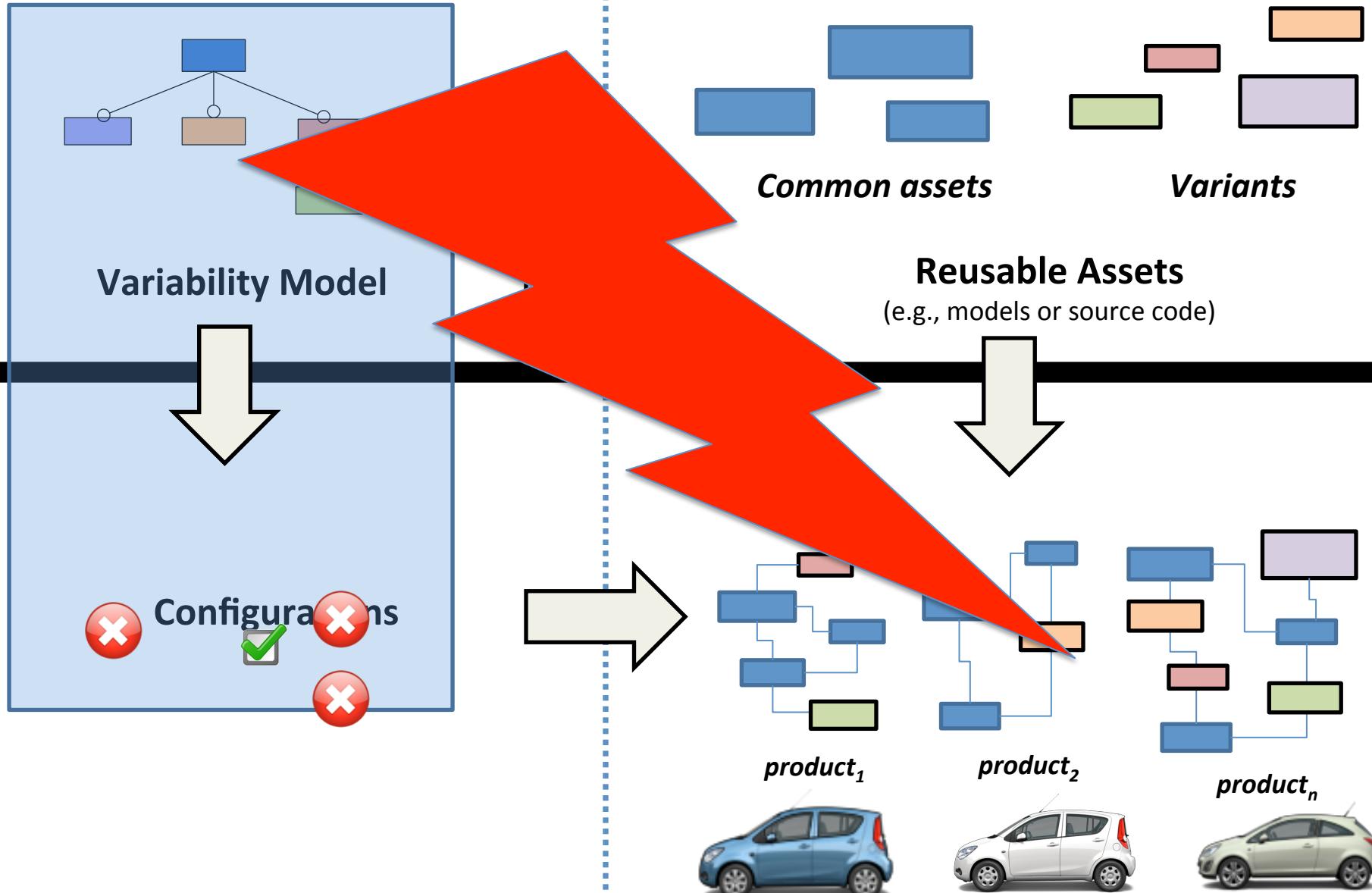
Below the table are several checkboxes for optional equipment, such as "Régulateur de vitesse" (checked), "Système d'aide au stationnement APS avant / arrière", and "Système d'aide au stationnement Advanced : APS avant et arrière et caméra arrière".

At the bottom, a Java code editor shows the logic for the constraint "AutomaticHeadLights ⇒ FrontFogLights". The code uses a switch statement based on the selected model and motor type to determine if fog lights should be selected.

```
switch (model) {  
    case "Audi R8 Spyder":  
        if (motor == "5.2 FSI quattro R tronic") {  
            if (headlights.isSelected()) {  
                fogLights.setSelected(true);  
            }  
        }  
        break;  
}
```

#1 precise specification is needed

Domain engineering (development for reuse)



Application engineering (development with reuse)

A police car is engulfed in intense orange and yellow flames, with thick black smoke billowing upwards. The car is a white sedan with "766" on the side and "POLICE" and "To Serve & Protect" on the front. It is parked on a city street at night. In the background, there are other buildings, including one with "GUESS" and "KIRK'S" signs, and another with "DRUM SHOP" and "JAS STORE". Several people are standing around the burning vehicle, some appearing to take pictures or videos. The scene is chaotic and suggests a protest or riot.

DRUM SHOP

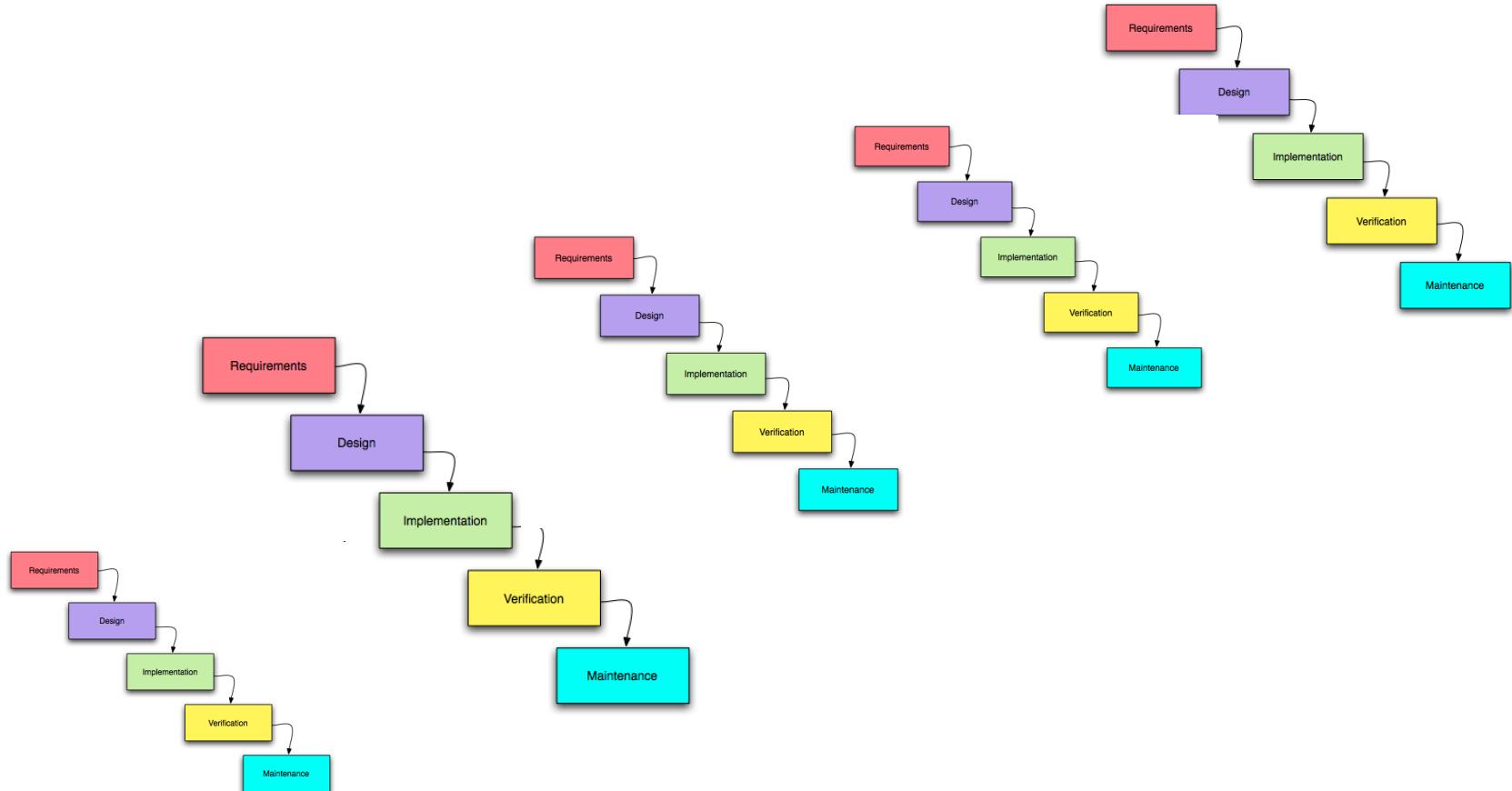
JAS
STORE

Illegal Variant

A photograph of an old, green-painted pickup truck that has been left to decay in a field. The truck is heavily rusted, particularly on the body and the front fenders. The driver's side door is open, revealing the interior frame and some remaining mechanical components. The truck is positioned in front of a dense thicket of green bushes and tall grass. In the bottom left corner, there are some wooden planks and metal debris, suggesting a construction or demolition site nearby.

Unused flexibility

**#2 variability cross cuts all
artefacts (from requirements to
code)**



httpd.conf -- win32 Apache

Building a Web Server, for Windows

Listen 80

ServerRoot "/www/Apache2"

DocumentRoot "/www/webroot"

ServerName localhost:80

ServerAdmin admin@localhost

ServerSignature On

ServerTokens Full

DefaultType text/plain

AddDefaultCharset ISO-8859-1

UseCanonicalName Off

HostnameLookups Off

Error Log	Identifier	License	Language	Storage	LicenseCostFee	RSS	Unicode
PBwiki	Confluence	Commercial	Java	Database	US10	Yes	Yes
MoinMoin	Nolimit		No	No	Yes	Yes	No
DokuWiki	GPL	Python	Files		No	Yes	Yes
PmWiki	GPL2	PHP	Files		No	Yes	Yes
DrupalWiki	GPL2	PHP	Database	Different Licences	Yes	Yes	
TWiki	GPL	Perl	FilesRCS	Community	Yes	Yes	
MediaWiki	GPL	PHP	Database	No	Yes	Yes	

KeepAlive On

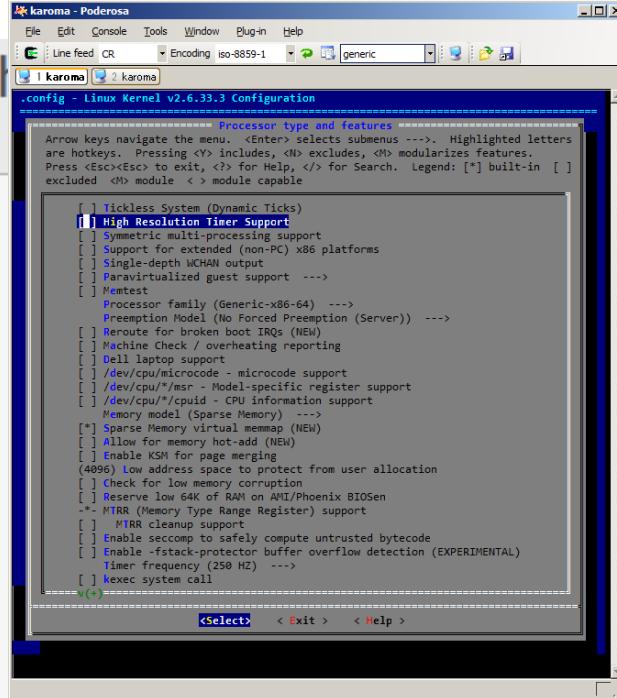
MaxKeepAliveRequests 100

KeepAliveTimeout 15

```
<IfModule mpm_winnt.c>
```

ThreadsPerChild 250

MaxRequestsPerChild 0



RENAULT VANS

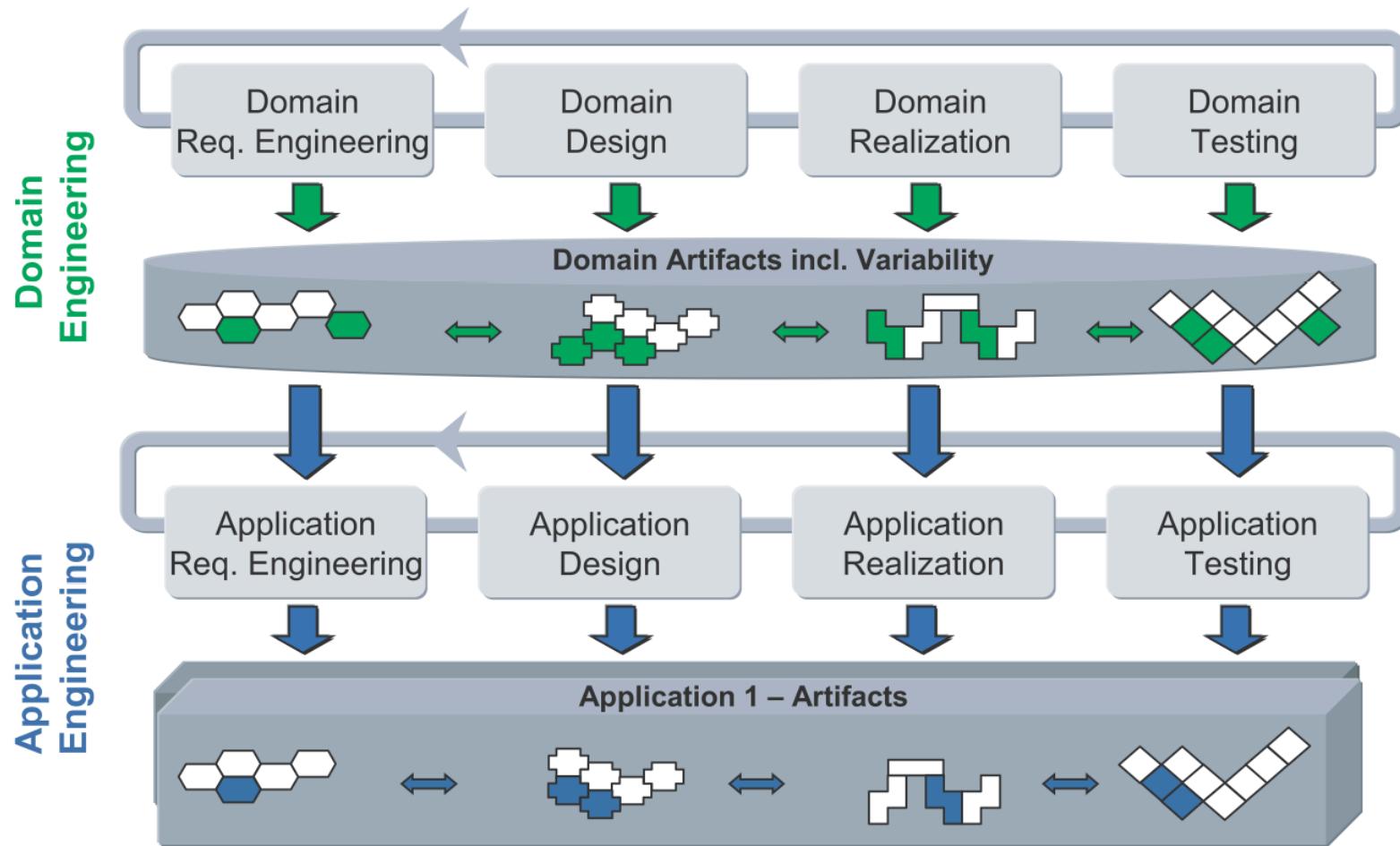
CARS | VANS | ELECTRIC VEHICLES | RENAULT BUSINESS | USED CARS | OWNER SERVICES | ABOUT RENAULT | RENAULT SHOP
Renault UK > Renault Vans > New Kangoo Van Range > Kangoo Van > Build your own Kangoo Van > Selected Options

NEW KANGOO VAN RANGE

A screenshot of the Renault Vans 'Build your own Kangoo Van' configuration page. It shows three tabs: '01 Preferences', '02 Version', and '03 Equipment & options'. Under 'OPTIONS', there are sections for 'COMFORT', 'DRIVING', and 'SAFETY & SECURITY' with various checkboxes and price tags. To the right is an image of a white Renault Kangoo van.

A screenshot of the Eclipse IDE. The top part shows the 'Preferences' dialog for 'General' settings, with checkboxes for 'Always run in background' and 'Show heap status' checked. Below it, the 'Open mode' section has 'Double click' selected. The bottom part shows the Java code editor with syntax highlighting and annotations, and the 'ASTView' perspective is active on the right.

Software Product-Line Engineering



#3 automation (exponential number of products)

Variability = Complexity

33 features



a unique variant for every
person on this planet

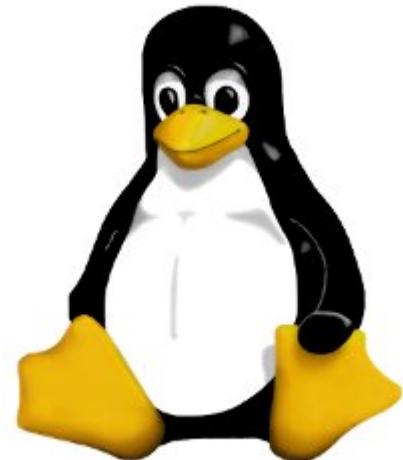
320^{optional, independent}
features

more variants than estimated
atoms in the universe



2000 features

10000
features



Variability Model

- **Abstraction** of some aspect of a system under study (SUS)
 - Some details are hidden or removed to **simplify** and focus attention
 - **general** concepts can be formulated by abstracting common properties of instances
- In a form that can be **mechanically analyzed**
 - properties of a SUS
 - model transformation
 - Generation of another abstraction (including code)

Variability Models: Why?

(3 arguments, summary)

precise specification

variability is orthogonal / cross cuts different
artefacts

automation

Variability models are needed. Now we are convinced...

#1 Formalism? Theory?
modeling principles, semantics, logics

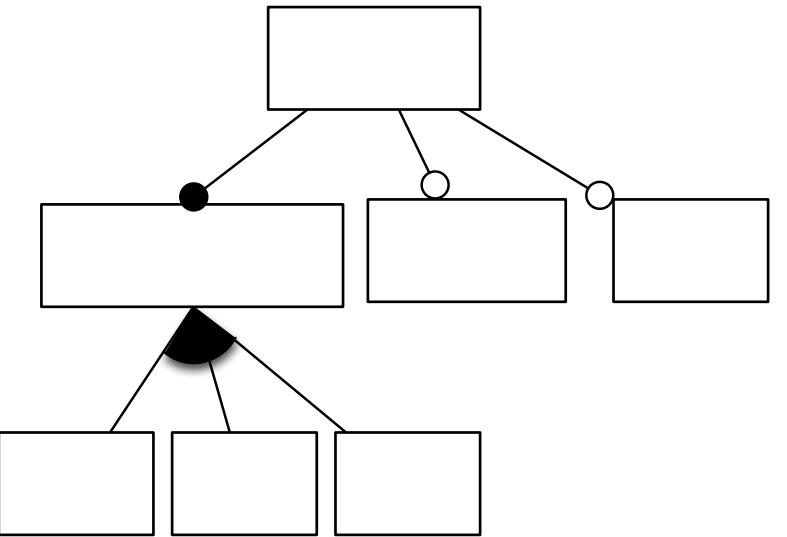
#2 Language?
to elaborate/build variability models

#3 Reasoning techniques?
to analyze properties of an SPL (scalable and automated way)

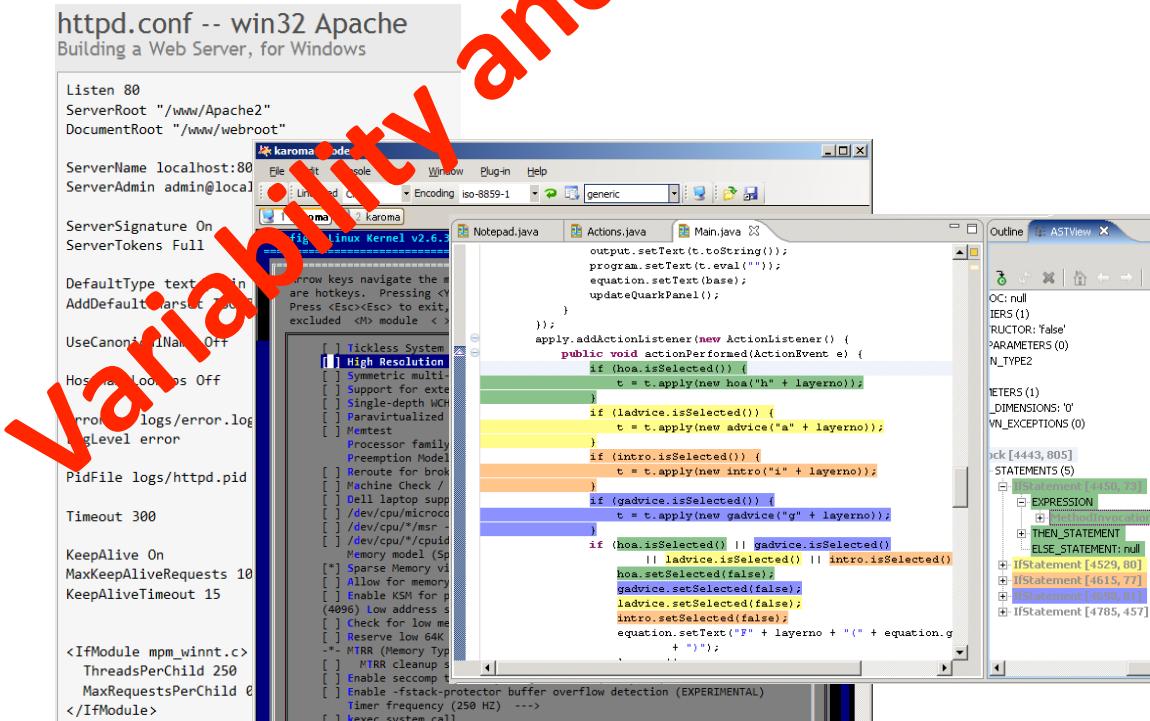
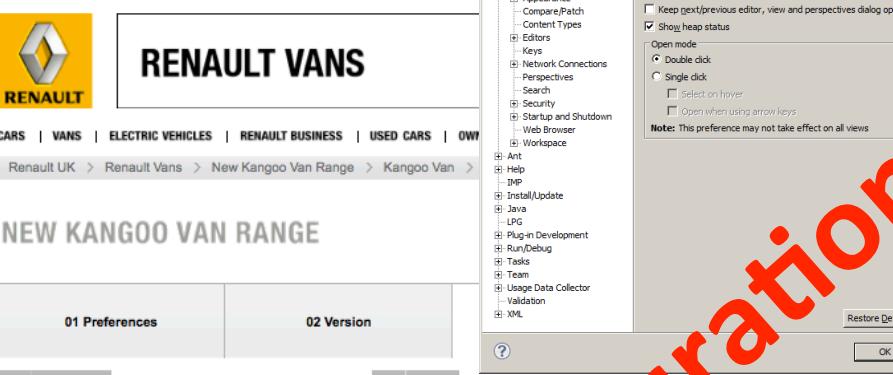
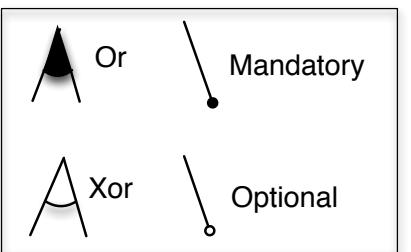
#4 Applications of variability models?
variability model and so what?

Feature models,
the defacto standard
for modeling variability

Feature Models



not, and, or, implies



Feature Model: *defacto* standard

- Central to many software product line approaches
 - More than 1000 citations of [Kang et al., 1990] per year
 - Generative programming [Czarnecki et al., 2000]
 - Research effort
 - Formal Semantics [Schobbens et al., 2007]
 - Automated Reasoning Techniques [Batory et al., 2005], [Czarnecki et al., 2007], [Mendonca et al., 2009], [Thuem et al., 2009], [Janota et al., 2010], [Benavides et al., 2010]
 - Tools
 - Commercial: pure::variants, Gears
 - Academic: fmp, FeatureIDE, FaMa, FAMILIAR, SPLOT, TVL, etc.
- 
- 
- 

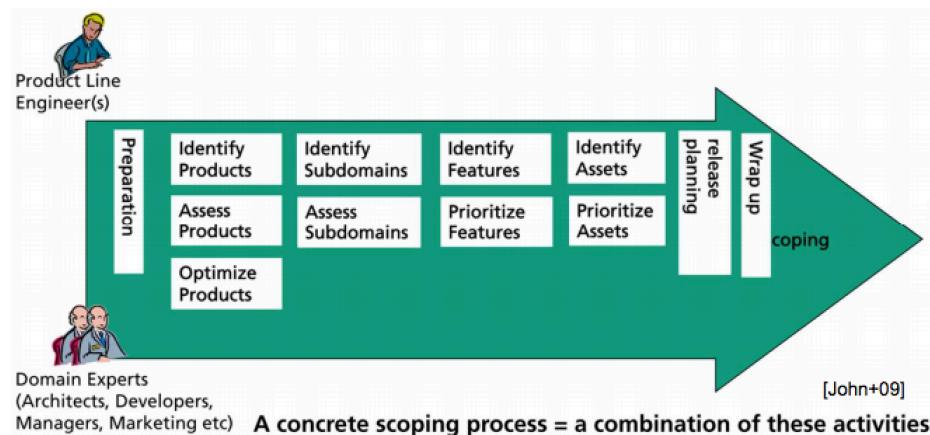
Feature Model: Applications

- Domain Analysis
- Configurators
- Automated Derivation of “Product”
- Testing
- Model Checking

Feature Models: Principles

Feature modeling is...

- ... about identifying
 - common features of concepts
 - variable features of concepts
 - and their dependencies
 - and documenting them in a coherent model, called a feature model
- Core activity of domain-engineering methods



Feature and Psychology

- The human mind processes lots of information
- Important function of human information processing: reducing the amount of information
- Describing concepts by **features**
 - Clustering
 - Abstraction (Classification/Generalization)
 - Vocabulary construction
- Classifying a perceptual unit as an instance of a concept
- Analysis of similarity
 - Features **common** to entities, activities, events, relationships, structures, etc.
- Analysis of variations
 - Features that are **unique** to entities, activities, events, relationships, structures, etc.

Modeling variability (and commonality)

- **Features** are a convenient way to think about commonalities and differences between family members
 - e.g. "MP3 player" – some phones have it, some don't
- Features are « usually » **coarse-grained abstractions**
 - Abstract from detailed requirements
 - e.g. "when in shuffle mode, the player selects the next track using a random function"
 - Abstract from design/implementation details
 - e.g. "MP3_Player is a concrete subclass of Media_Player »
- Depends on what you are modeling and for which purpose
 - Eliciting requirements with customers (informal document)
 - VS building a representation of your code (based on CPP directives)

What's a **feature**? (survey excerpt)

- [Kang et al.] “a prominent or distinctive **user-visible aspect, quality or characteristic** of a software system or systems”
- [Kang et al.] “distinctively identifiable **functional abstractions** that must be implemented, tested, delivered, and maintained”
- [Eisenecker and Czarnecki]. “**anything users** or client programs might want to **control about a concept**”
- [Bosch et al.] “A **logical unit of behaviour** specified by a set of **functional and non-functional requirements**.”
- [Chen et al.] “a **product characteristic** from user or customer views, which essentially consists of a cohesive **set of individual requirements**”
- [Batory] “an elaboration or augmentation of an entity(s) that introduces a **new service, capability** or relationship”

Feature Models: Syntax



R8 Spyder

5.2 FSI quattro R tronic

Prix total

171.216,00 EUR

Prix de base

170.490,00 EUR

Equipements optionnels

726,00 EUR

- ▶ Informations détaillées
- ▶ Entrez l'Audi Code
- ▶ Générer un PDF
- ▶ Nouvelle configuration



[+] Plein écran / Dimensions

▶ Fermer la capote

Habitacle

Tableau de bord

Packs

Aucun pack n'est proposé pour ce modèle.

Couleurs

Blanc Ibis

Noir

Prix: 0,00 EUR



Couleurs métallisées à partir de 0,00 EUR



Couleurs à effet perlé à partir de 0,00 EUR



Couleurs personnalisées Audi exclusive



Couleur capote

Noir



Jantes

4 Jantes alu 5 BRANCHES ROTOR finition titane 8,5 x 19 à l'avant, 11 x 19 à l'arrière. Pneus 235/35 R19 à l'avant et 305 /30 R19 à l'arrière

Prix: 726,00 EUR

19" à partir de 0,00 EUR





R8 Spyder 5.2 FSI quattro R tronic

Prix total

185.899,35 EUR

Prix de base

170.490,00 EUR

Equipements optionnels **15.409,35 EUR**

- ▶ Informations détaillées
- ▶ Entrez l'Audi Code
- ▶ Générer un PDF
- ▶ Nouvelle configuration

[+] Plein écran / Dimensions Vue extérieure Tableau de bord

- ▶ Packs d'équipements
- ▶ Extérieur
- ▶ Jantes & pneumatiques
- ▶ Intérieur
- ▶ Volants
- ▶ Sièges
- Sécurité & technique**
- ▶ Infotainment

- ▶ Châssis
- ▶ Freins
- Systèmes d'assistance**
- ▶ Autres

excludes



<input checked="" type="checkbox"/> Régulateur de vitesse		320,65 EUR
<input type="checkbox"/> Système d'aide au stationnement APS avant / arrière		931,70 EUR
<input type="checkbox"/> Système d'aide au stationnement APS avant / arrière avec affichage dans l'écran MMI		1.373,35 EUR
<input checked="" type="checkbox"/> Système d'aide au stationnement Advanced : APS avant et arrière et caméra arrière		1.790,80 EUR
<input checked="" type="radio"/> Audi hill assist : assistance au démarrage en côte		Série
<input type="checkbox"/> Réinitialiser la sélection		

Attention:

Le prix peut varier en fonction du choix de moteur et des équipements.

Un aperç des équipements:

Mode expert



A5 Sportback 3.0 TDI quattro S tronic

Prix total

54.460,15 EUR

Prix de base

50.570,00 EUR

Equipements optionnels

3.890,15 EUR

▶ Informations détaillées

▶ Entrez l'Audi Code

▶ Nouvelle configuration

Vérification de votre sélection

Cet équipement nécessite un équipement complémentaire:

GPS Plus avec disque dur



2.934,25 EUR

Voici les équipements complémentaires possibles:

Ordinateur de bord en couleur avec programme efficiency



181,50 EUR

Remarque: uniquement sur les modèles avec système Start-Stop et uniquement disponible en combinaison avec l'autoradio Concert, l'autoradio Symphony ou un système de navigation

Pack Intenso Plus



3.100,00 EUR

Sans appareil de navigation

Série

[+] Plein écran / Dimensions

- ▶ Packs d'équipements
- ▶ Extérieur
- ▶ Jantes & pneumatiques
- ▶ Intérieur
- ▶ Volants
- ▶ Sièges
- ▶ Sécurité & technique

Infotainment

Attention:

Le prix peut varier en fonction du choix de moteur et des équipements.

Un aperç des équipements:

Mode expert

Réinitialiser la sélection

1 Modèle

2 Moteur

3 Extérieur

4 Intérieur

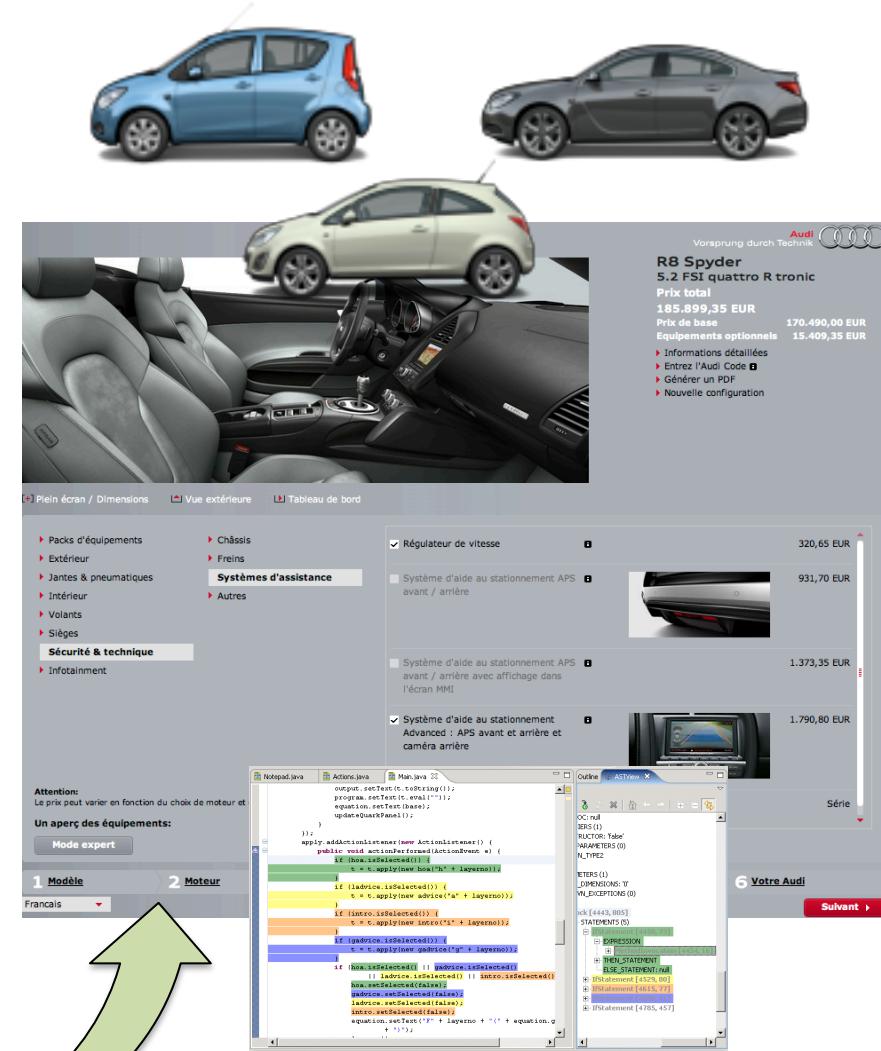
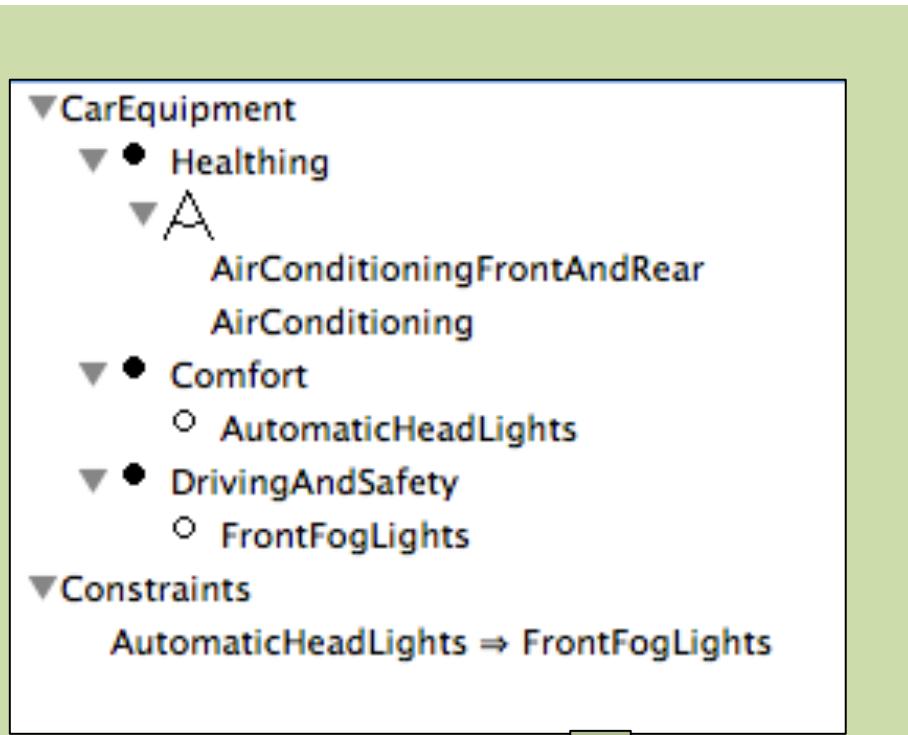
5 Option

6 Votre Audi

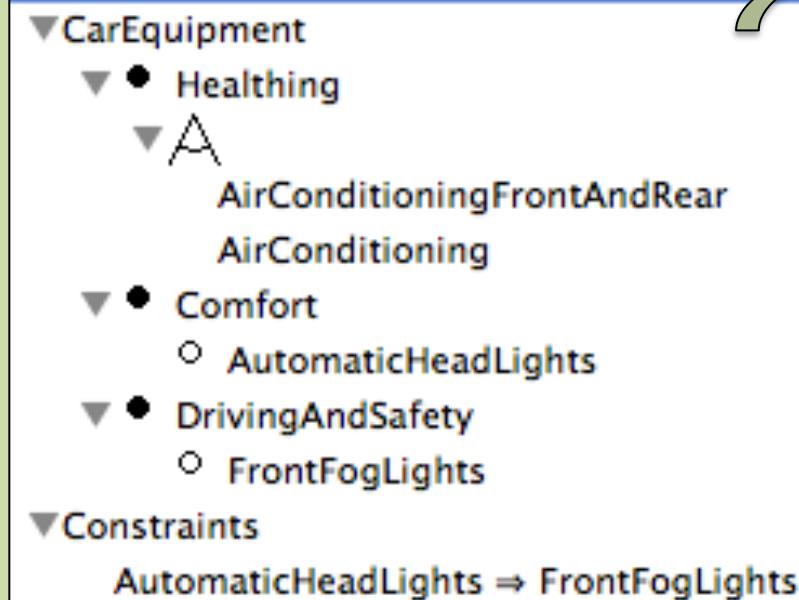
Français ▾

Suivant ▶

Feature Models



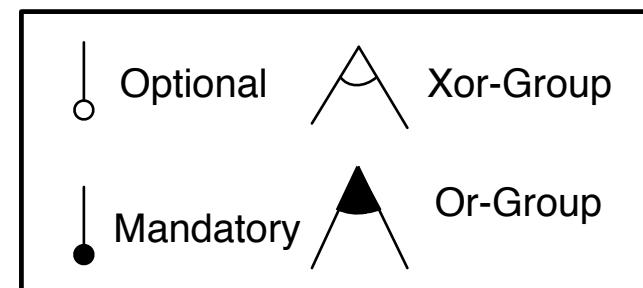
Feature Models (Background)

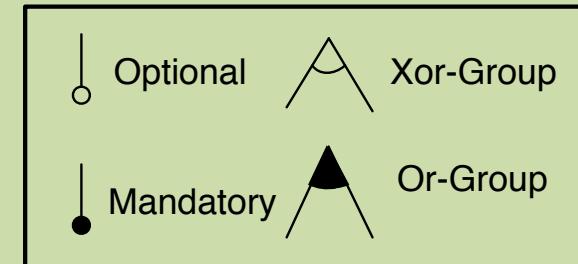
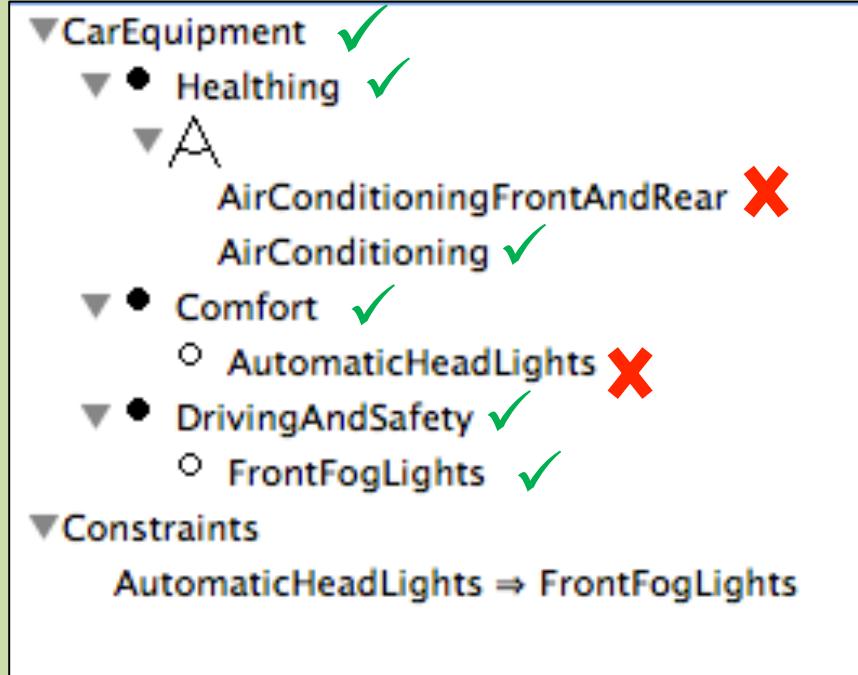


Hierarchy: rooted tree

Variability:

- mandatory,
- optional,
- Groups: exclusive or inclusive features
- Cross-tree constraints





Hierarchy + Variability

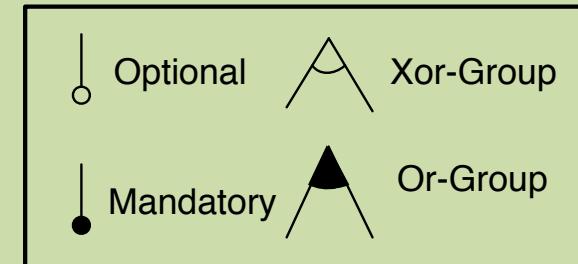
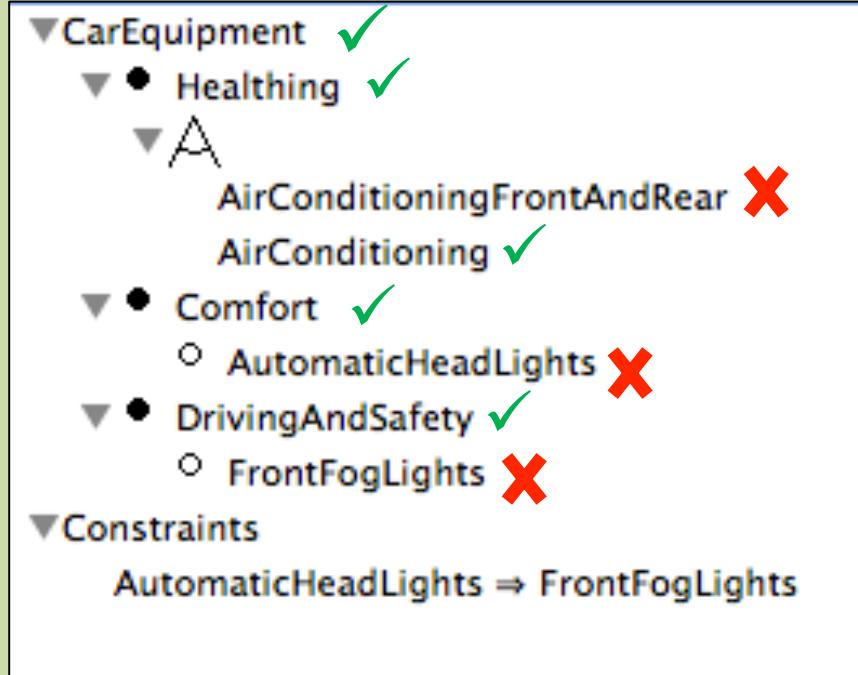
=

set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning, FrontFogLights}





Hierarchy + Variability

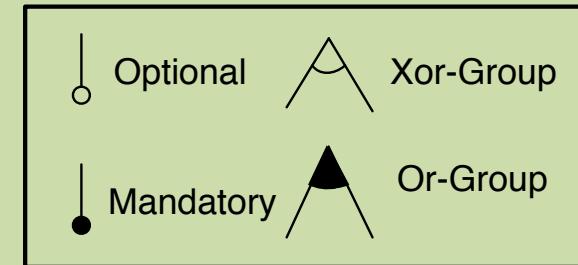
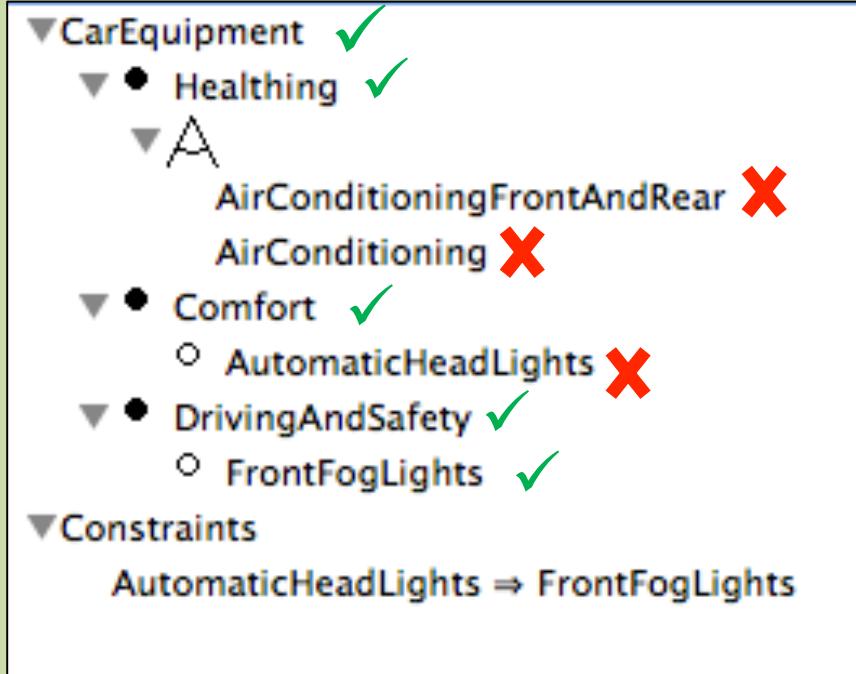
=

set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}





Hierarchy + Variability

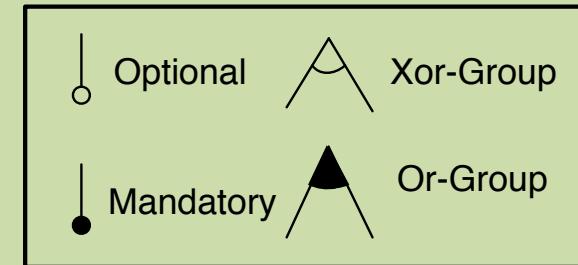
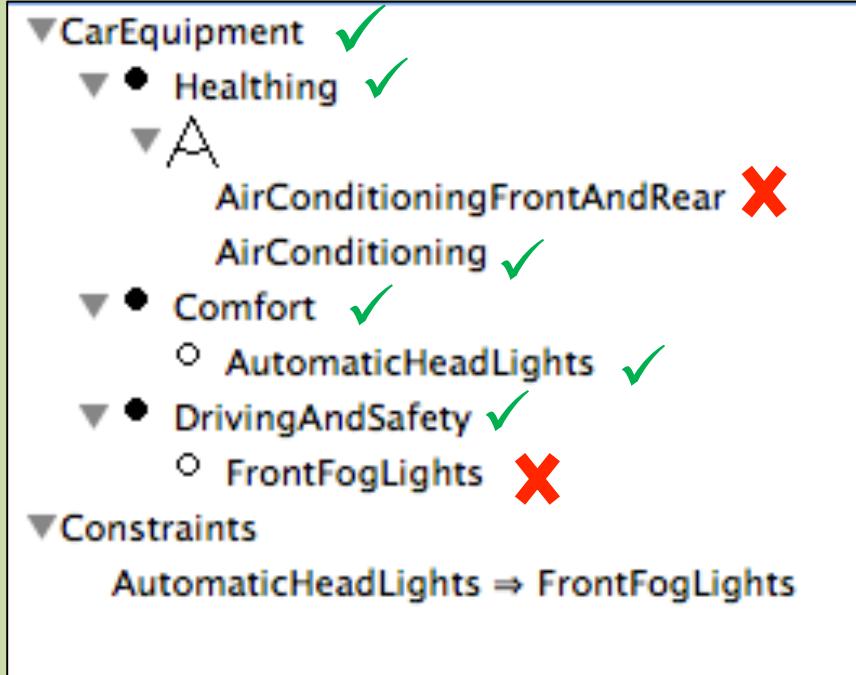
=

set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning, AirConditioningFrontAndRear, FrontFogLights}





Hierarchy + Variability

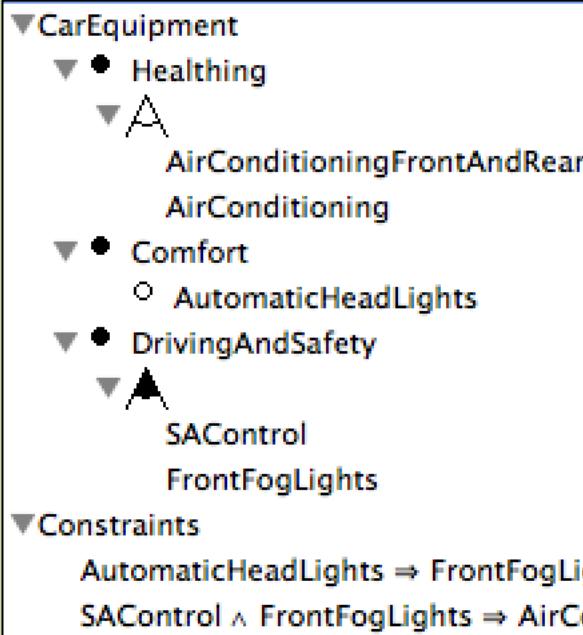
=

set of valid configurations

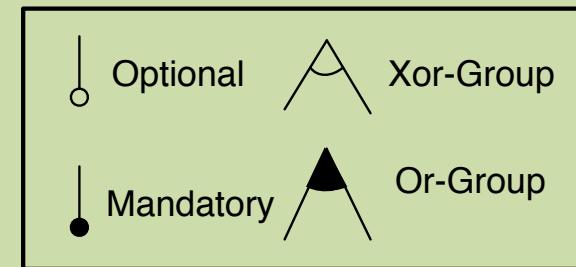
configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning, AutomaticHeadLights}





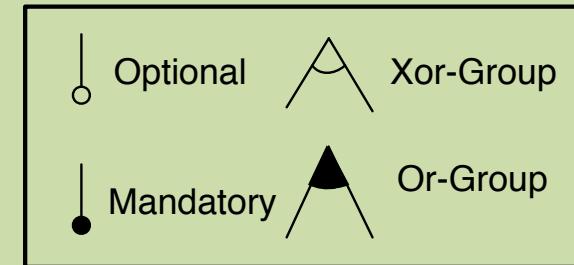
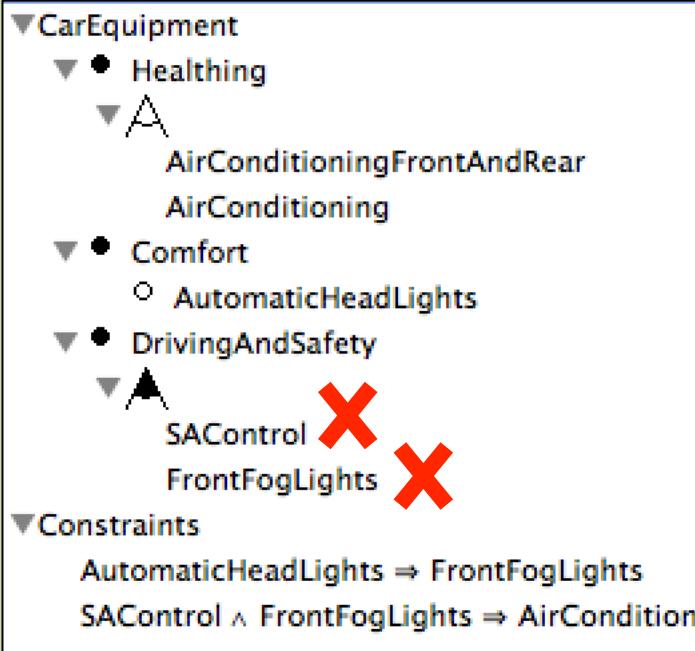
Boolean logic: \wedge , \vee , not, implies



Hierarchy + Variability
=

set of valid configurations





Hierarchy + Variability

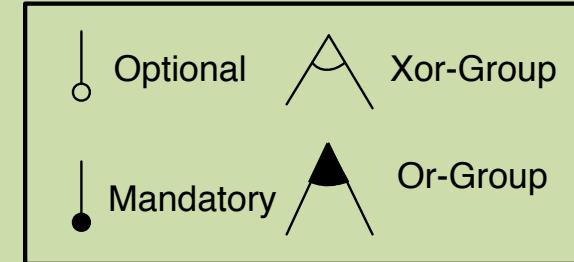
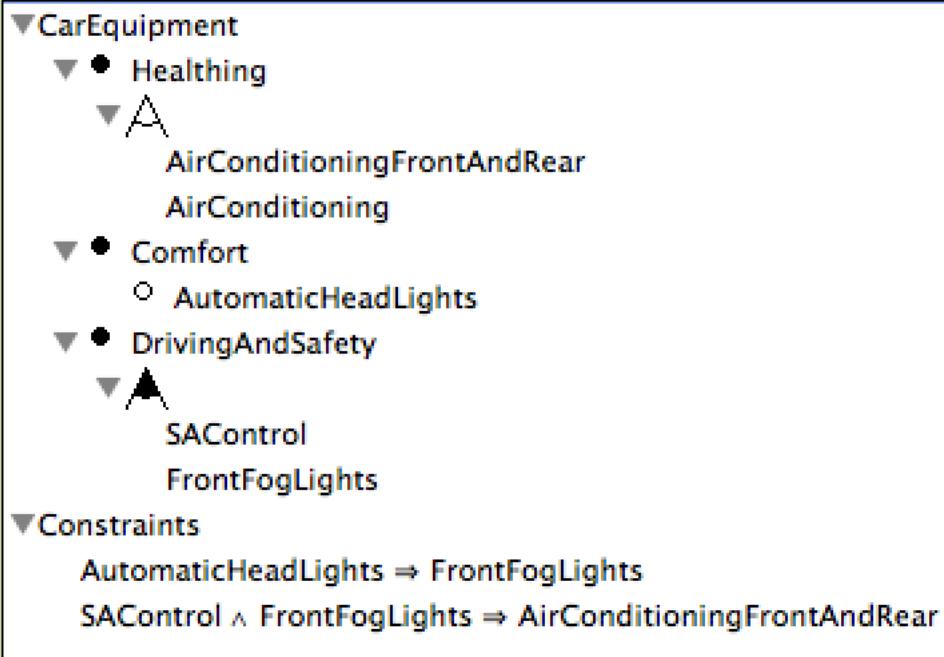
=

set of valid configurations



Or-group: at least one!





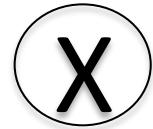
Hierarchy + Variability

=

set of valid configurations



{CarEquipment, Comfort,
DrivingAndSafety,
Healthing}

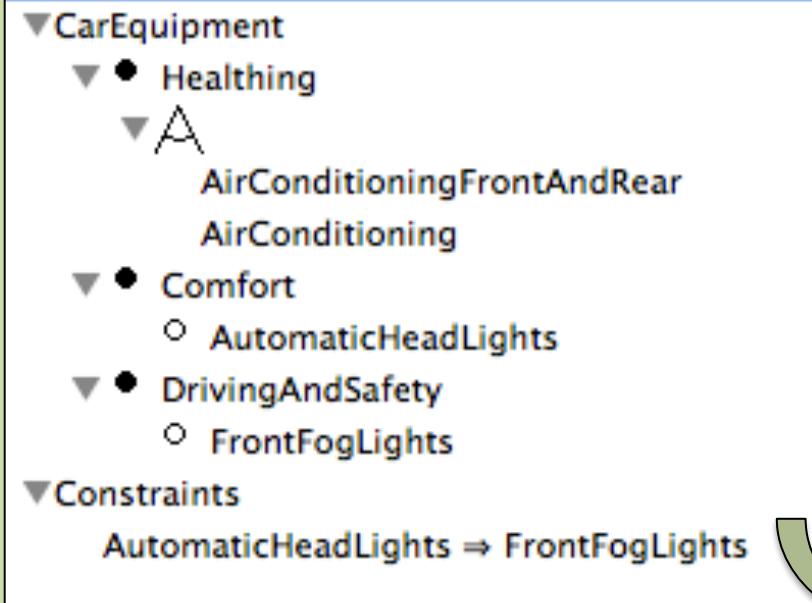
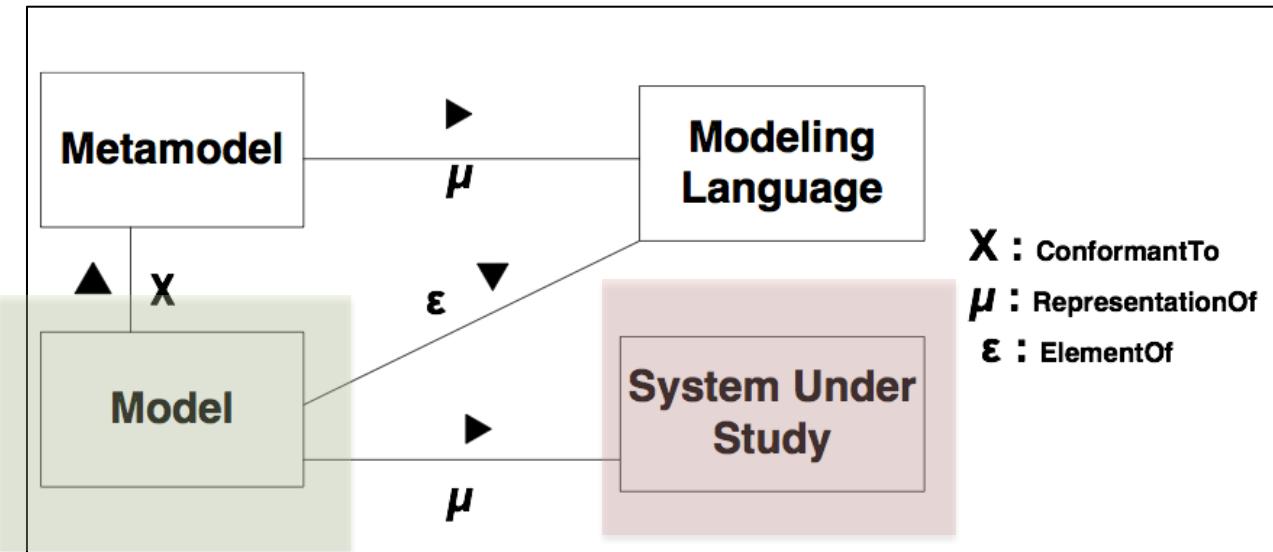


- {AirConditioningFrontAndRear, FrontFogLights, SAControl}
- {AirConditioningFrontAndRear, SAControl}
- {AutomaticHeadLights, AirConditioning, FrontFogLights}
- {AirConditioningFrontAndRear, SAControl, AutomaticHeadLights, FrontFogLights}
- {FrontFogLights, AirConditioning}
- {AutomaticHeadLights, AirConditioningFrontAndRear, FrontFogLights}
- {FrontFogLights, AirConditioningFrontAndRear}
- {SAControl, AirConditioning}

Languages:

How to specify feature models?

Feature Models



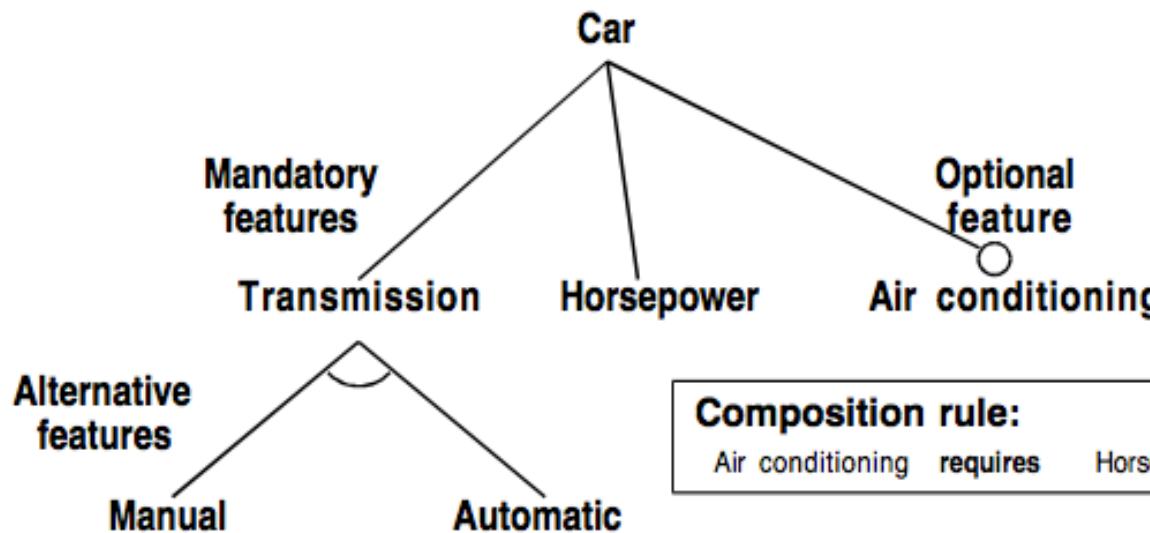
History of Feature Models

~ often called Feature Diagrams
(suggest a visual representation)

graphical languages

Feature Models

Kang et al. (1990)

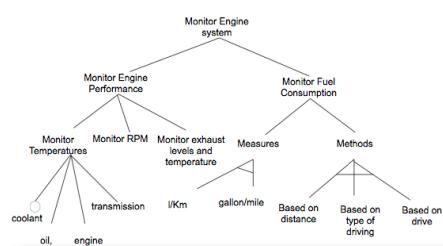


Composition rule:

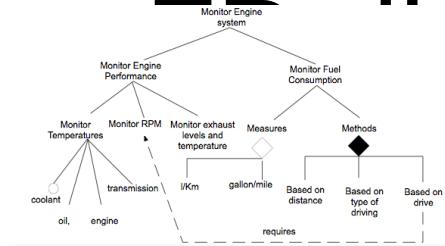
Air conditioning requires Horsepower > 100

Rationale:

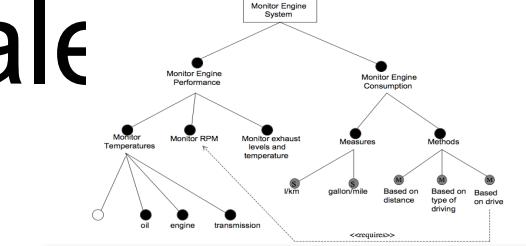
Manual more fuel efficient



FODA (OFT)
[Kang et al., 1990]

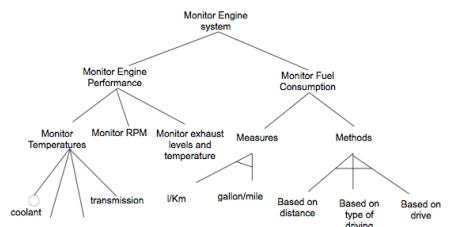


FeatuRSEB (RFD)
[Griss et al., 1998]

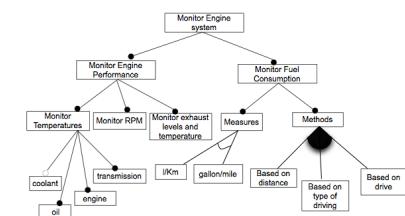


PLUSS (PFT)
[Eriksson et al., 2005]

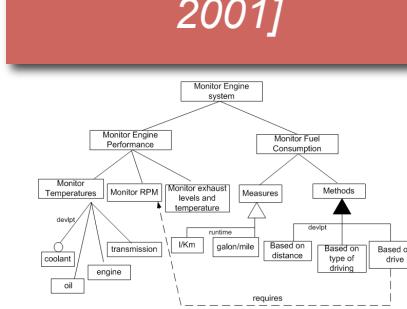
FORM (OFD)
[Kang et al., 1998]



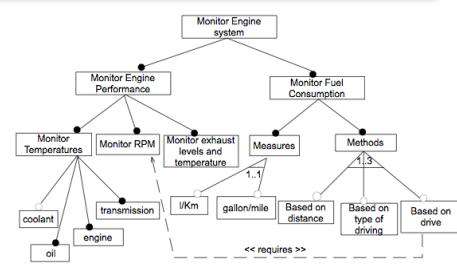
Gen. Prog. (GPFT)
[Czarnecki et al., 2000]



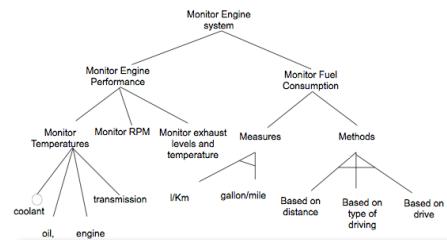
VBFD
[van Gurp et al., 2001]



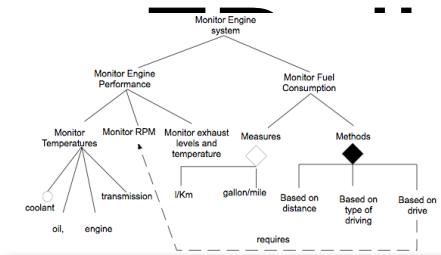
EFD
[Riebisch et al., 2002]



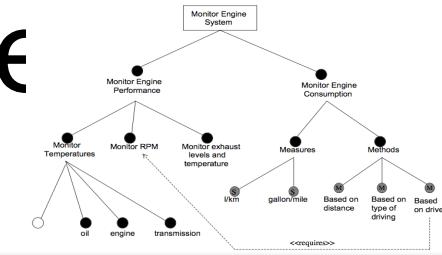
ale



FODA (OFT)
[Kang et al., 1990]



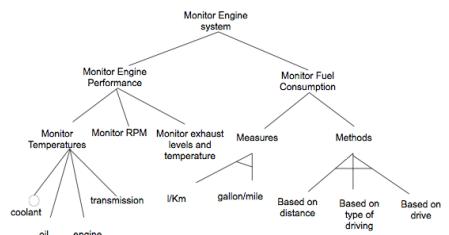
FeatuRSEB (RFD)
[Griss et al., 1998]



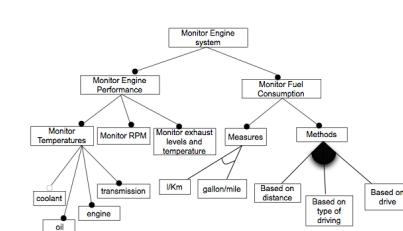
PLUSS (PFT)
[Eriksson et al., 2005]

Aesthetic differences

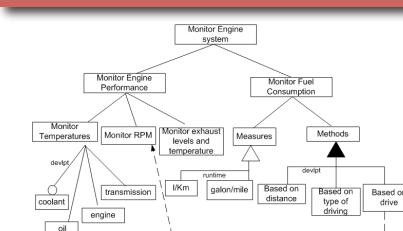
FORM (OFD)
[Kang et al., 1998]



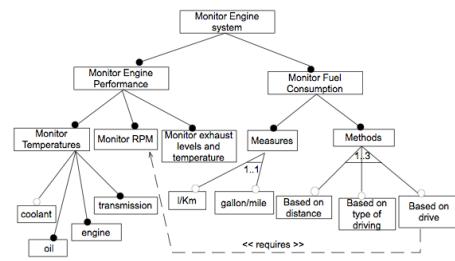
Gen. Prog. (GPFT)
[Czarnecki et al., 2000]



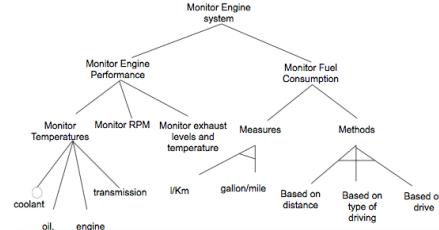
VBFD
[van Gurp et al., 2001]



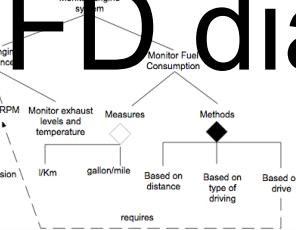
EFD
[Riebisch et al., 2002]



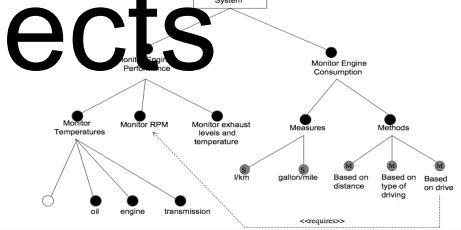
FD dialects



FODA (OFT)
[Kang et al., 1990]



FeatuRSEB (RFD)
[Griss et al., 1998]



PLUSS (PFT)
[Eriksson et al., 2005]

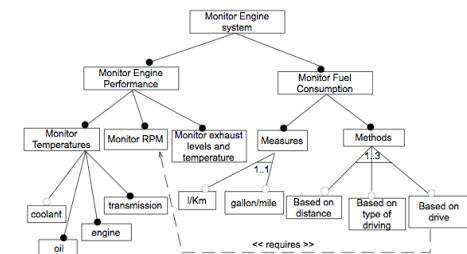
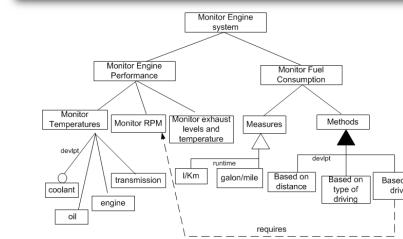
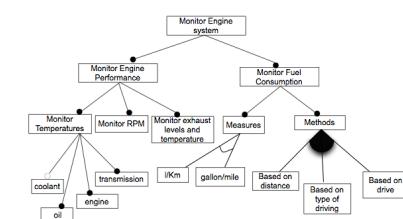
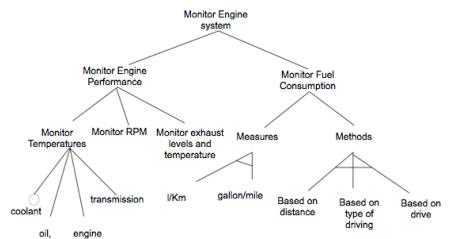
Aesthetic differences
Stronger claims

FORM (OFD)
[Kang et al., 1998]

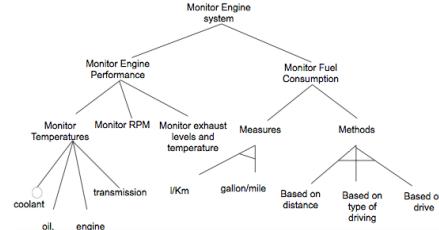
Gen. Prog. (GPFT)
[Czarnecki et al., 2000]

VBFD
[van Gurp et al., 2001]

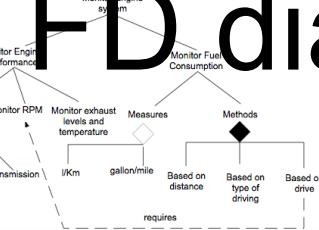
EFD
[Riebisch et al., 2002]



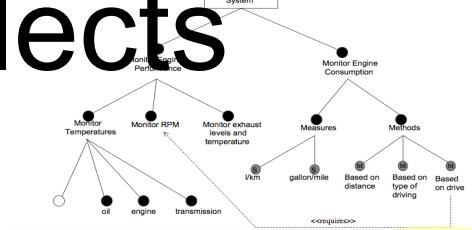
FD dialects



FODA (OFT)
[Kang et al., 1990]



FeatuRSEB (RFD)
[Griss et al., 1998]



PLUSS (PFT)
[Eriksson et al., 2001]

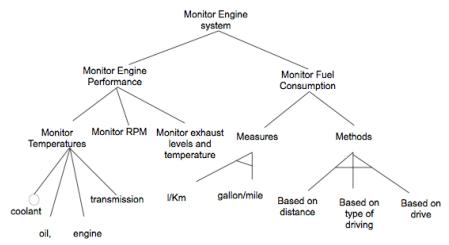
Aesthetic differences

Stronger claims

Vague use of terms
syntax, semantics,
expressiveness, ambiguity ...

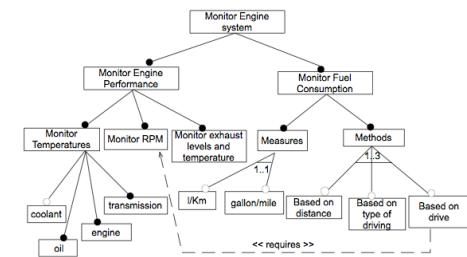
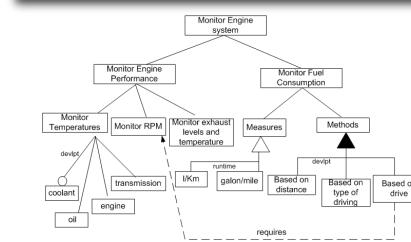
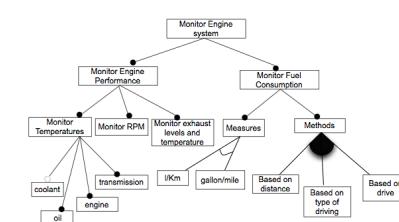
FORM (OFD)
[Kang et al., 1998]

Gen. Prog. (GPFT)
[Czarnecki et al., 2000]



VBFD
[van Gurp et al., 2001]

EFD
[Riebisch et al., 2002]



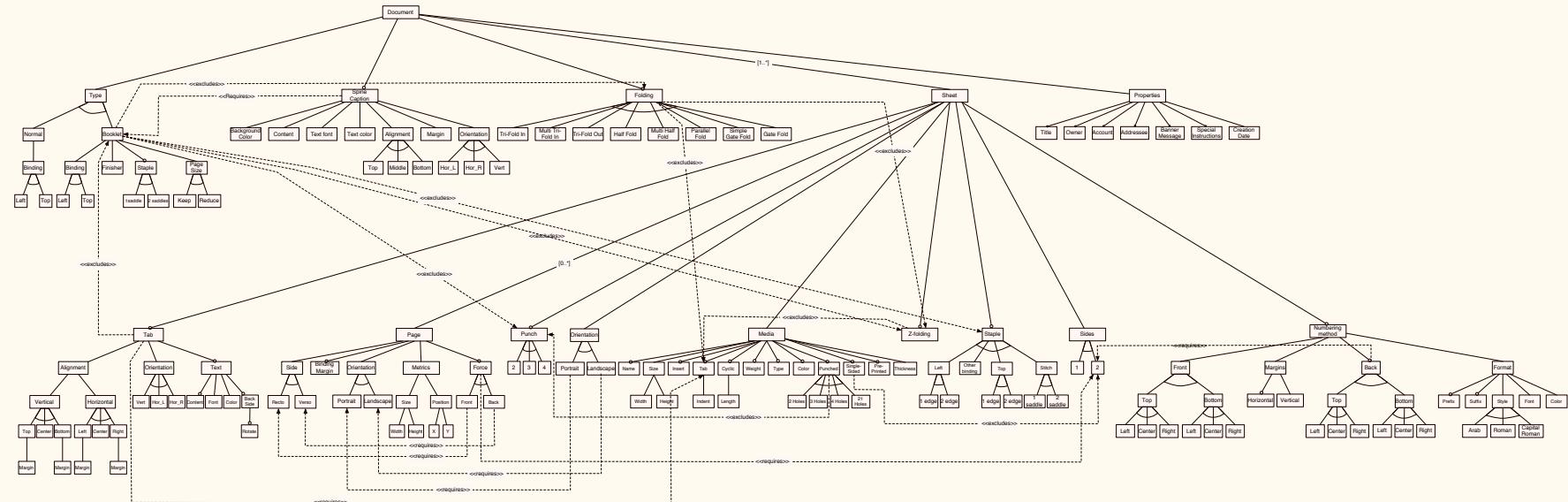
Concrete (graphical)
syntax matters...

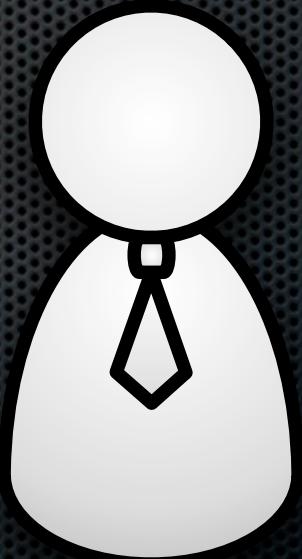
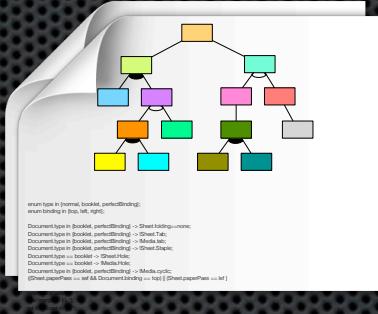
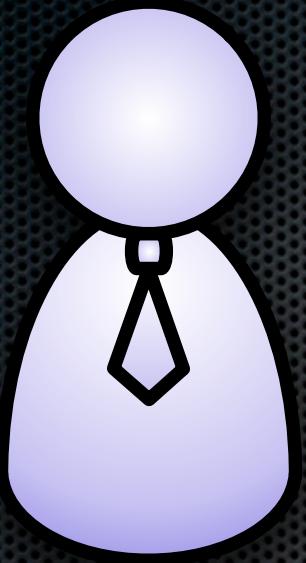
But semantics even more!

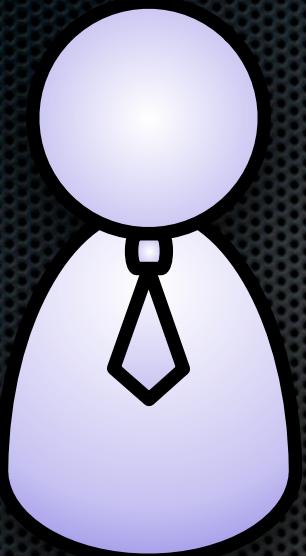
Feature Models

~ Feature Diagrams
(suggest a visual representation)

but that's not mandatory
that does not necessary scale
or respond to users' concerns







Visualisation will help, right ?

- Tools create their own problems
- Lock-in, dependance
- Some information is inevitably textual
 - OCL constraints in UML
 - Minispecs in StateCharts
 - Attributes and constraints in FD

Why not use text altogether ?

- There are powerful tools for that
- Helps with
 - Editing
 - Transformation
 - Versioning
 - Information exchange
 - ...
- Graphical views can be generated anyway

Not a new idea

Language	User Friendly	Attributes	Cardinalities	Cross tree constraints	Modularisation
Van Deursen et al.	✓				
GUIDSL (AHEAD)	✓				
SXFM	✓				
XML-based (FAMA etc.)		✓	✓	✓	



+ CML
+ VSL

Furthermore

- Intuitive C-like syntax
- Formal semantics
- Statically typed
- Editing facilities
 - Textual editor with auto-completion

TVL : allOf – and decomposition

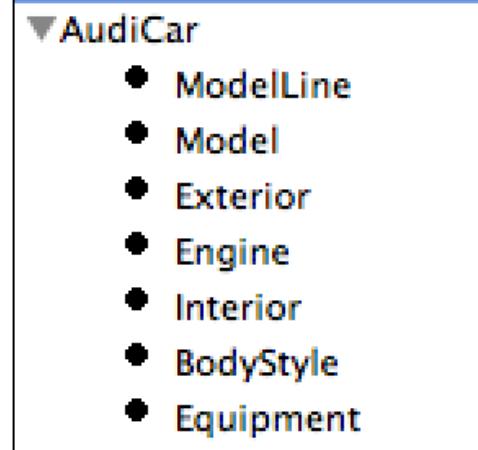
```
root AudiCar {  
    group allOf {  
        ModelLine,  
        BodyStyle,  
        Model,  
        Engine,  
        Exterior,  
        Interior,  
        Equipment  
    }  
}
```

root feature

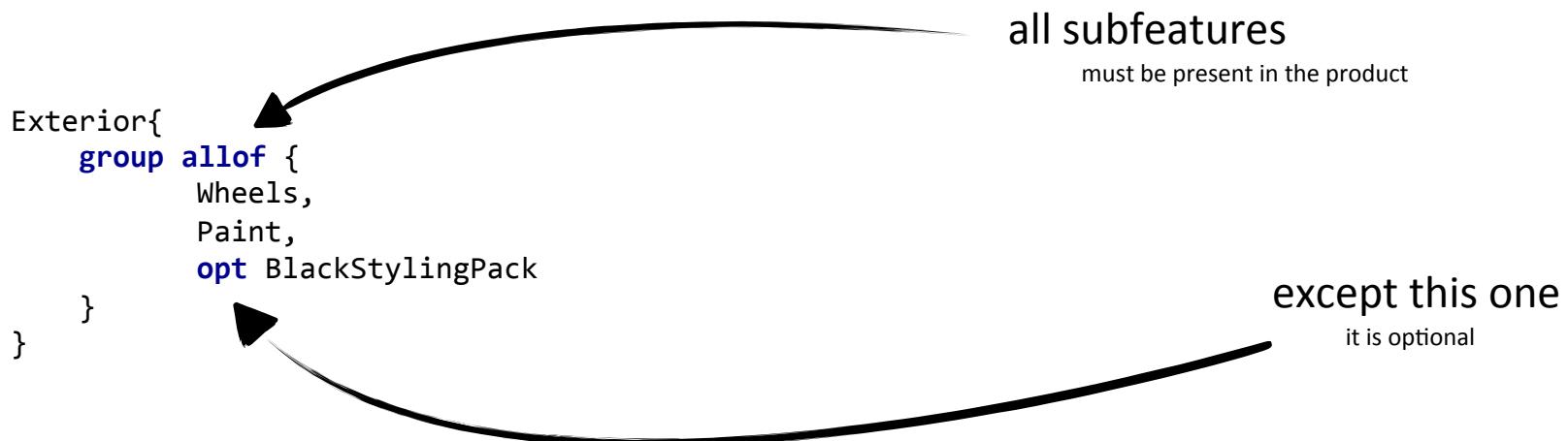
and decomposition

all subfeatures must be present in the product

sub features



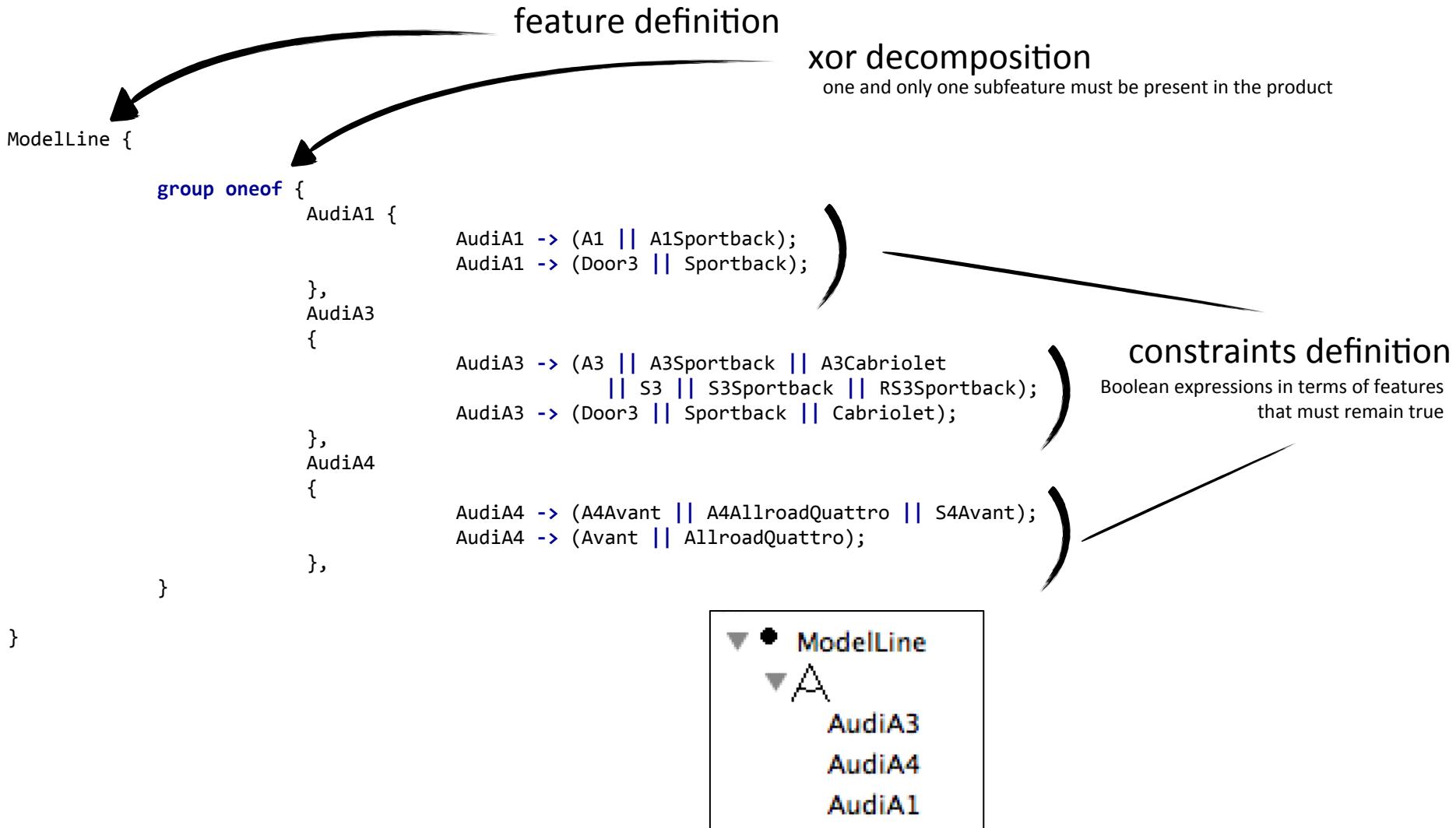
TVL : optional features



▼ Exterior

- BlackStylingPack
- Paint
- Wheels

TVL : one of – xor decomposition

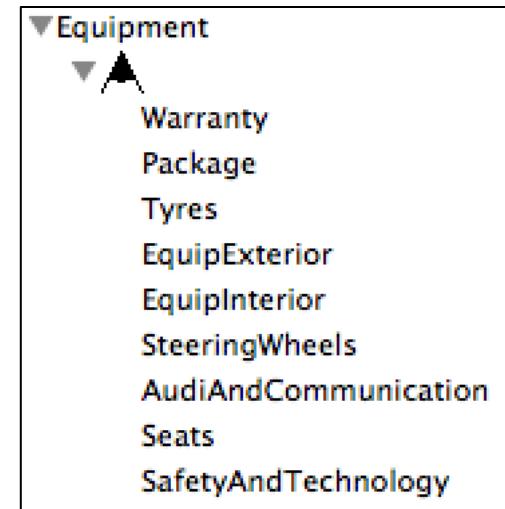


TVL : someOf – or decomposition

feature definition

```
Equipment{  
    group someof {  
        Package,  
        EquipExterior,  
        Tyres,  
        EquipInterior,  
        SteeringWheels,  
        Seats,  
        SafetyAndTechnology,  
        AudiAndCommunication,  
        Warranty  
    }  
}
```

or decomposition
one or more subfeatures must be present in the product



TVL : Modularity

```
root Car {  
    group allOf {  
        BodyStyle,  
        Model,  
        Engine,  
        Equipment }  
  
}  
  
include(BodyStyle.tvl);  
include(Model.tvl);  
include(Engine.tvl);  
include(Equipment.tvl);
```



Split model in reusable parts
using external files

TVL : Modularity

BodyStyle.tvl

```
BodyStyle {  
    group oneof {  
        Avant,  
        Door3,  
        Sportback,  
        Coupe,  
        Cabriolet,  
        Roadster,  
        Spyder,  
        SUV  
    }  
}
```

Model.tvl

```
Model {  
    group oneof {  
        A1,  
        A3,  
        S3,  
        A4,  
        S4,  
        A5,  
        S5,  
        RS5,  
        A6,  
        A7,  
        A8,  
        A8L,  
        R8  
        Q3,  
        Q5,  
        Q7,  
        TT,  
        TTS,  
        TT RS  
    }  
}
```

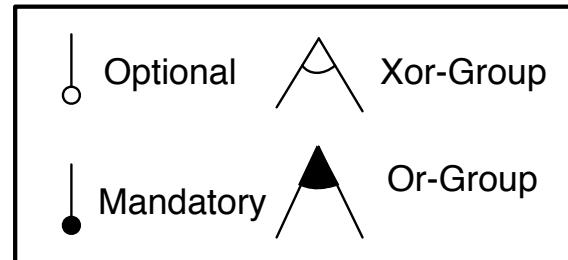
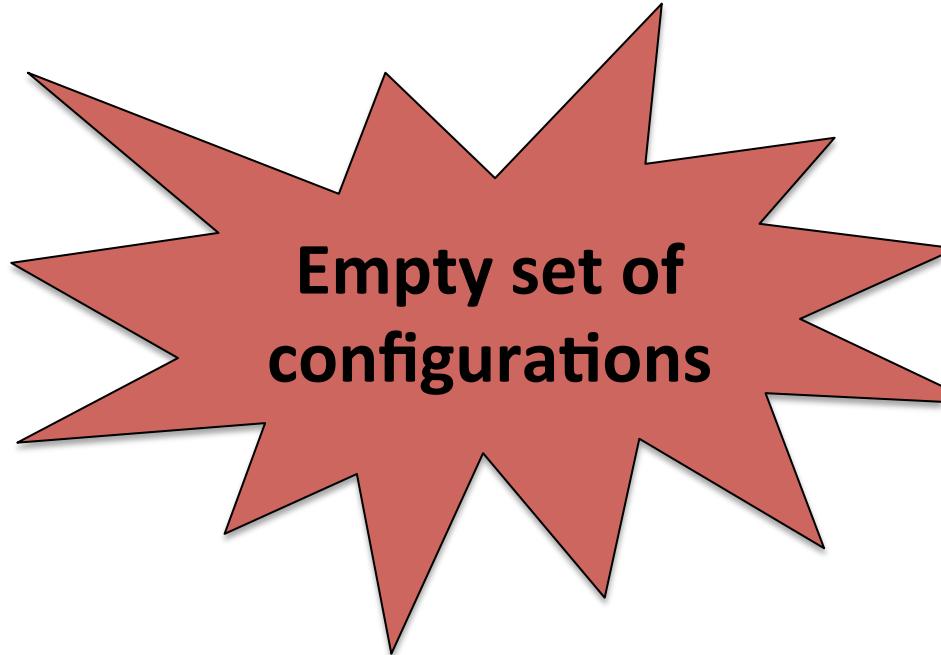
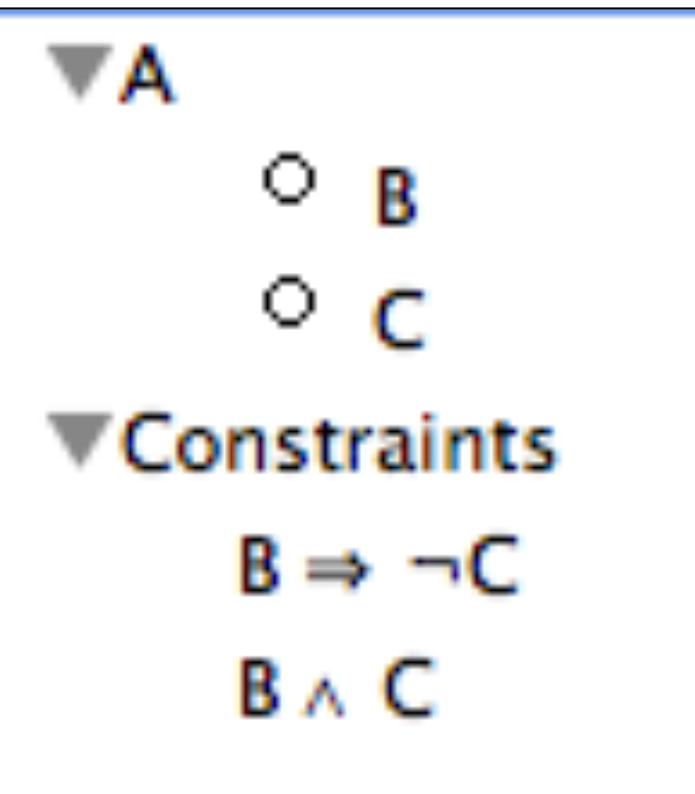
Engine.tvl

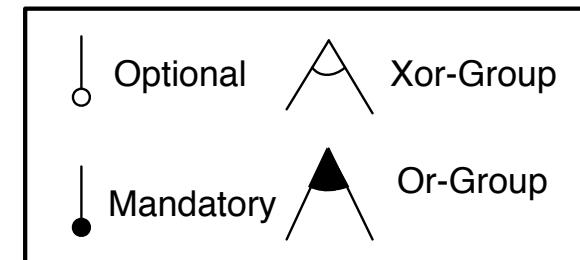
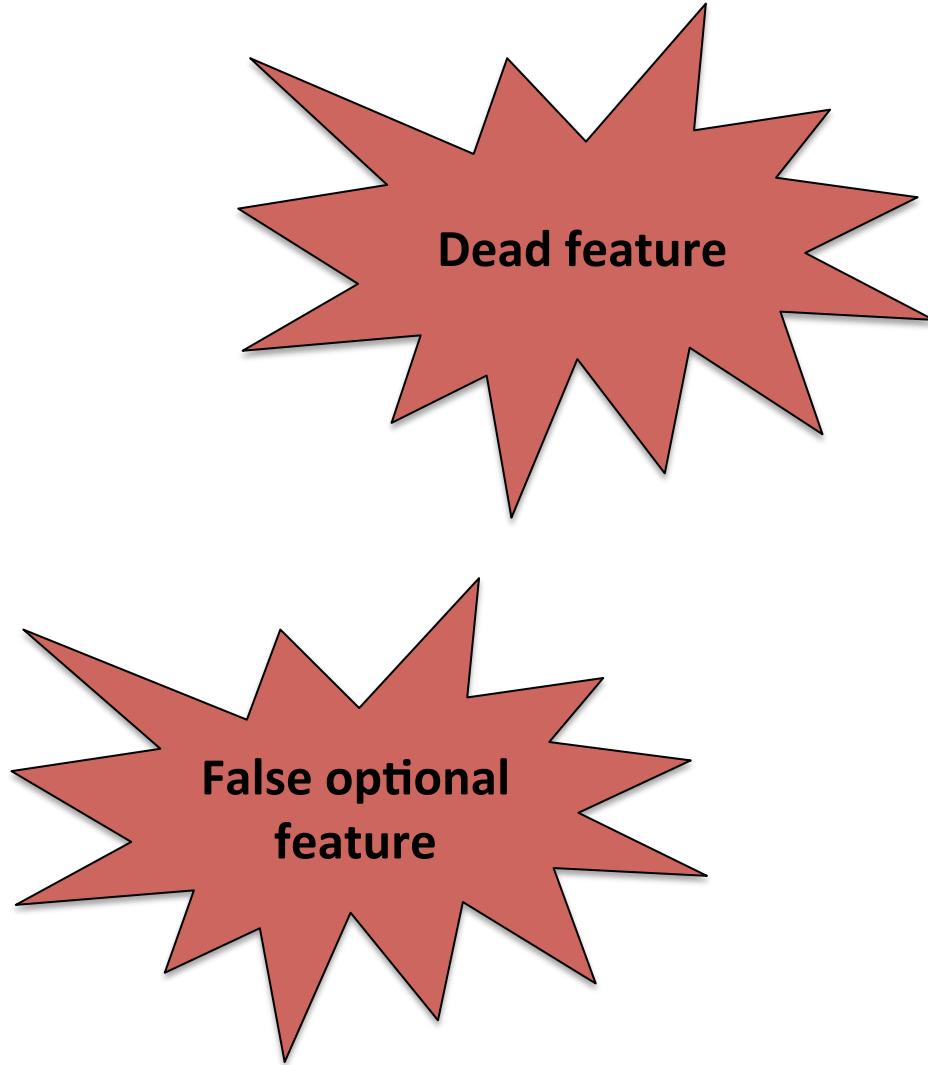
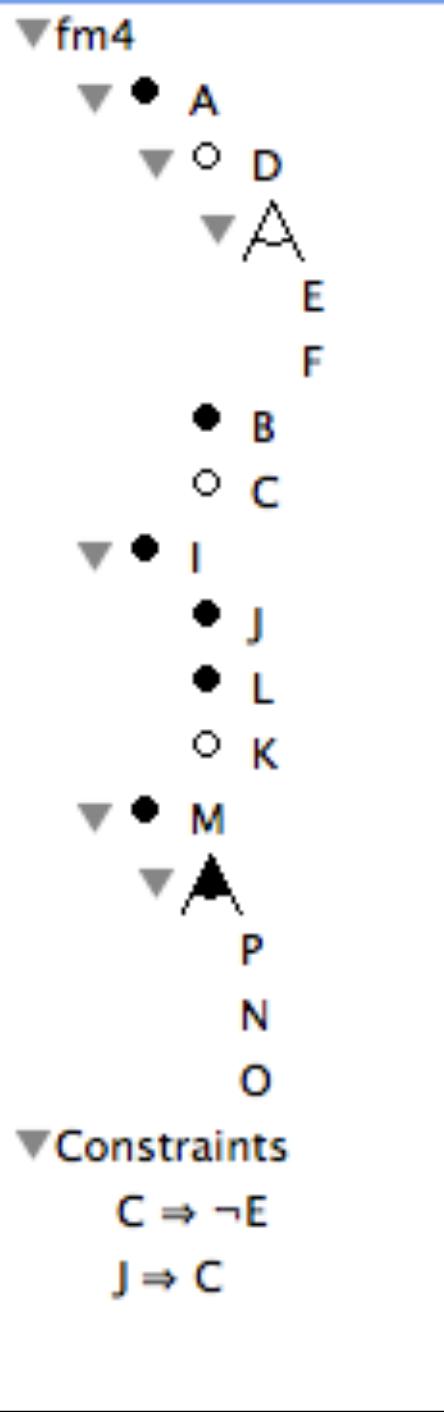
```
Engine {  
    group oneof {  
        SE12TFSI,  
        CE14TFSI,  
        CE14TFSI119,  
        Sport12TFSI,  
        Sport14TFSI,  
        Sport14TFSI119,  
        BlackEdition14TFSI,  
        SLine12TFSI,  
        SLine14TFSI,  
        SE16TDI,  
        CE16TDI,  
        CE20TDI,  
        ...  
    }  
}
```

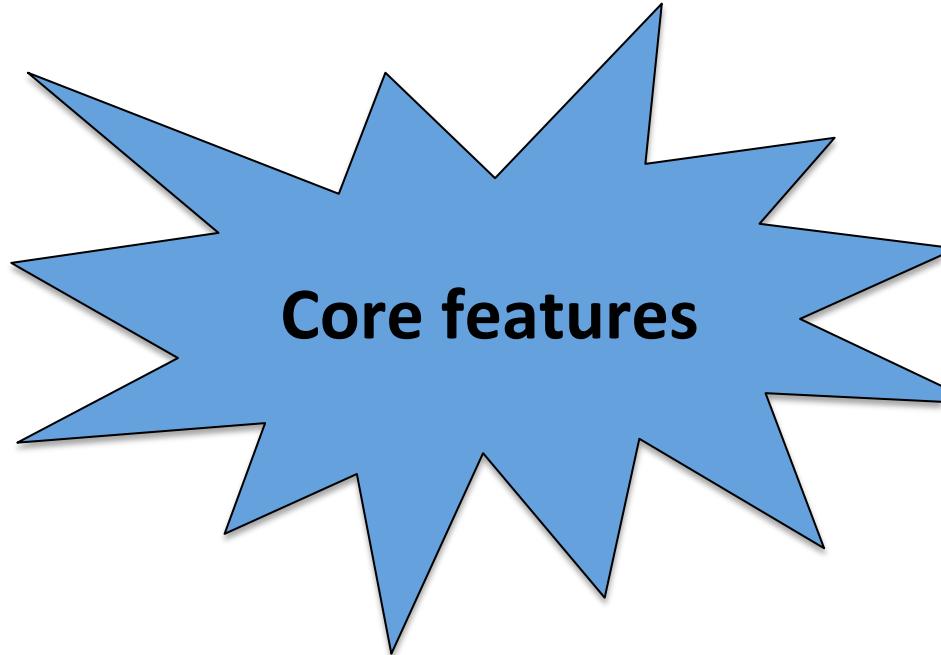
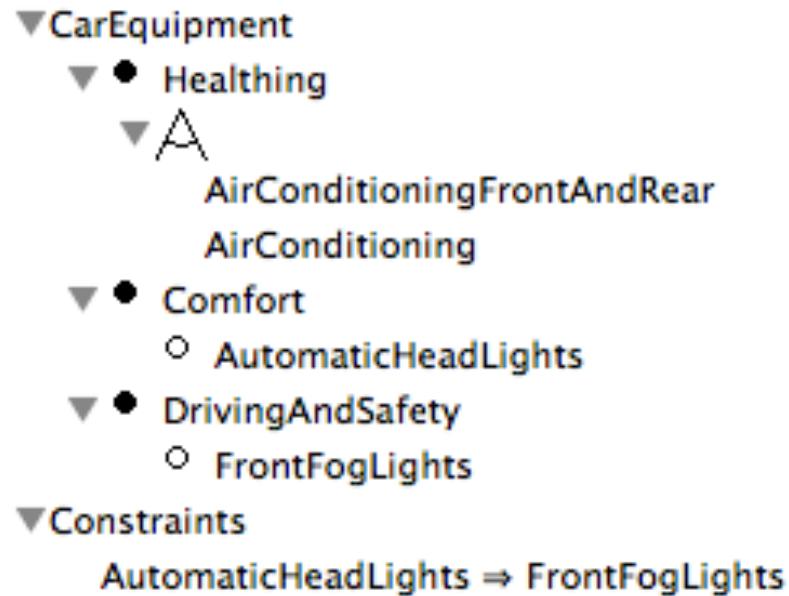
TVL : Tools

- Key references
 - Classen, A.; Boucher, Q. and Heymans, P. A Text-based Approach to Feature Modelling: Syntax and Semantics of TVL. In Science of Computer Programming (SCP), Special Issue on Software Evolution
 - Hubaux, A.; Boucher, Q.; Hartman, H.; Michel, R. and Heymans, P. **Evaluating a Text-based Feature Modelling Language: Four Industrial Case (SLE 2010)**
 - <http://info.fundp.ac.be/tvl/>
- Support
 - Parser / Syntactic and semantic checker
 - SAT-based checker for Boolean models
- Lessons learned
 - Textual vs graphical languages
 - Semantics of your language matter

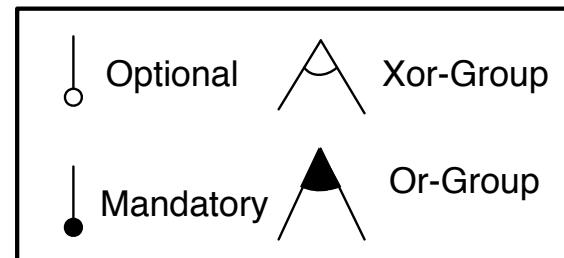
I want to analyze and
play with my specification!

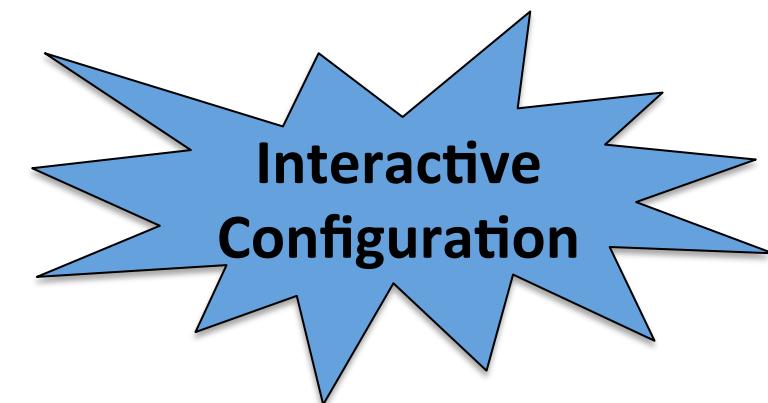
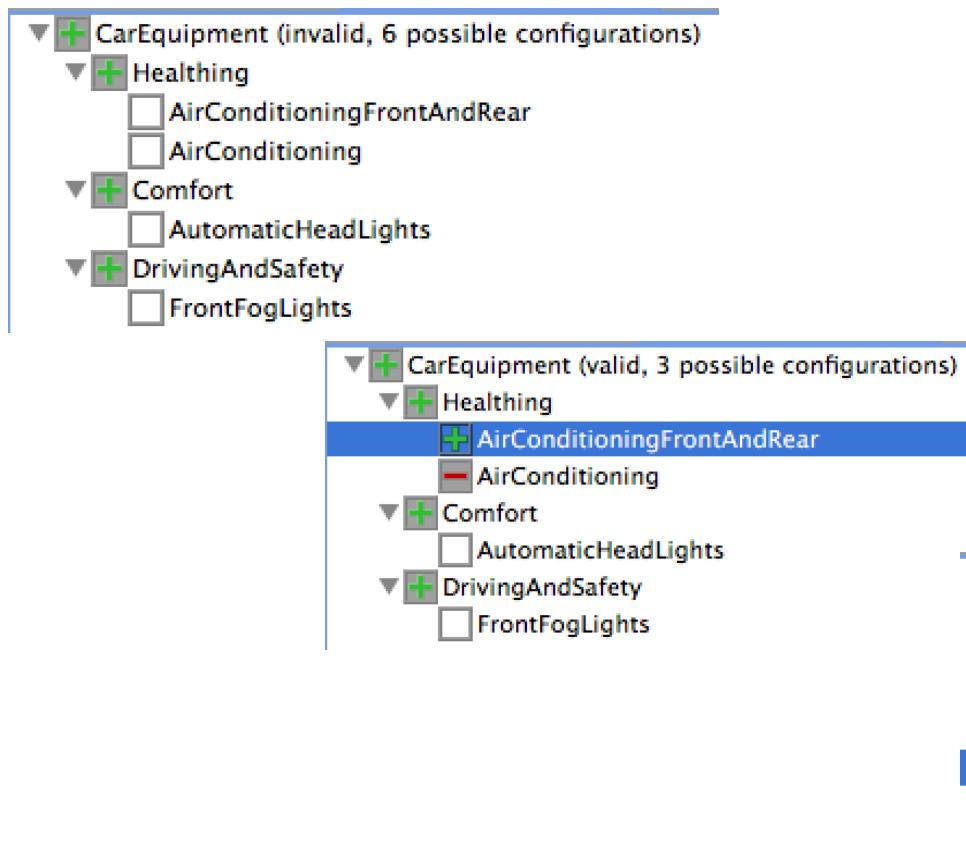
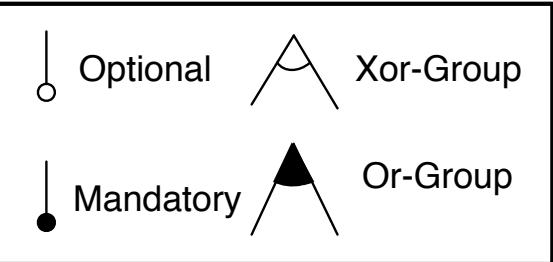
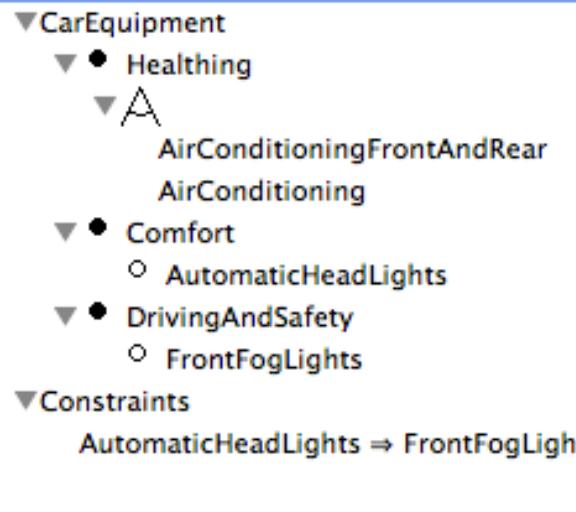






{CarEquipment, Comfort,
DrivingAndSafety, Healthing}





Feature Models and Automated Reasoning

Benavides et al. survey, 2010

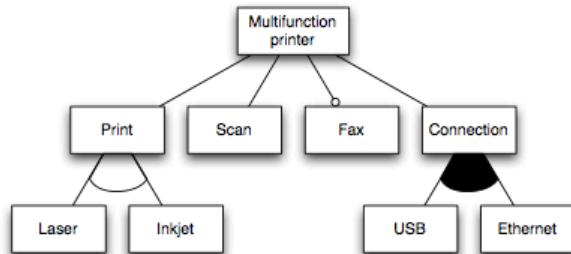
Decision problems and complexity

- Validity of a feature model
- Validity of a configuration
- Computation of dead and core features
- Counting of the number of valid configurations
- Equivalence between two feature models
- Satisfiability (SAT) problem
 - NP-complete

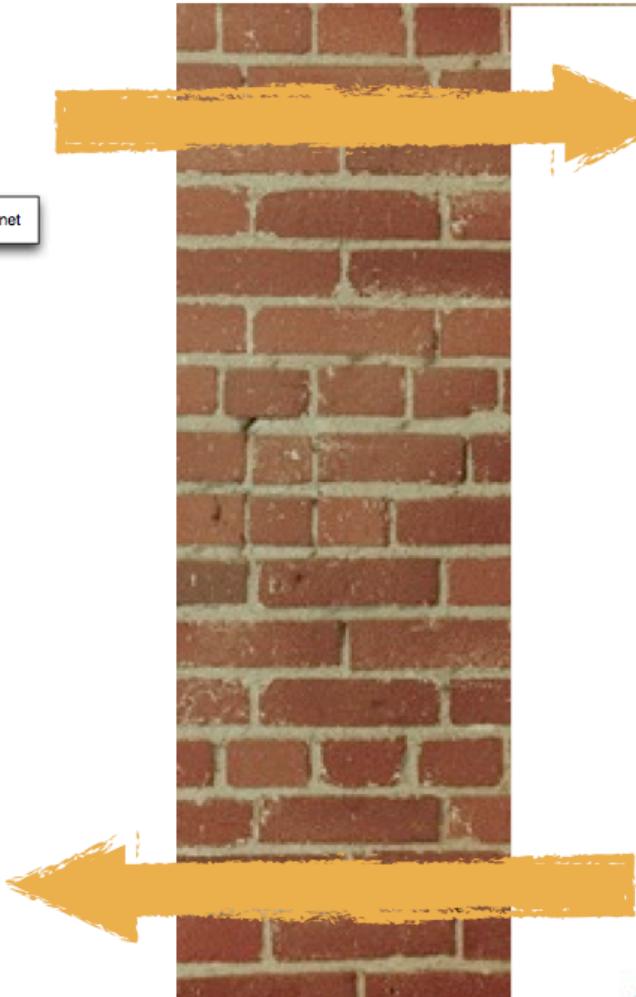
How to automate analysis of your feature models?

Binary Decision Diagram (BDD)
SAT solver

Typical implementations



Fontsource (Attributed - Free Processing 2012) (Attributed - Creative Commons)



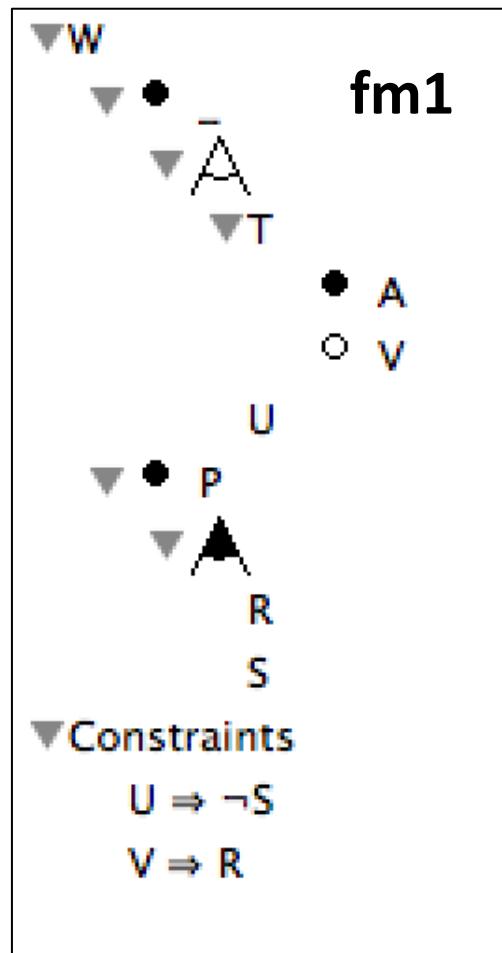
logics
solvers



Z3

(Boolean) Feature Models

Hierarchy + Variability = set of valid configurations



$\llbracket fm1 \rrbracket = \{$

$\{W, P, R, S, T, A, V\},$

$\{W, P, S, T, A\},$

$\{W, P, R, T, A\},$

$\{W, P, R, U\},$

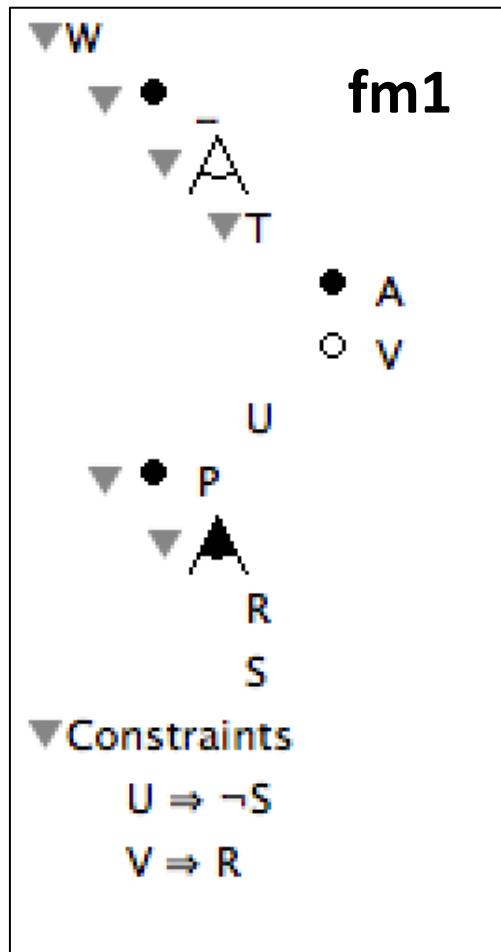
$\{W, P, R, T, V, A\},$

$\{W, P, R, S, T, A\},$

$\}$

(Boolean) Feature Models

~ Boolean formula

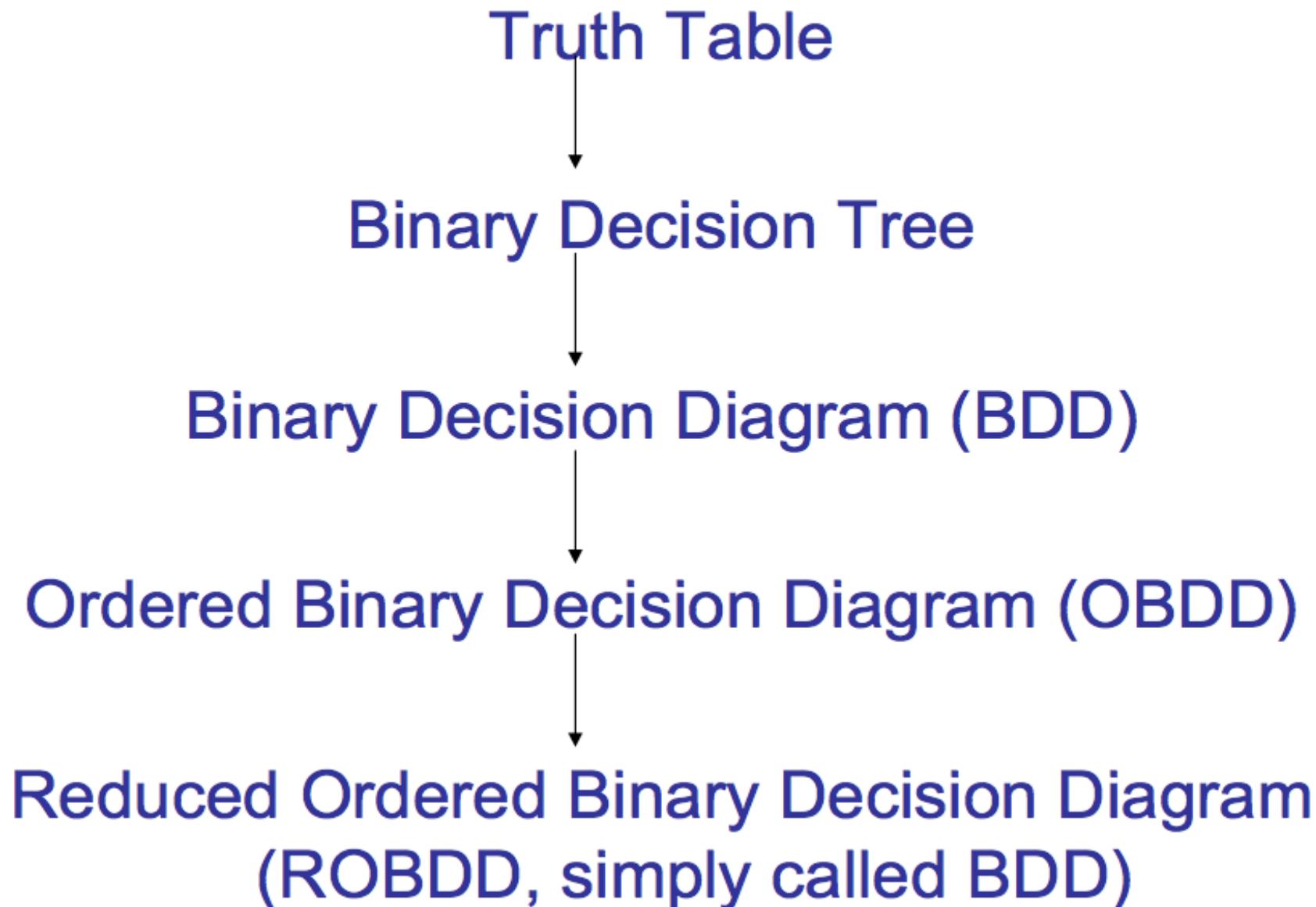


$\phi_{fm_1} = W // \text{root}$
 $\wedge W \Leftrightarrow P // \text{mandatory}$
 $// \text{Or-group}$
 $\wedge P \Rightarrow R \vee S$
 $\wedge R \Rightarrow P \wedge S \Rightarrow P$
 $\wedge V \Rightarrow T // \text{optional}$
 $\wedge A \Leftrightarrow T // \text{mandatory}$
 $// \text{Xor-group}$
 $\wedge T \Rightarrow W$
 $\wedge U \Rightarrow W$
 $\wedge \neg T \vee \neg U$
 $// \text{constraints}$
 $\wedge V \Rightarrow R // \text{implies}$
 $\wedge \neg U \Rightarrow \neg S // \text{excludes}$

Truth table, boolean function

from		to		
x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

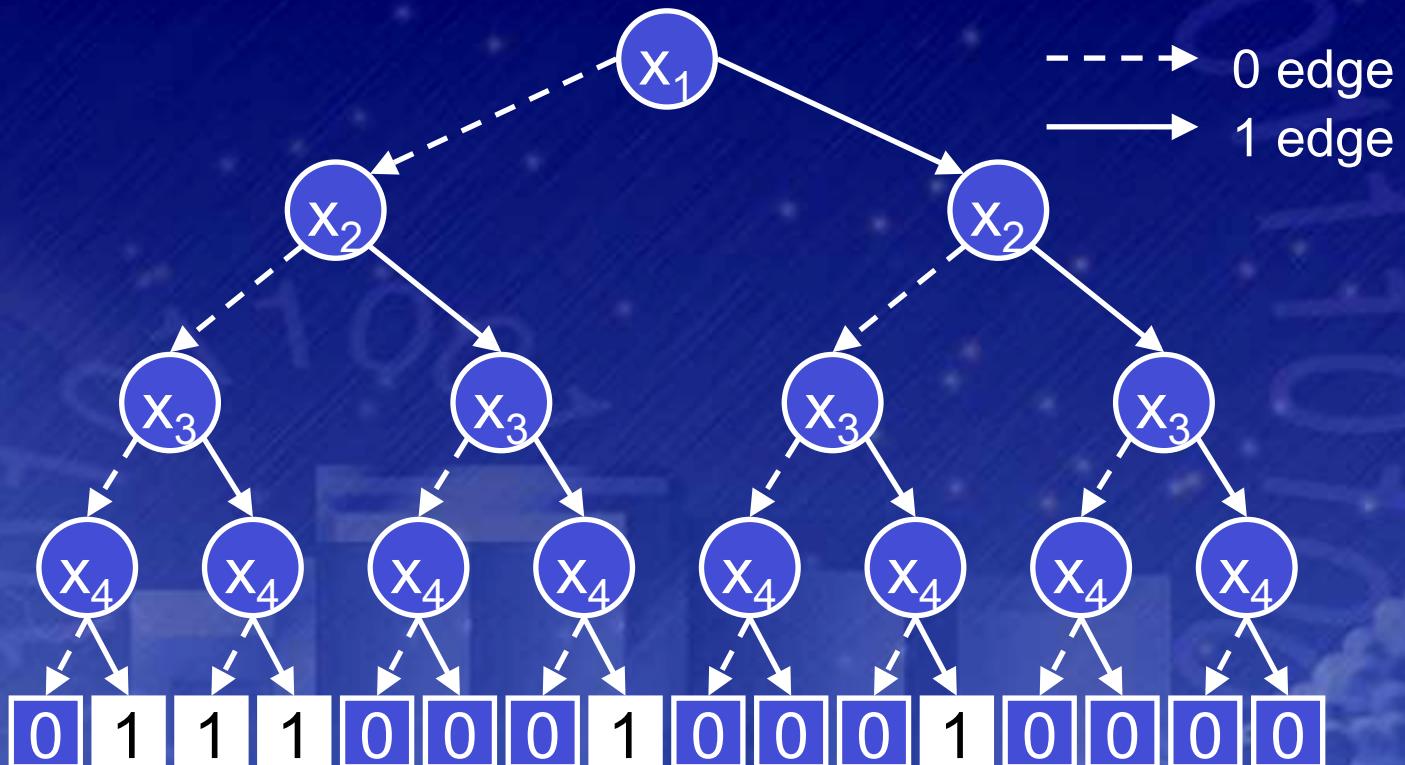
BDDs from Truth Tables



Binary Decision Diagrams

(Bryant 1986)
encoding of a truth table.

<i>from</i>		<i>to</i>		
x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Reduction

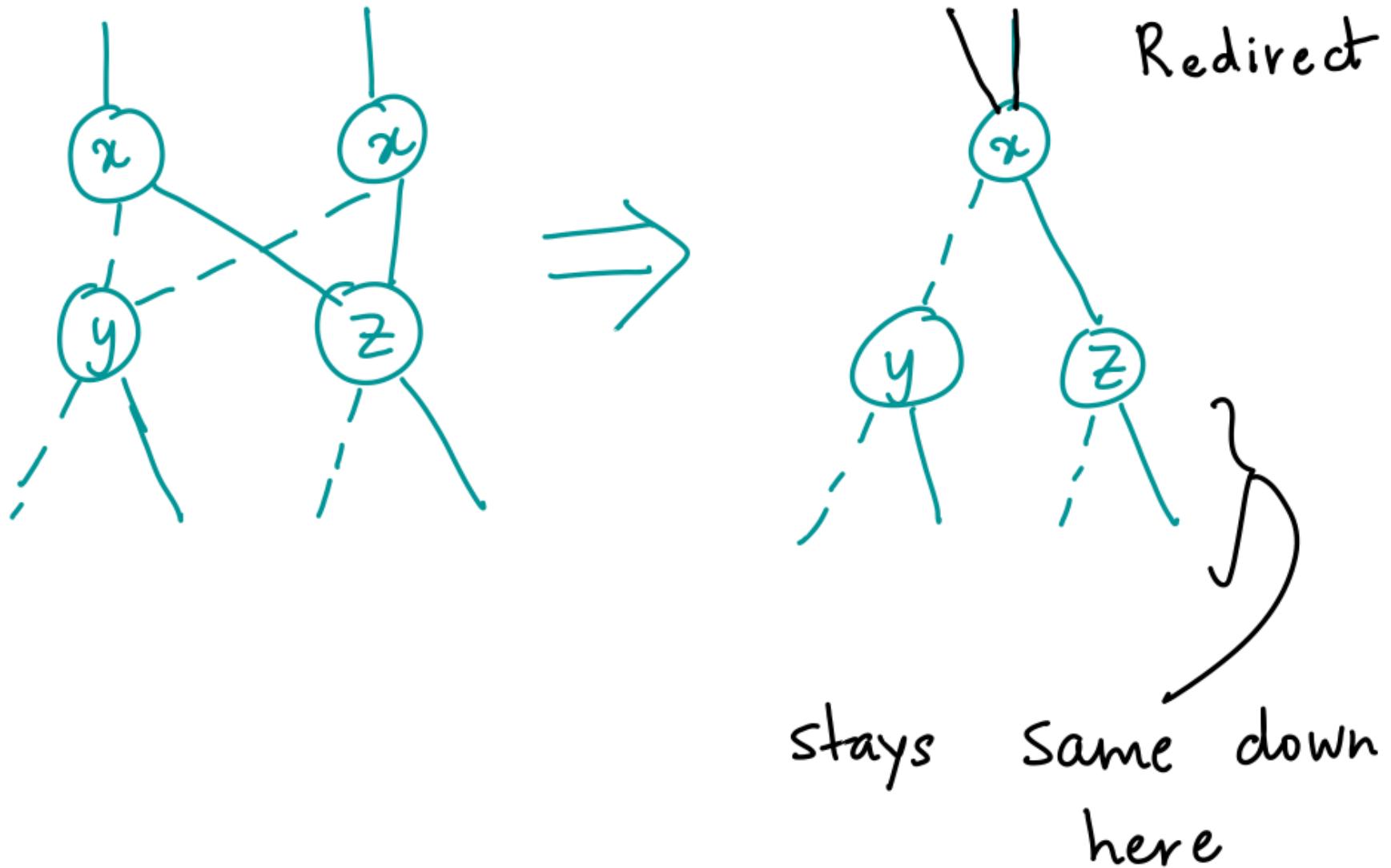
- Identify Redundancies
- 3 Rules
 - Merge equivalent leaves
 - Merge isomorphic nodes
 - Eliminate redundant tests

Merge equivalent leaves

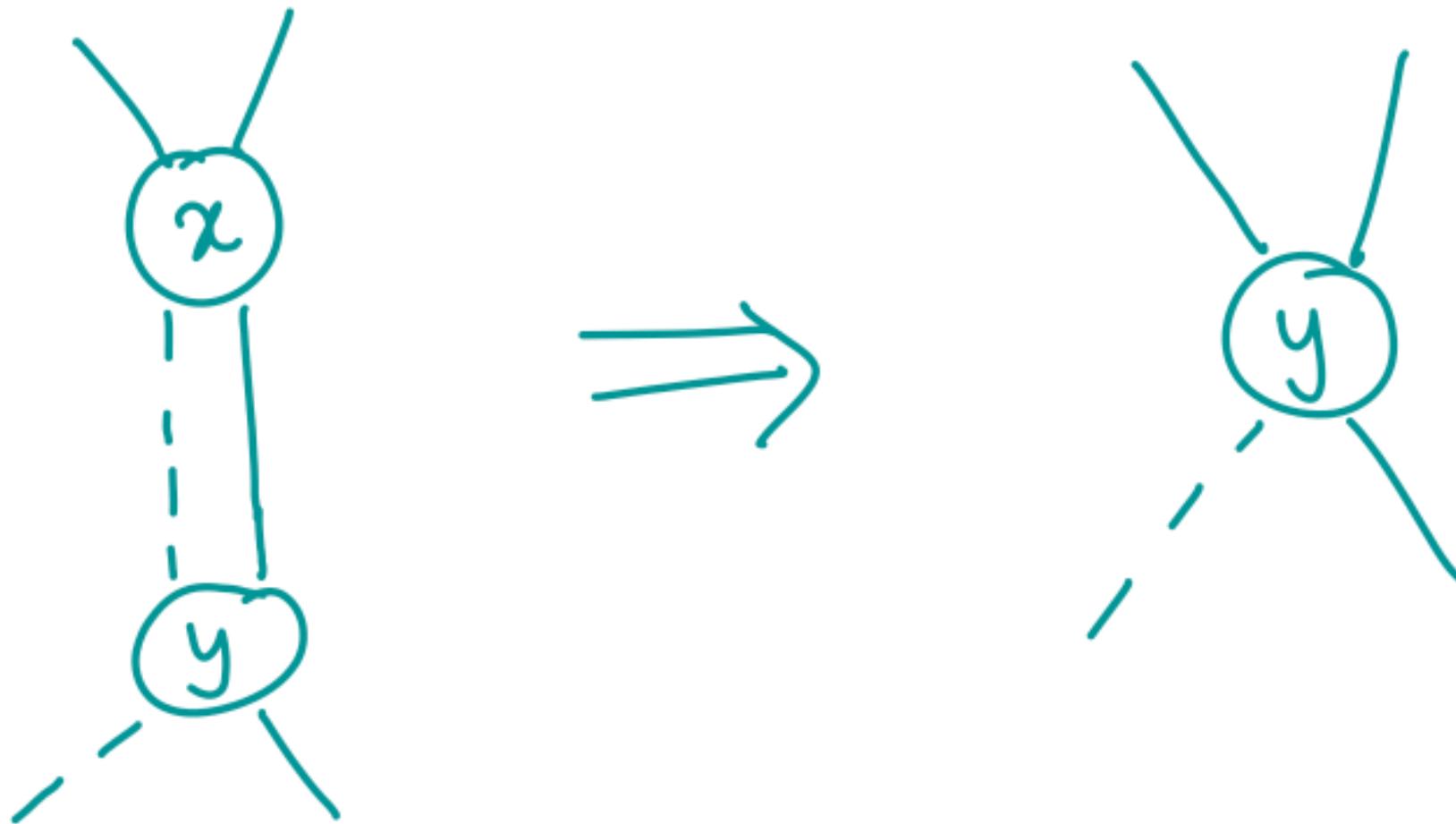


"a" is either 0 or 1

Merge isomorphic nodes

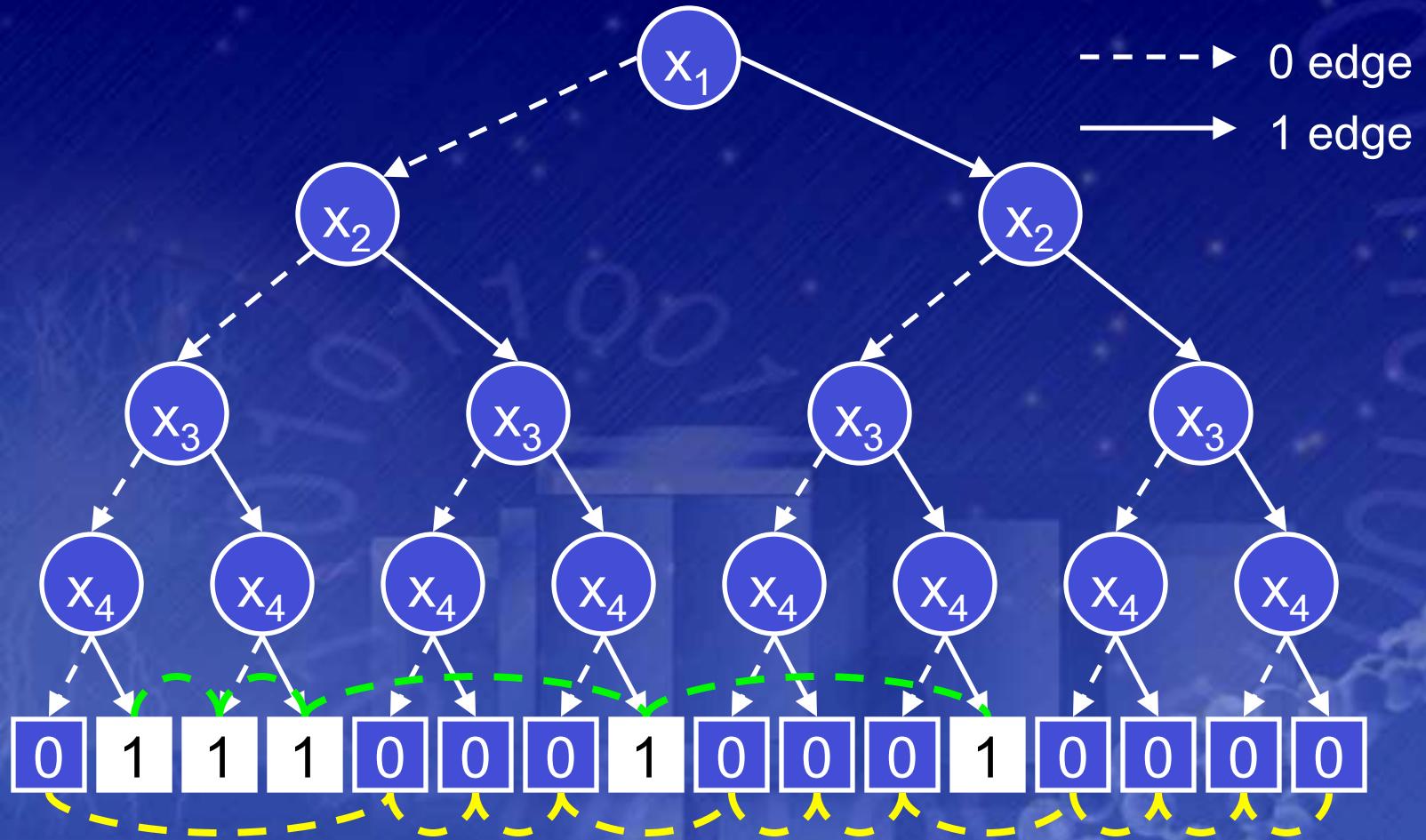


Eliminate redundant tests



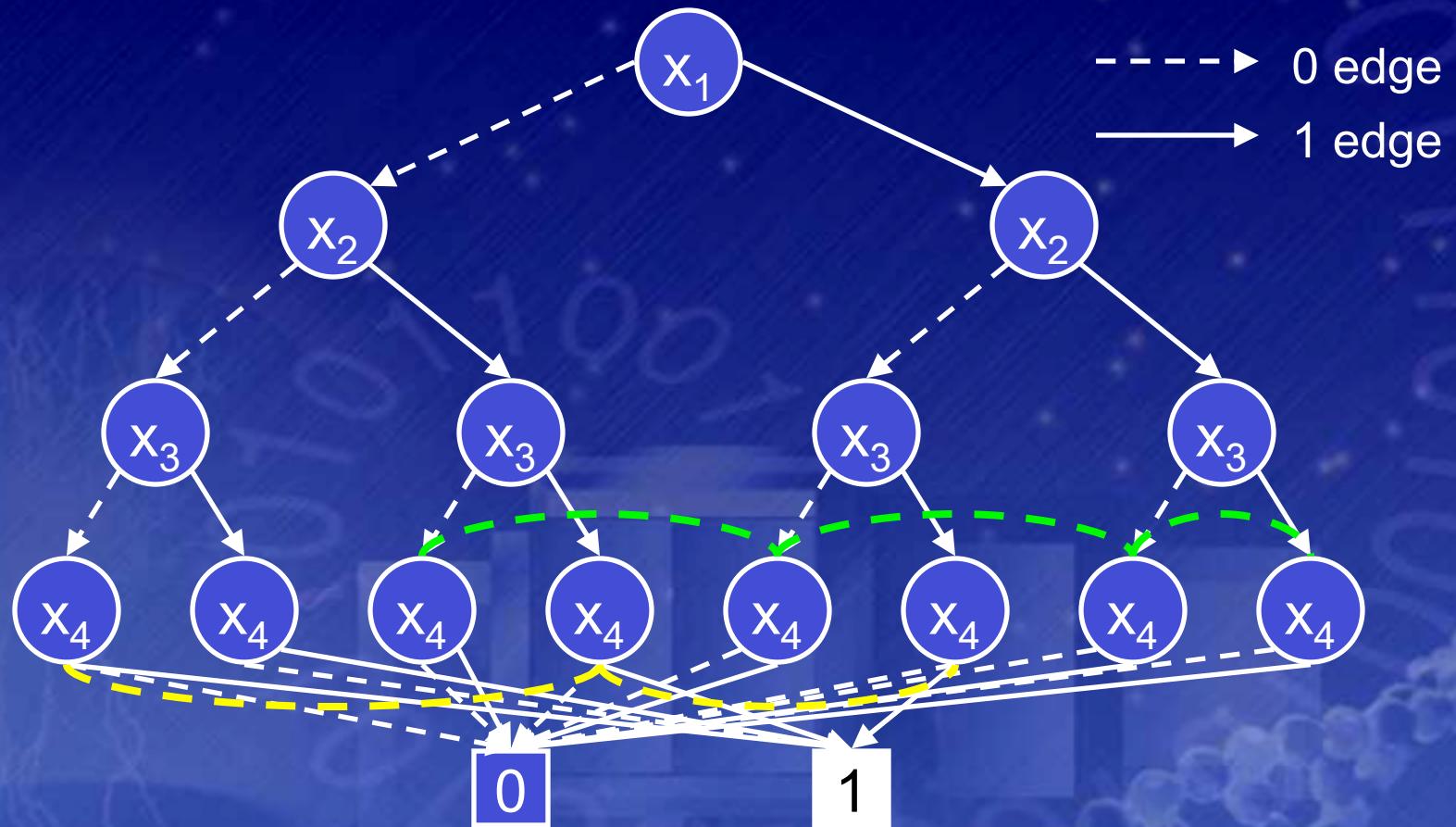
Binary Decision Diagrams

- Collapse redundant nodes.



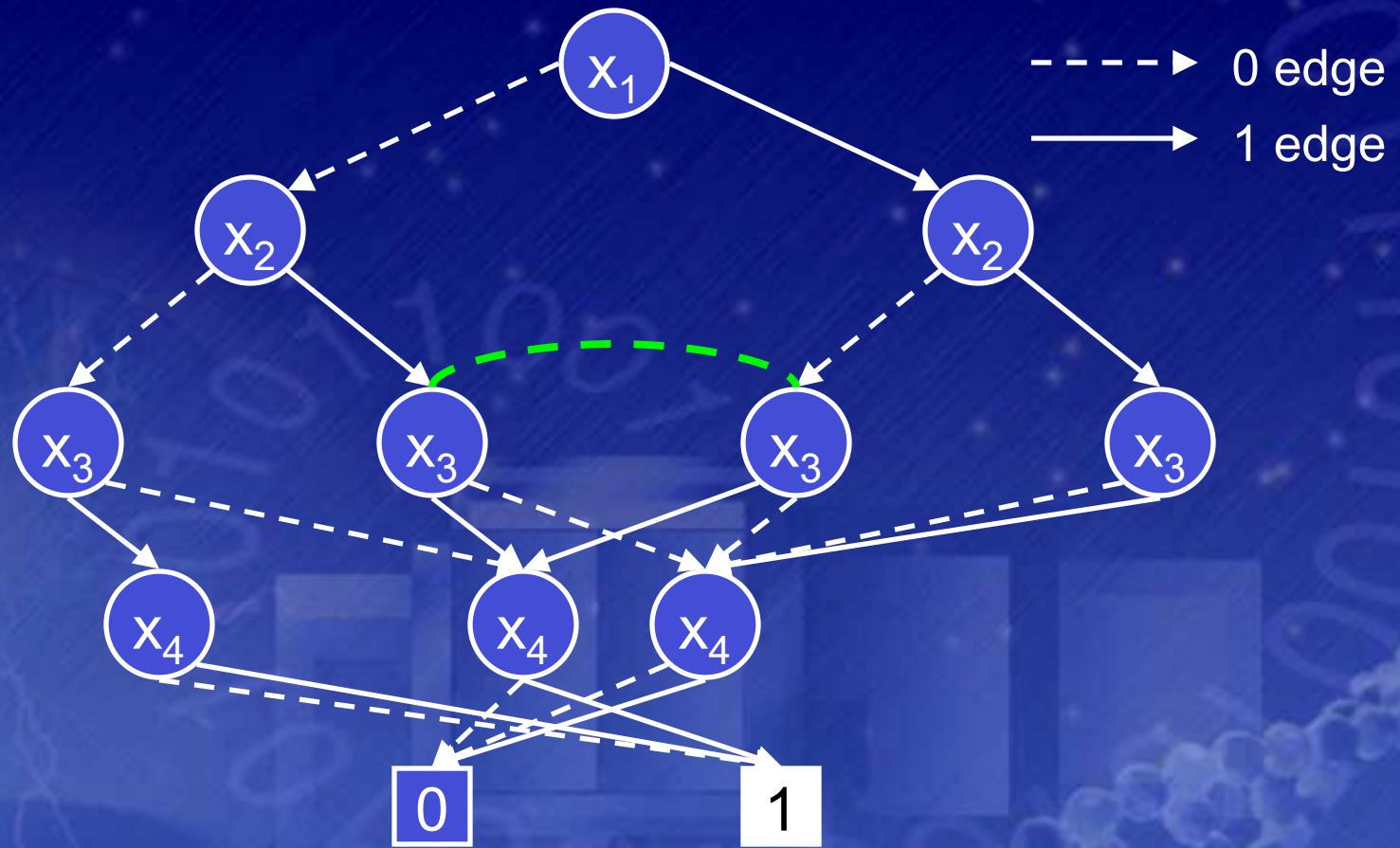
Binary Decision Diagrams

- Collapse redundant nodes.



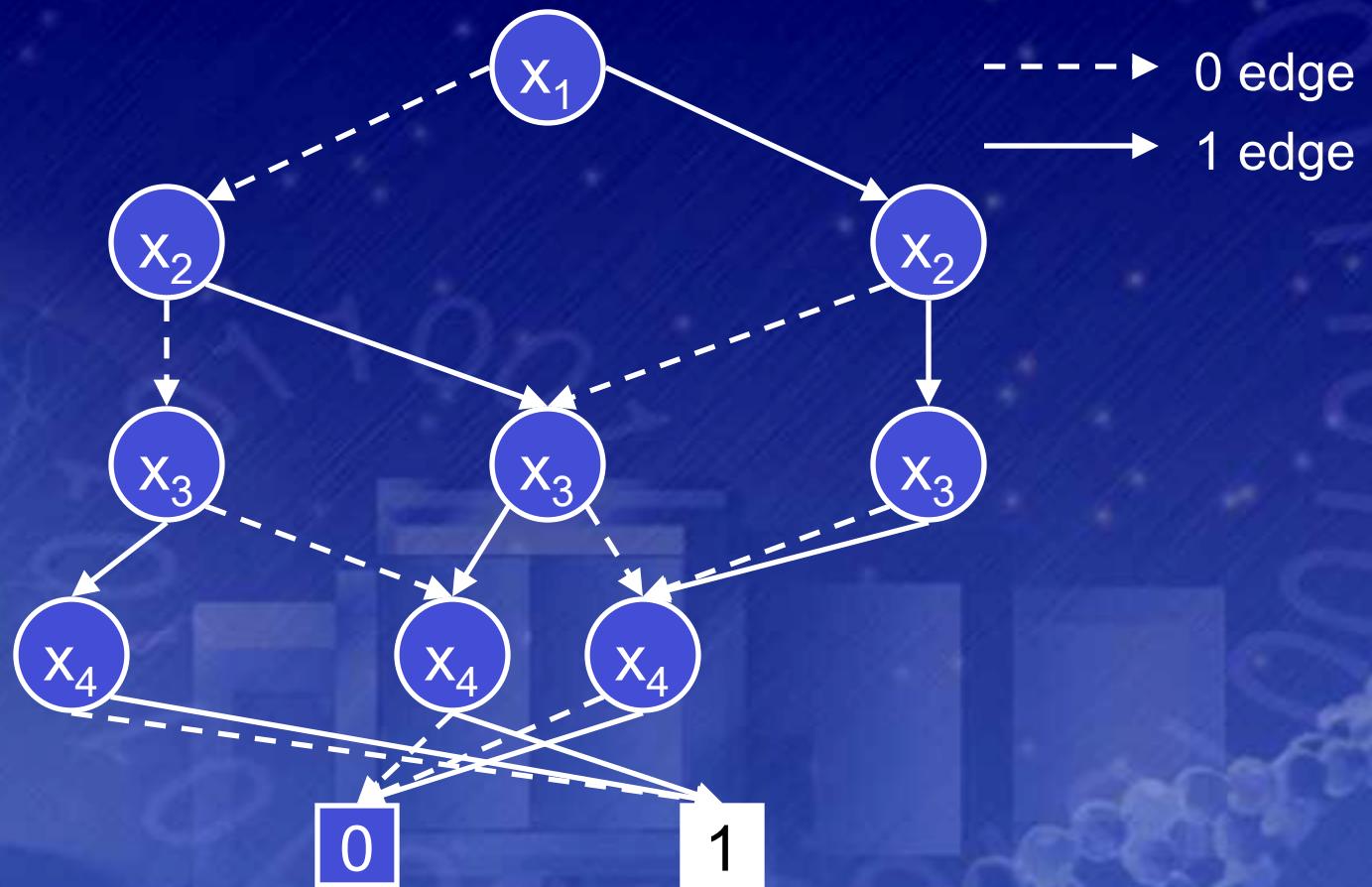
Binary Decision Diagrams

- Collapse redundant nodes.



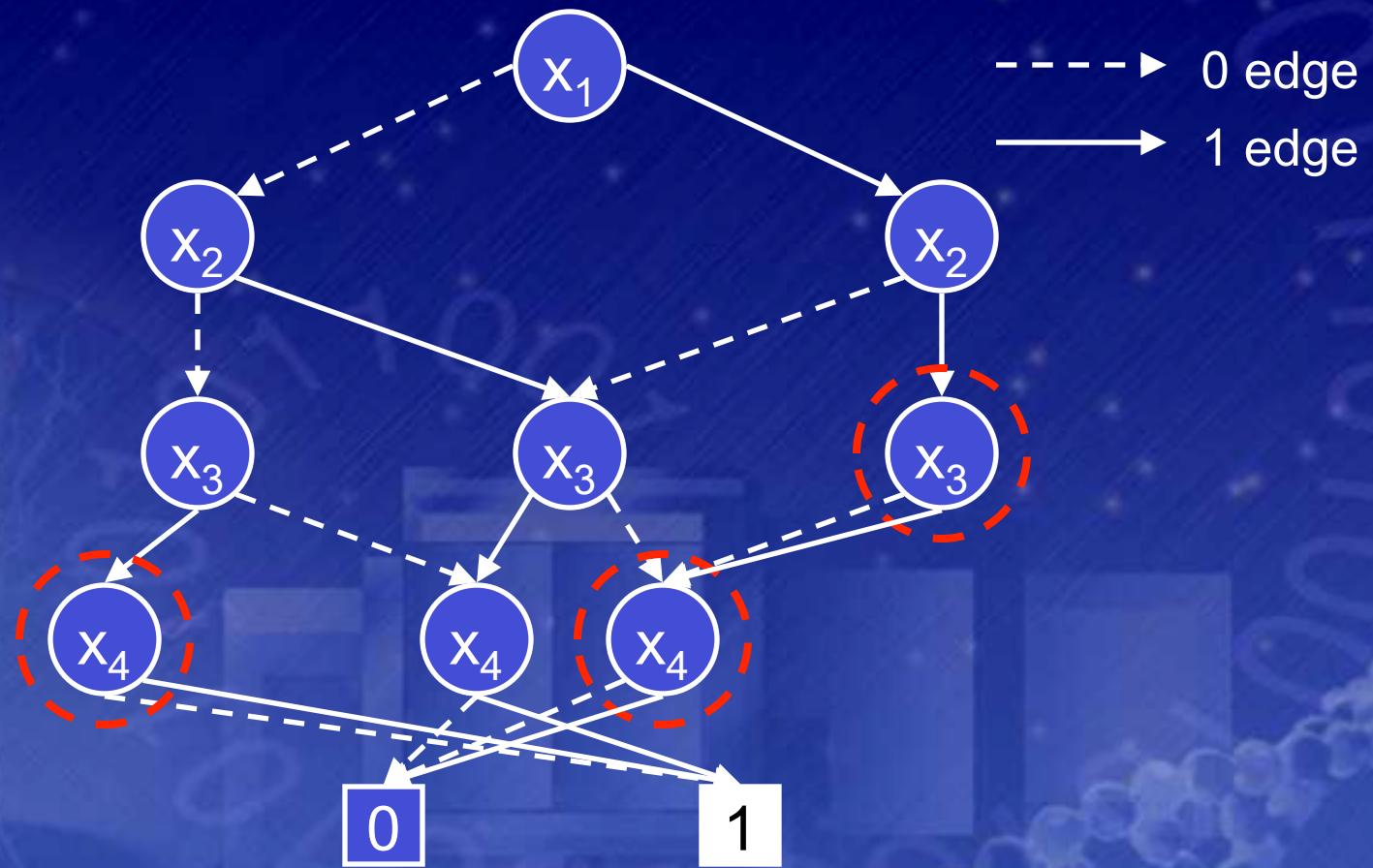
Binary Decision Diagrams

- Collapse redundant nodes.



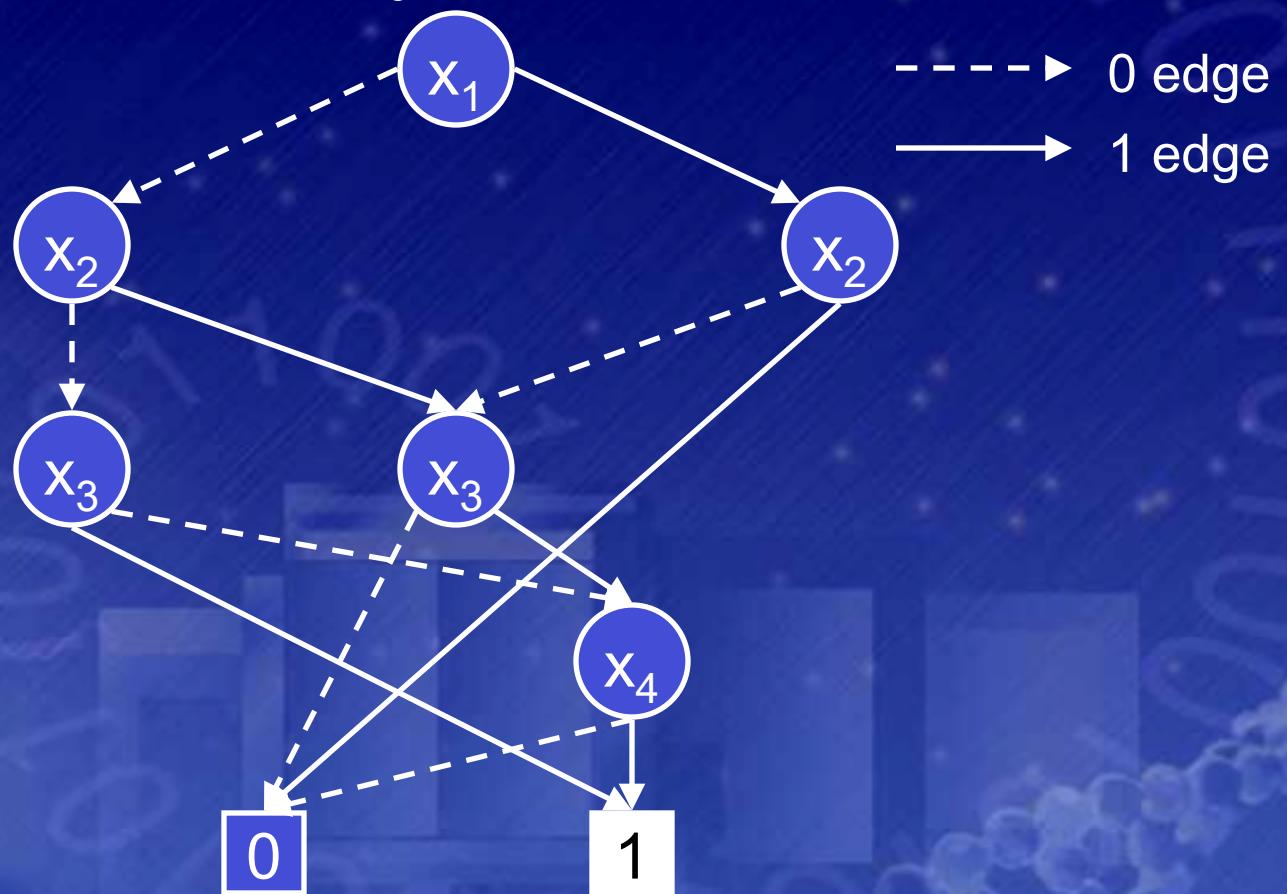
Binary Decision Diagrams

- Eliminate unnecessary nodes.



Binary Decision Diagrams

- Eliminate unnecessary nodes.

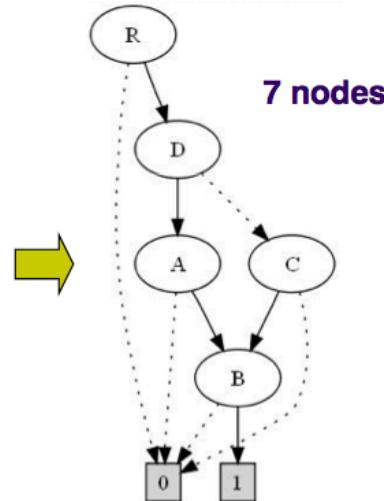
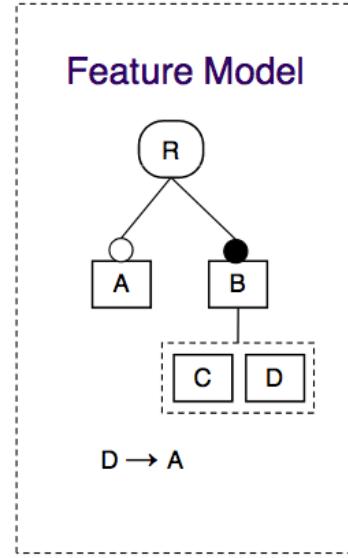
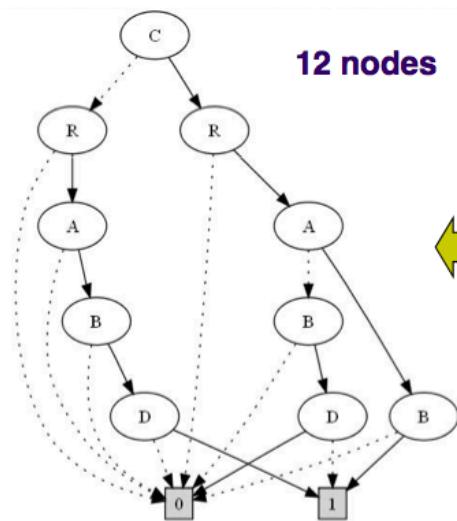


Binary Decision Diagrams (BDDs)

- Very efficient structure for most of the satisfiability operations
- Polynomial in time for checking satisfiability and determining equivalence between two BDDs
- Graph traversal
- So great?

Binary Decision Diagrams (BDDs): Theoretical Problem

- The size of the BDD is very sensitive to the order of the BDD variables
 - e.g. two equivalent BDDs for the same feature

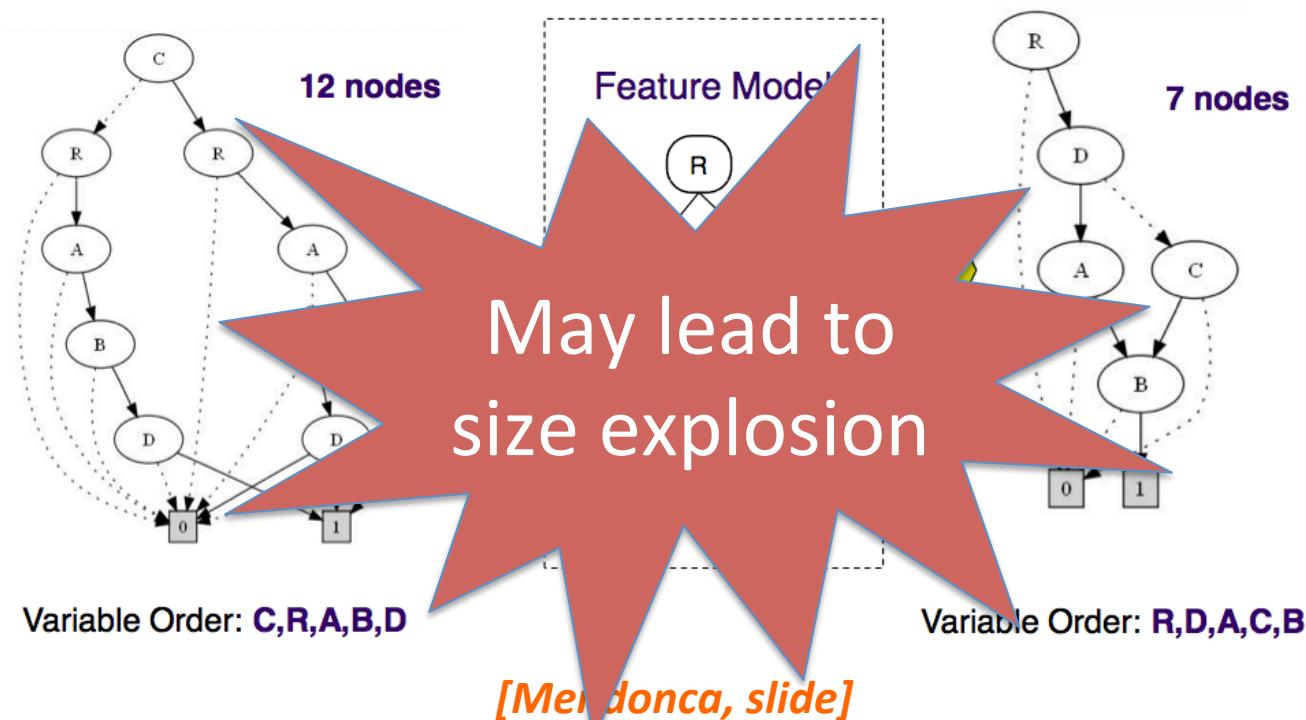


Variable Order: **C,R,A,B,D**

Variable Order: **R,D,A,C,B**

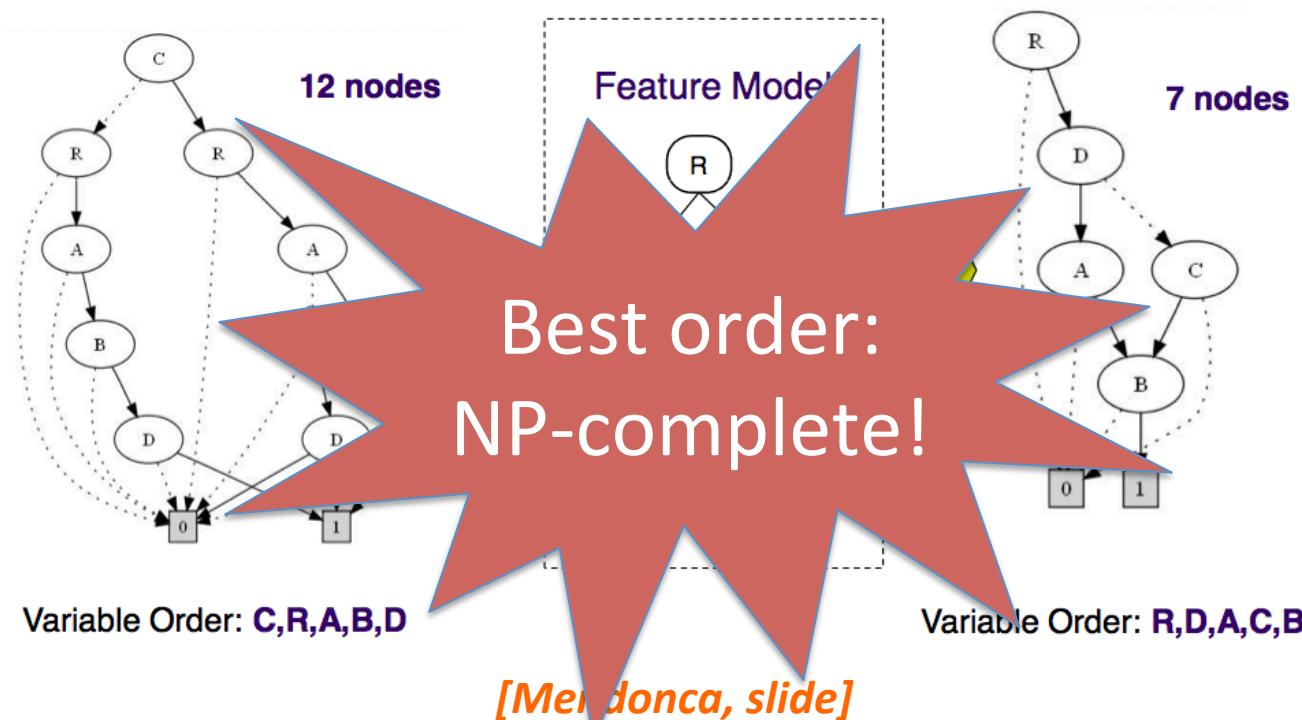
Binary Decision Diagrams (BDDs): Theoretical Problem

- The size of the BDD is very sensitive to the order of the BDD variables
 - e.g. two equivalent BDDs for the same feature



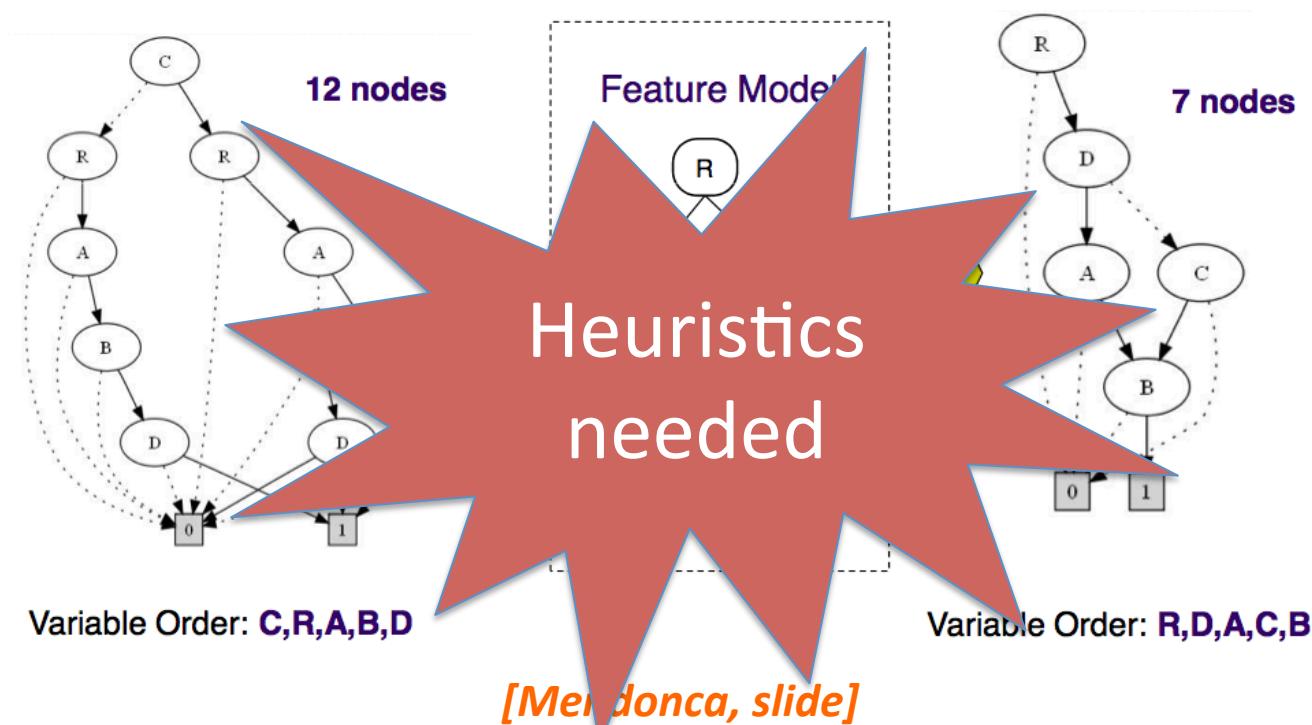
Binary Decision Diagrams (BDDs): Theoretical Problem

- The size of the BDD is very sensitive to the order of the BDD variables
 - e.g. two equivalent BDDs for the same feature



Binary Decision Diagrams (BDDs): Practical Problem

- The size of the BDD is very sensitive to the order of the BDD variables. In practice: **BDDs cannot be build for feature models with 2000+ features**

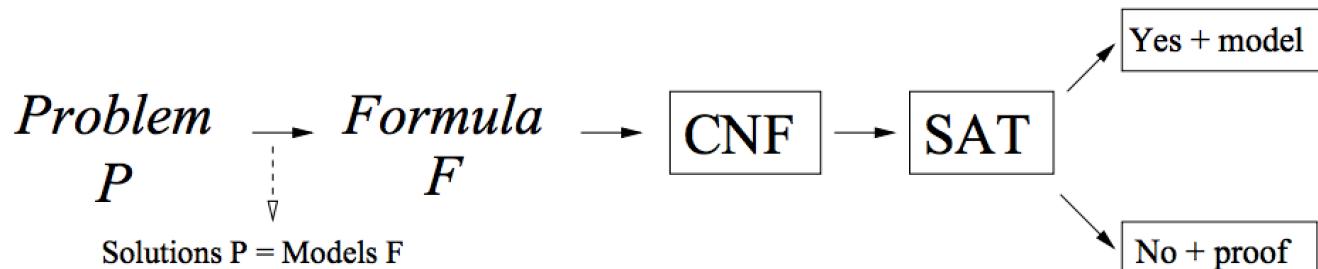


How to automate analysis of your feature models?

Let us try with SAT solvers

Satisfiability (SAT) solver

- A “SAT solver” is a program that automatically decides whether a propositional logic formula is satisfiable.
 - If it is satisfiable, a SAT solver will produce an example of a truth assignment that satisfies the formula.



- Basic idea: since all NP-complete problems are mutually reducible:
 - Write one really good solver for NP-complete problems (in fact, get lots of people to do it. Hold competitions.)
 - Translate your NP-complete problems to that problem.

SAT solver and CNF

- All current fast SAT solvers work on CNF
- Terminology:
 - A literal is a propositional variable or its negation (e.g., p or $\neg q$).
 - A clause is a disjunction of literals (e.g., $(p \vee \neg q \vee r)$). Since \vee is associative, we can represent clauses as lists of literals.
- A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses
 - e.g., $(p \vee q \vee \neg r) \wedge (\neg p \vee s \vee t \vee \neg u)$

SAT solver and Unit Propagation

- Whenever all the literals in a clause are false except one, the remaining literal must be true in any satisfying assignment (such a clause is called a **unit clause**).
 - Therefore, the algorithm can assign it to true immediately. After choosing a variable there are often many unit clauses.
 - Setting a literal in a unit clause often creates other unit clauses, leading to a cascade.

$$\{\neg p \vee q, \neg p \vee \neg q \vee r, p, \neg r\}.$$

$$\begin{array}{c|c} \begin{array}{l} \neg p \vee q \\ \neg p \vee \neg q \vee r \\ p \\ \neg r \end{array} & \begin{array}{l} q \\ \neg q \vee r \\ \neg r \end{array} \end{array}$$

- A good SAT solver often spends 80-90% of its time in unit propagation.

$$\begin{aligned}\mathcal{F}_{\text{unit}} := & (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \\ & (\neg x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5) \\ & (x_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)\end{aligned}$$

$$\begin{aligned}
\mathcal{F}_{\text{unit}} := & (\neg \textcolor{red}{x}_1 \vee \neg x_3 \vee x_4) \wedge (\neg \textcolor{red}{x}_1 \vee \neg x_2 \vee x_3) \\
& (\neg \textcolor{red}{x}_1 \vee x_2) \wedge (\textcolor{green}{x}_1 \vee x_3 \vee x_6) \wedge (\neg \textcolor{red}{x}_1 \vee x_4 \vee \neg x_5) \\
& (\textcolor{green}{x}_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)
\end{aligned}$$

$$\varphi = \{x_1=1\}$$

$$\begin{aligned}\mathcal{F}_{\text{unit}} := & (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \\ & (\neg x_1 \vee x_2) \wedge (\textcolor{green}{x_1} \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5) \\ & (\textcolor{green}{x_1} \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)\end{aligned}$$

$$\varphi = \{x_1=1, \textcolor{teal}{x_2=1}\}$$

$$\begin{aligned}\mathcal{F}_{\text{unit}} := & (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \\ & (\neg x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5) \\ & (x_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)\end{aligned}$$

$$\varphi = \{x_1=1, x_2=1, x_3=1\}$$

$$\begin{aligned}
\mathcal{F}_{\text{unit}} := & (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \\
& (\neg x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5) \\
& (x_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)
\end{aligned}$$

$$\varphi = \{x_1=1, x_2=1, x_3=1, x_4=1\}$$

SAT solver and Unit Propagation

BCP():

Repeatedly search for unit clauses, and
set unassigned literal to required value.

If a literal is assigned conflicting values, return F
else return T;

satisfy(ϕ) {

if every clause of ϕ has a true literal, return T;

if BCP() == F, return F;

assign appropriate values to all pure literals;

choose an $x \in V$ that is unassigned in A ,

and choose $v \in \{T, F\}$.

$A(x) = v$;

if satisfy(ϕ) return T;

$A(x) = \neg v$;

if satisfy(ϕ) return T;

unassign $A(x)$; // undo assignment for backtracking.

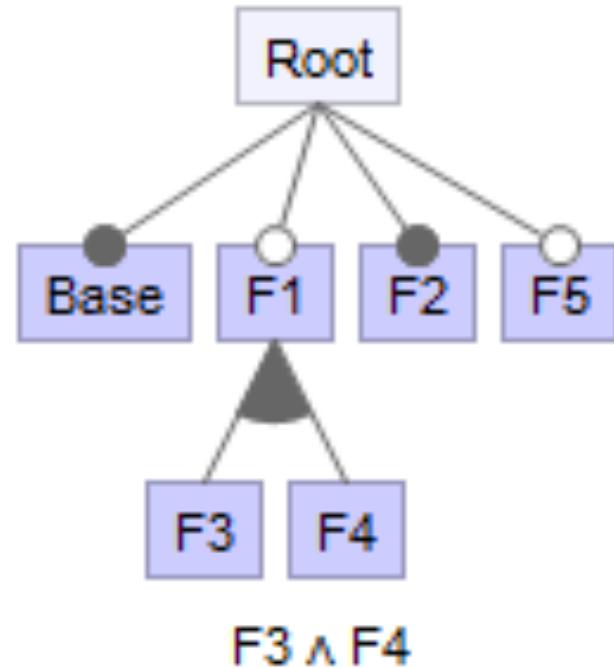
return F; }

How to automate analysis of your feature models?

Let us use BDDs and SAT solvers

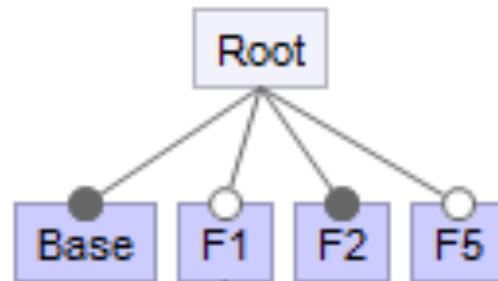
Consistency

- SAT-Solver
 - SAT(FM)



Core and dead features

- Dead : $\text{SAT}(\text{FM} \wedge F)$
- Core: $\text{SAT}(\text{FM} \wedge \text{not}(F))$



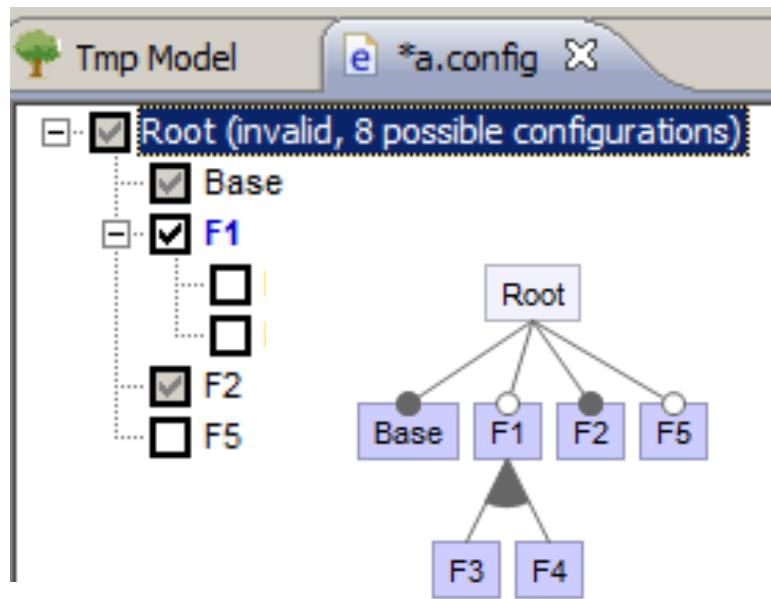
$$F5 \Rightarrow F4 \vee \text{Base}$$

$$F3 \Rightarrow F2 \wedge F5$$

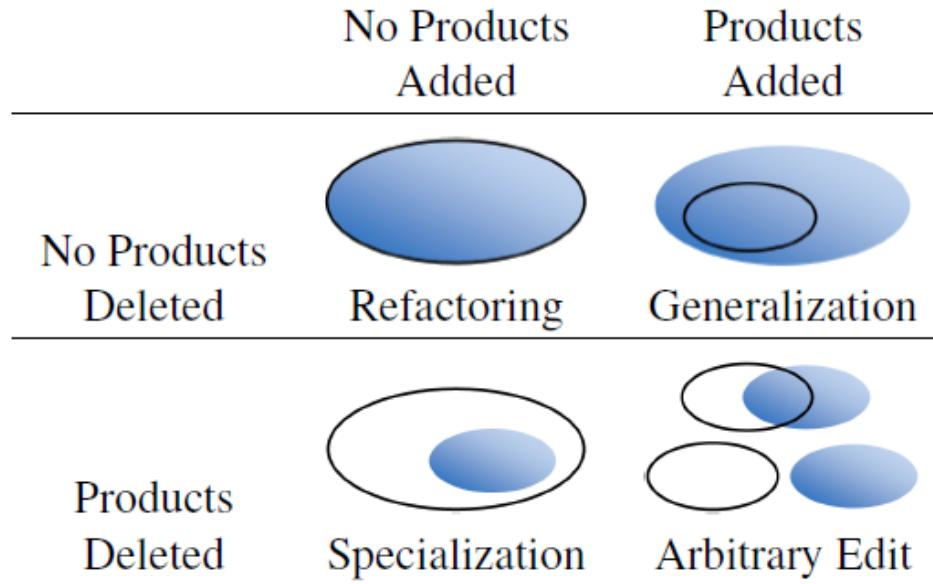
$$\neg(F4 \wedge F2)$$

Partial configuration

- $\text{SAT}(\text{FM} \wedge \text{PK} \wedge \text{F})$
- $\text{SAT}(\text{FM} \wedge \text{PK} \wedge \text{not(F)})$



Relationship between feature models



- Refactoring
 - Tautology: $(FM1 \Leftrightarrow FM2)$
 $= \text{not SAT}(\text{not } (FM1 \Leftrightarrow FM2))$

How can I use automated
analysis in practice?

Like this?

```
// Feature model path
String featureModelPath = "/Users/mathieuacher/Documents/workspaceScala/SPLAR/resources/models/simple_bike_fm.xml";

// Create feature model object from an XML file (SXF format - see www.splot-research.org for details)
// If an identifier is not provided for a feature use the feature name as id
FeatureModel featureModel = new XMLFeatureModel(featureModelPath, XMLFeatureModel.USE_VARIABLE_NAME_AS_ID);
// load feature model from
featureModel.loadModel();

// create BDD variable order heuristic
new FTPreOrderSortedECTraversalHeuristic("Pre-CL-MinSpan", featureModel, FTPreOrderSortedECTraversalHeuristic.FORCE_SORT);
VariableOrderingHeuristic heuristic = VariableOrderingHeuristicsManager.createHeuristicsManager().getHeuristic("Pre-CL-MinSpan");

// BDD construction parameters
// - Tuning this parameters can be tricky at times and may require playing a bit
// - For the purpose of this example let's assume "large enough" values
int bddNodeNum = 10000;      // sets the initial size of the BDD table
int bddCacheSize = 10000;    // sets the size of the BDD cache table

// Creates the BDD reasoner
ReasoningWithBDD reasoner = new FMReasoningWithBDD(featureModel, heuristic, 50000, 50000, 60000, "pre-order");

// Initialize the reasoner (BDD is created at this moment)
reasoner.init();

// Use the reasoner
System.out.println("BDD has " + reasoner.getBDD().nodeCount() + " nodes and was built in " + reasoner.getBDDBuildingTime() + " ms");

// Check if feature model is consistent, i.e., has at least one valid configuration
System.out.println("Feature model is " + (reasoner.isConsistent()? "" : " NOT ") + "consistent!");

// Count feature model solutions
System.out.println("Feature model has " + reasoner.countValidConfigurations() + " possible configurations");
```

No. Like this!

```
fml = FM ("fml.tvl") // many other formats
b1 = isValid fml // check consistency
n1 = counting fml // count the number of valid configurations
```

```
// Feature model path
String featureModelPath = "/Users/mathieuacher/Documents/workspaceScala/SPLAR/resources/models/simple_bike_fm.xml";

// Create feature model object from an XML file (SXF format - see www.splot-research.org for details)
// If an identifier is not provided for a feature use the feature name as id
FeatureModel featureModel = new XMLFeatureModel(featureModelPath, XMLFeatureModel.USE_VARIABLE_NAME_AS_ID);
// load feature model from
featureModel.loadModel();

// create BDD variable order heuristic
new FTPreOrderSortedECTraversalHeuristic("Pre-CL-MinSpan", featureModel, FTPreOrderSortedECTraversalHeuristic.FORCE_SORT);
VariableOrderingHeuristic heuristic = VariableOrderingHeuristicsManager.createHeuristicsManager().getHeuristic("Pre-CL-MinSpan");

// BDD construction parameters
// - Tuning this parameters can be tricky at times and may require playing a bit
// - For the purpose of this example let's assume "large enough" values
int bddNodeNum = 10000; // sets the initial size of the BDD table
int bddCacheSize = 10000; // sets the size of the BDD cache table

// Creates the BDD reasoner
ReasoningWithBDD reasoner = new FMReasoningWithBDD(featureModel, heuristic, 50000, 50000, 60000, "pre-order");

// Initialize the reasoner (BDD is created at this moment)
reasoner.init();

// Use the reasoner
System.out.println("BDD has " + reasoner.getBDD().nodeCount() + " nodes and was built in " + reasoner.getBDDBuildingTime() + " ms");

// Check if feature model is consistent, i.e., has at least one valid configuration
System.out.println("Feature model is " + (reasoner.isConsistent()? "" : " NOT ") + "consistent!");

// Count feature model solutions
System.out.println("Feature model has " + reasoner.countValidConfigurations() + " possible configurations");
```

FAMILIAR: domain-specific language

```
fml = FM ("fml.tvl") // many other formats
b1 = isValid fml // check consistency
n1 = counting fml // count the number of valid configurations
```

more concise and readable
no need to learn a (set of) API
interoperability facilities

```
// Feature model path
String featureModelPath = "/Users/mathieuacher/Documents/workspaceScala/SPLAR/resources/models/simple_bike_fm.xml";
// Create feature model object from XML file (SXF format - see www.splot-research.org for details)
// If feature identifiers in the feature model do not use the feature name as id
FeatureModel featureModel = new XMLFeatureModel(featureModelPath, XMLFeatureModel.USE_VARIABLE_NAME_AS_ID);
// load feature model from
featureModel.loadModel();

// create BDD variable order heuristic
new FPTreePreOrderSortedECTraversalHeuristic("Pre-CL-MinSpan", featureModel, FPTreePreOrderSortedECTraversalHeuristic.FORCE_SORT);
VariableOrderingHeuristic heuristic = VariableOrderingHeuristicsManager.createHeuristicsManager().getHeuristic("Pre-CL-MinSpan");

// BDD construction parameters
// - Tuning this parameters can be tricky at times and may require playing a bit
// - For the purpose of this example let's assume "large enough" values
int bddNodeNum = 10000;      // sets the initial size of the BDD table
int bddCacheSize = 10000;    // sets the size of the BDD cache table

// Creates the BDD reasoner
ReasoningWithBDD reasoner = new FMReasoningWithBDD(featureModel, heuristic, 50000, 50000, 60000, "pre-order");

// Initialize the reasoner (BDD is created at this moment)
reasoner.init();

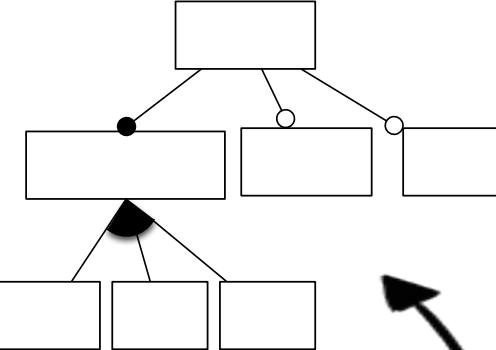
// Use the reasoner
System.out.println("BDD has " + reasoner.getBDD().nodeCount() + " nodes and was built in " + reasoner.getBDDBuildingTime() + " ms");

// Check if feature model is consistent, i.e., has at least one valid configuration
System.out.println("Feature model is " + (reasoner.isConsistent()?"": " NOT ") + "consistent!");

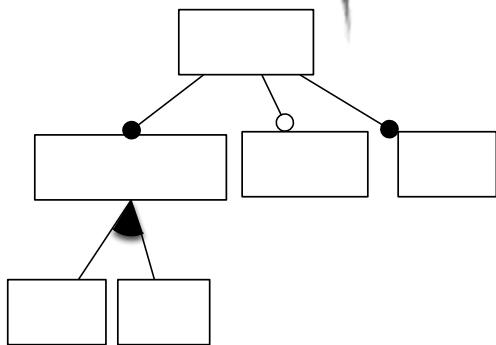
// Count feature model solutions
System.out.println("Feature model has " + reasoner.countValidConfigurations() + " possible configurations");
```

Managing (multiple) feature models

Multiple Feature Models



not, and, or, implies



The image is a composite of two screenshots. The top half shows a Renault UK website's vehicle configuration tool for the New Kangoo Van Range. It features a central character illustration, navigation links like 'CARS', 'VANS', 'ELECTRIC VEHICLES', etc., and a large red diagonal watermark reading 'Variability and Configuration'. The configuration interface shows various options like 'Central storage console & a seats' (checked), 'Electric door mirrors' (unchecked), and 'ESC (Electronic Stability Control) with traction and understeer control' (checked). The bottom half shows a Java code editor with several files open, including 'Notepad.java', 'Actions.java', and 'Main.java'. The code is related to a graphical user interface, with parts of 'Actions.java' and 'Main.java' visible. The Java code includes comments and variable names like 'output', 'program', 'equation', 'hoa', 't', 'advice', 'intro', and 'g'. The Java interface also has its own red diagonal watermark.



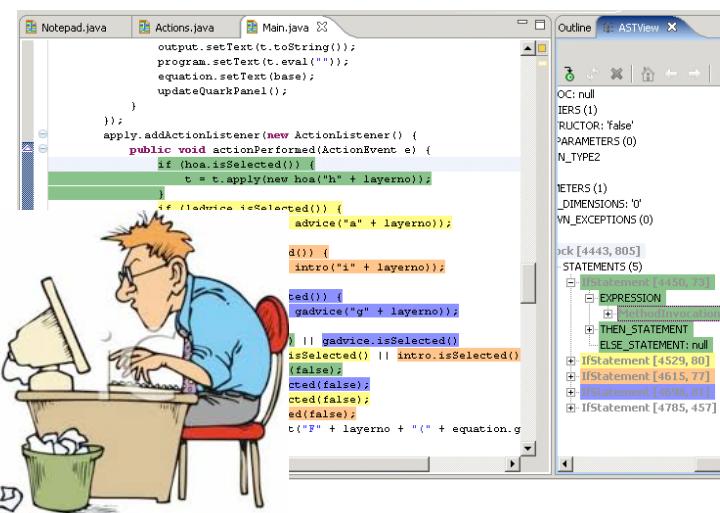
RENAULT VANS

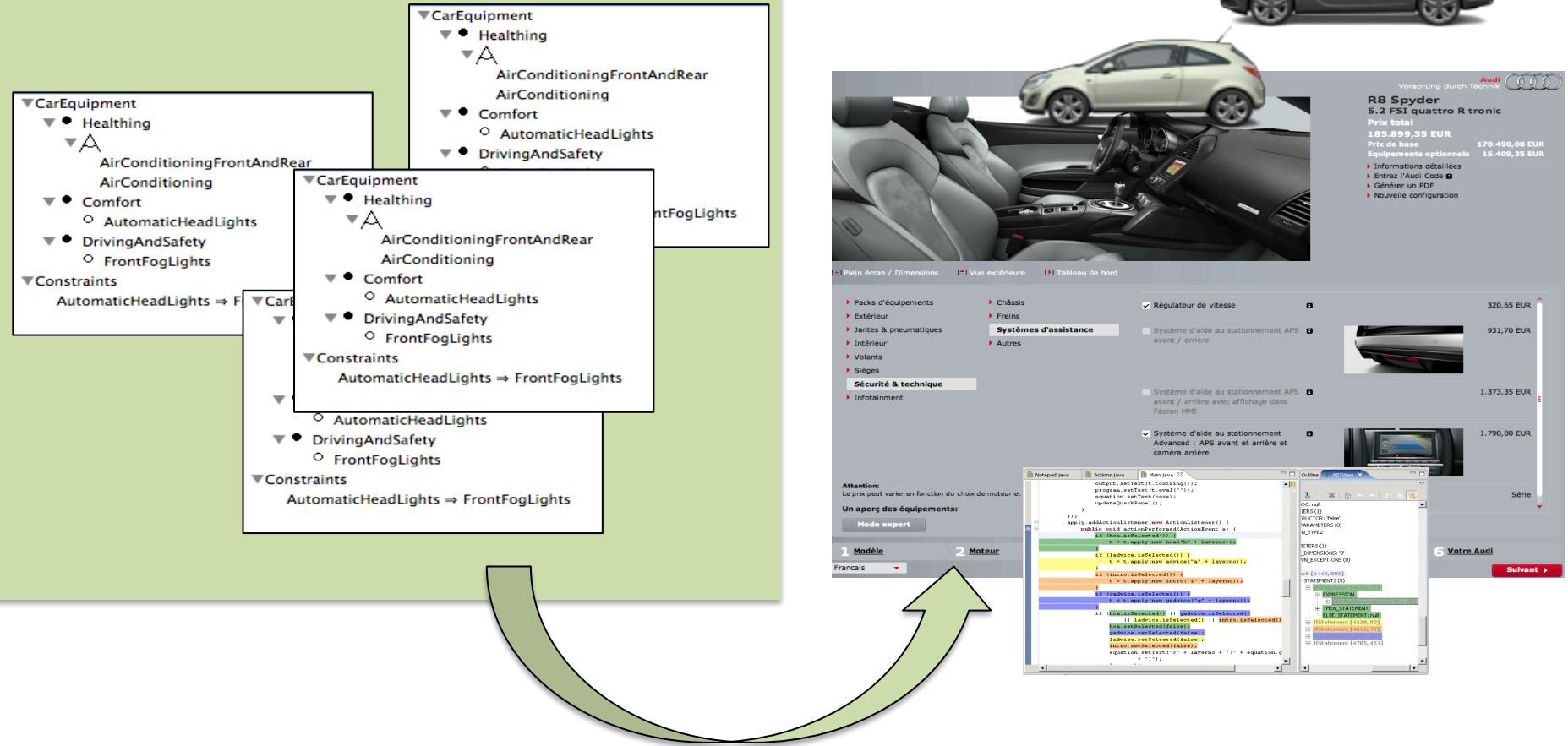


CARS | VANS | ELECTRIC VEHICLES | RENAULT BUSINESS | USED CARS | OWNER SERVICES | ABOUT RENAULT | RENAULT SHOP

Renault UK > Renault Vans > New Kangoo Van Range > Kangoo Van > Build your own Kangoo Van > Select Options

NEW KANGOO VAN RANGE

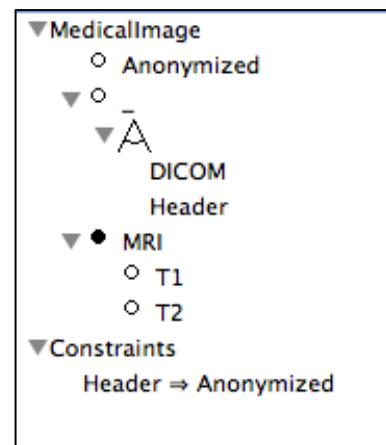
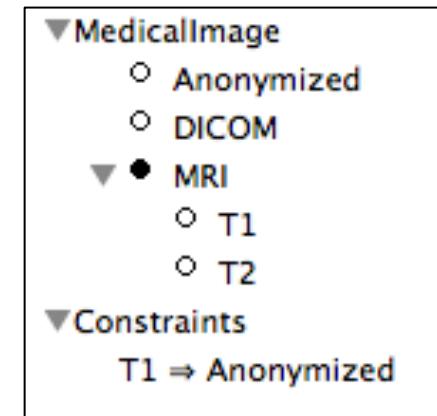
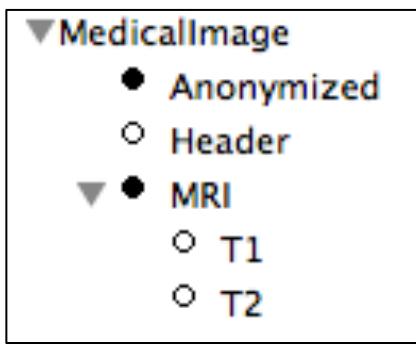
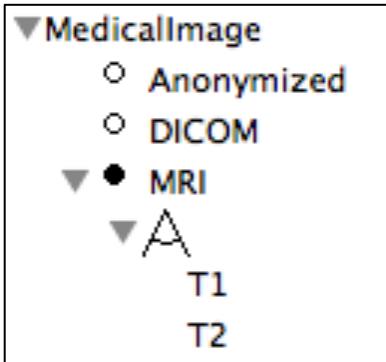




Composition/Decomposition for managing large, complex and multiple feature models

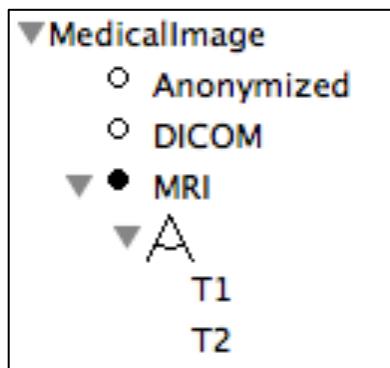
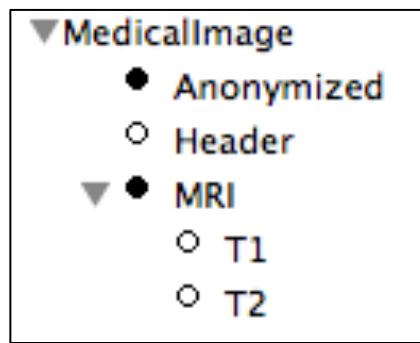
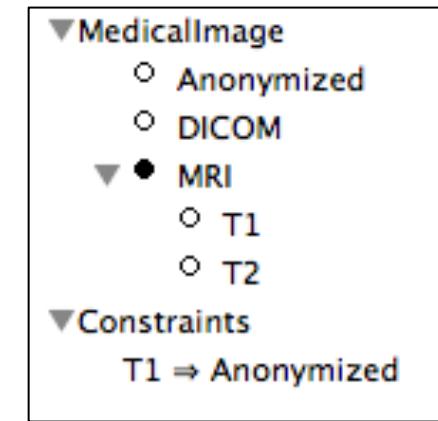
FORM 1998, Tun et al. 2009 (SPLC), Hartmann 2008 (SPLC), Lee et al. 2010, Czarnecki 2005, Reiser et al. 2007 (RE journal), Hartmann et al. 2009 (SPLC), Thuem et al. 2009 (ICSE), Classen et al. 2009 (SPLC), Mendonca et al. 2010 (SCP), Dunghana et al. 2010, Hubaux et al. 2011 (SoSyM), Zaid et al. 2010 (ER), She et al., 2011 (ICSE), etc.

Composition (Merging)



Offering: union of sets of configurations

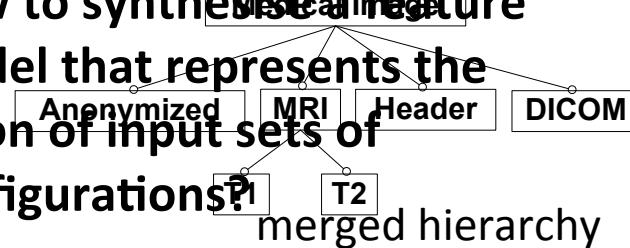
Merging operation: algorithm


 Φ_1

 Φ_2

 Φ_3
 Φ_{123}

merged propositional formula

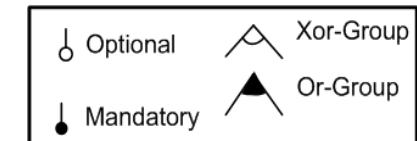
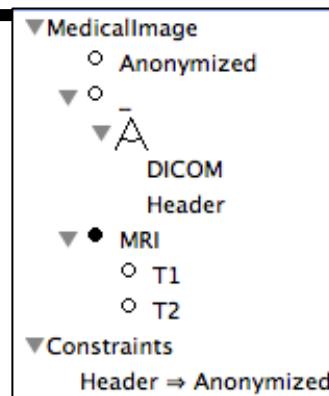


How to synthesise a feature model that represents the union of input sets of configurations?



merged hierarchy

Set mandatory features
Detect Xor and Or-groups
Compute “implies/excludes” constraints



Building “views” of a feature model

```

not Audi car
group all{
    ModelLine;
    Model;
    ModelType;
    Model;
    Exterior;
    Interior;
    Equipment;
}
3

ModelLine {
    group all{
        AudiA1 {
            AudiA1 {
                AudiA1 -> Q1 || AllSportback || A3Cabriolet || S3 || S3Sportback || RS3Sportback;
                AudiA1 -> Q3 || Sportback || Cabriolet;
            }
            AudiA1
        }
        AudiA3 {
            AudiA3 -> Q2 || AllSportback || A3Cabriolet || S3 || S3Sportback || RS3Sportback;
            AudiA3 -> Q3 || Sportback || Cabriolet;
        }
        AudiA4 {
            AudiA4 -> A4Allroad || A4Avant || AllroadQuattro || S4Saloon || S4Avant;
            AudiA4 -> Q5Crossover || Avant || AllroadQuattro;
        }
        AudiA4
        AudiA5 {
            AudiA5 -> Q5Coupé || AllSportback || A5Cabriolet || S5Coupé || S5Sportback || S5Cabriolet || RS5Coupé;
            AudiA5 -> Q5Sportback || Coupé || Cabriolet;
        }
        AudiA6 {
            AudiA6 -> Q7 || All || A6Allroad;
            AudiA6 -> Q7Crossover || Avant;
        }
        AudiA7 {
            AudiA7 -> Q7Coupé || Sportback;
            AudiA7 -> Cabriolet;
        }
        AudiA8 {
            AudiA8 -> A8 || A8Allroad;
            AudiA8 -> A8Cabriolet;
        }
        AudiQ5 {
            AudiQ5 -> Q5;
            AudiQ5 -> Q5SUV;
            AudiQ5 -> Q5Coupé;
        }
        AudiQ7 {
            AudiQ7 -> Q7;
            AudiQ7 -> Q7SUV;
            AudiQ7 -> Q7Coupé;
        }
        AudiRS {
            AudiRS -> RS3;
            AudiRS -> RS4;
        }
        AudiTT {
            AudiTT -> TT;
            AudiTT -> TTS;
            AudiTT -> TTSRoadster || TTSCoupe || TTSRoadster || TTRoadster;
            AudiTT -> TTS || Roadster;
        }
        AudiRS
        AudiRS {
            AudiRS -> RS3Coupé || RS3Sportback;
            AudiRS -> Coupé || Spyder;
        }
    }
}

bodyStyle {
    group all{
        sedan {
            Saloon -> A4Saloon || S4Saloon || A6Saloon || A8 || A8Allroad;
        };
        Avant {
            Avant -> A4Avant || S4Avant || A6Avant;
        };
        Sportback {
            Sportback -> A3 || A3 || S3;
            Sportback -> A3Sportback || AllSportback || S3Sportback || RS3Sportback || RS3Sportback || AllSportback;
        };
        Coupe {
            Coupe -> A4Coupé || S4Coupé || TTSCoupe || TTSCoupe || TRSCoupe || R8Coupé;
        };
        Cabriolet {
            Cabriolet -> A4Cabriolet || A5Cabriolet || S5Cabriolet;
        };
        Roadster {
            Roadster -> TTRoadster || TTSCabriolet || TTSRoadster;
        };
        Spyder {
        };
    }
}

```

The screenshot shows a grid of Audi vehicles. In the top row, there are three sedans: A1 (red), A3 (black), and A4 (silver). In the middle row, there are two SUVs: A8 (black) and Q5 (white). In the bottom row, there are two coupés: Q3 (black) and TT (white). To the right of the cars, the Audi logo and the slogan "Vorsprung durch Technik" are visible.

Model **Engine** **Exterior** **Interior** **Equipment** **Your Audi**

Sitemap Terms of Use

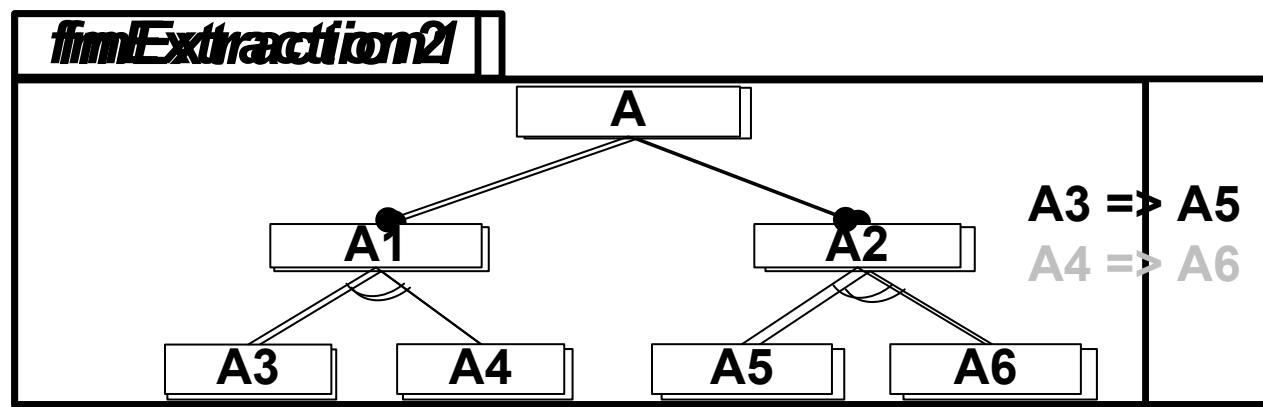
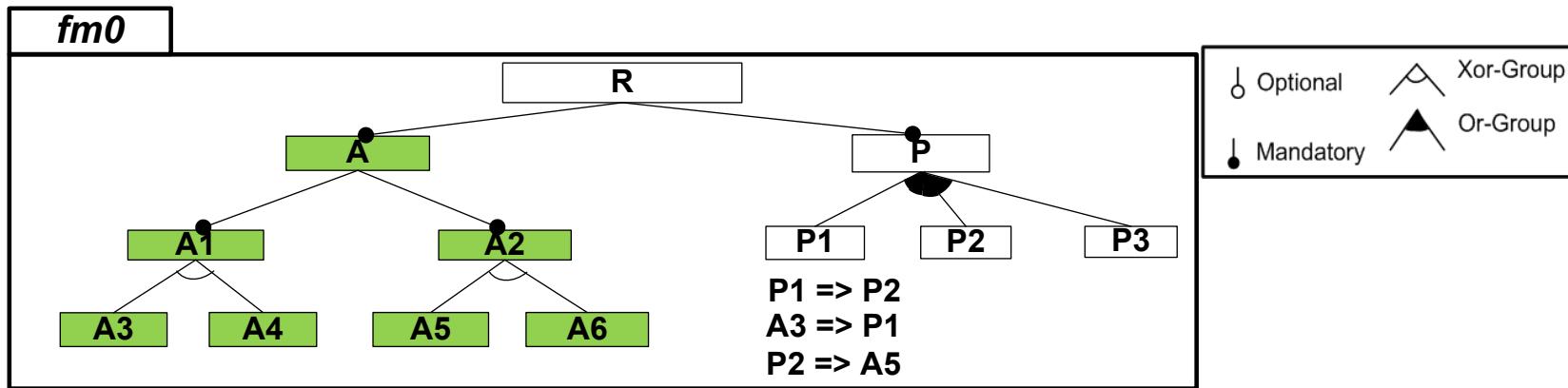
1 Model 2 Engine 3 Exterior 4 Interior 5 Equipment 6 Your Audi

Next >

Building “views” of a feature model

- Problem: given a feature model, how to decompose it into smaller feature models?
- Semantics?
 - What’s the hierarchy
 - What’s the set of configurations?

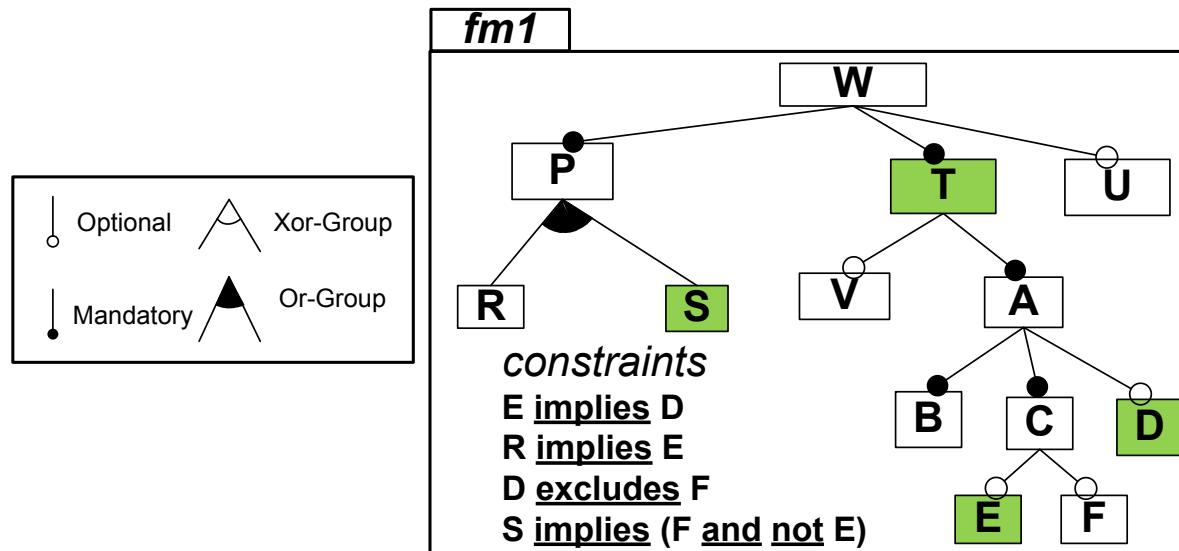
A first try



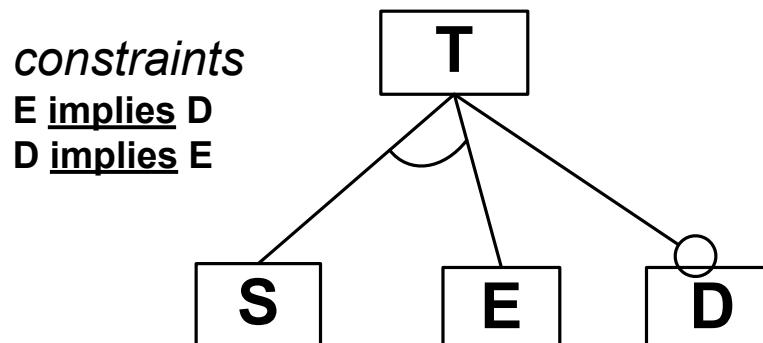
Problem: You can select **A3** without **A5**

Slicing Operator

slicing criterion : an arbitrary set of features, relevant for a feature model user

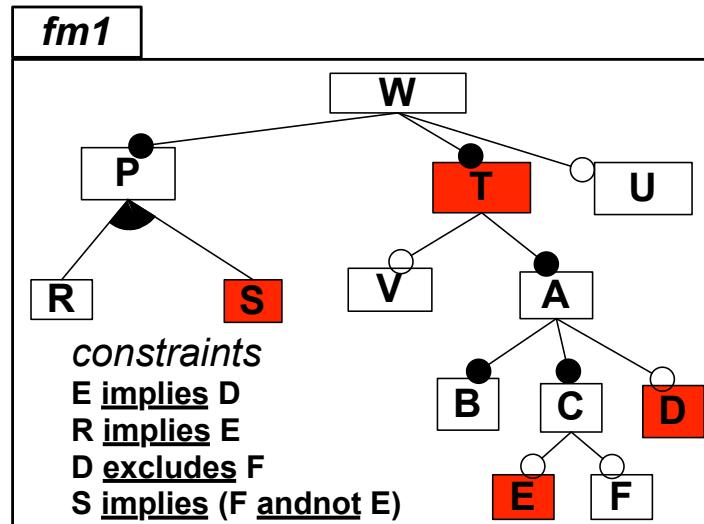
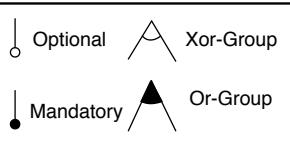


slice : a new feature model, representing a projected set of configurations



Slicing operator: going into details

projected set of configurations

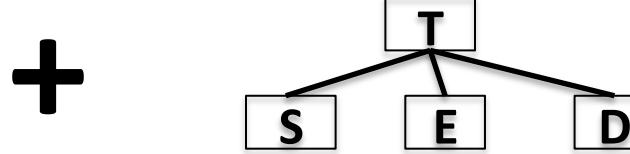
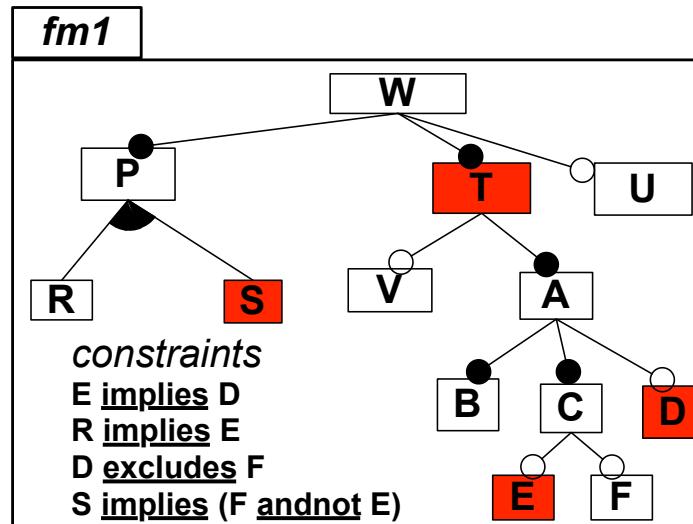
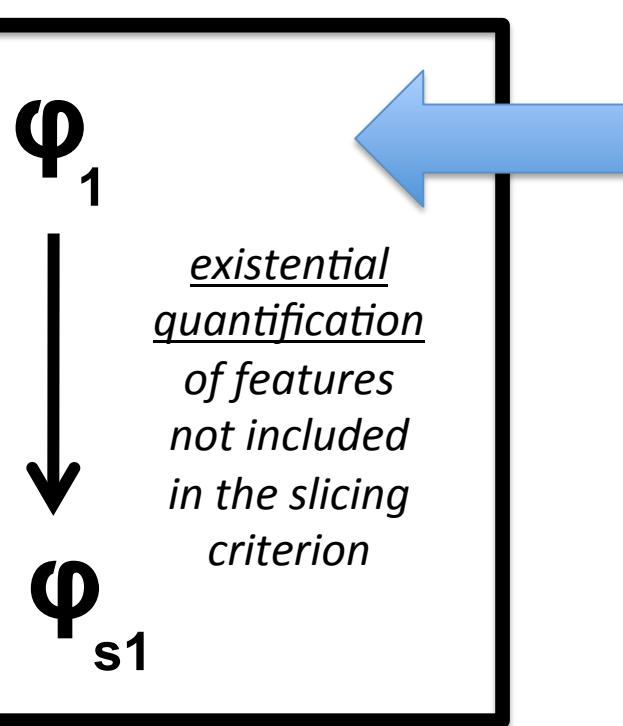


```
fm1 != {  
  {A,B,C,D,E,R,T,U},  
  {A,B,C,E,P,S,T,U},  
  {A,B,C,D,E,R,T,U},  
  {A,B,C,E,P,S,T,U},  
  {A,B,C,E,P,S,T,U},  
  {A,B,C,E,P,S,T,U},  
  {A,B,C,D,E,R,T,U},  
  {A,B,C,E,P,S,T,U},  
}
```

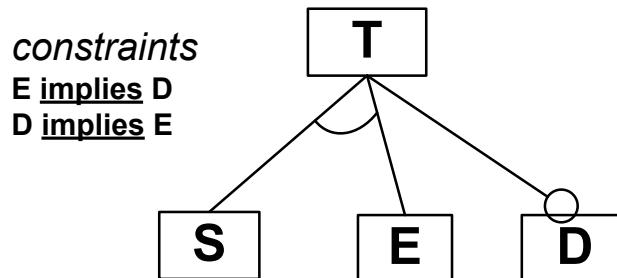
```
fm1p = {  
  {D,E,T},  
  {S,T},  
  {B,E,T},  
  {S,T},  
  {S,T},  
  {S,T},  
  {D,E,T}  
}
```

Slicing operator: going into details

synthesizing the corresponding feature model



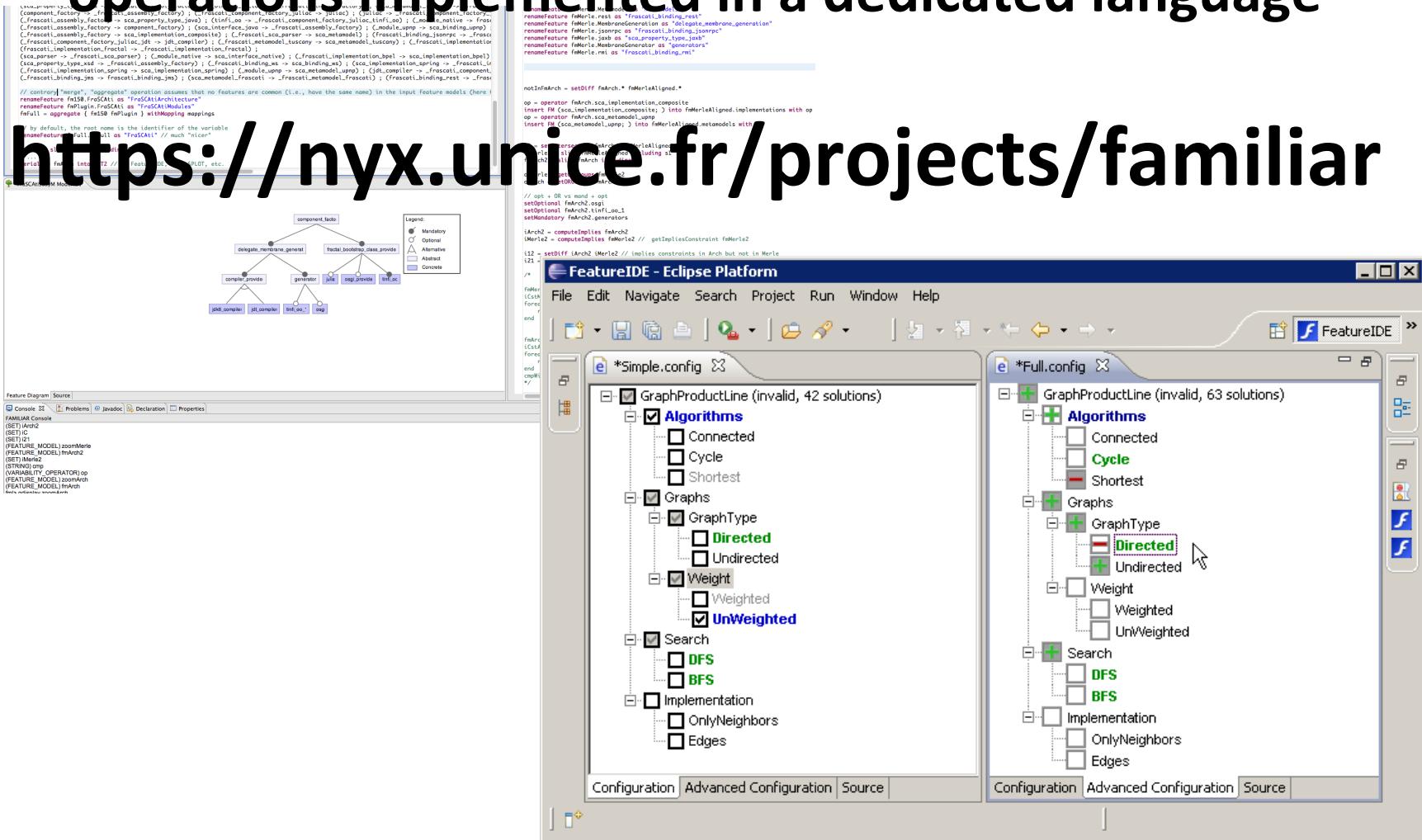
```
fm1p = {  
  {D,E,T},  
  {S,T}  
}
```



FAMILIAR language and environment

Reasoning/composition/decomposition.....

operations implemented in a dedicated language



Feature models in practice

Feature model in the SPL framework

- Not restricted to document/communicate
- Product/model derivation
 - reminder of previous course
- Safe composition and automated reasoning
- Generation of Configurators

Feature Models

+

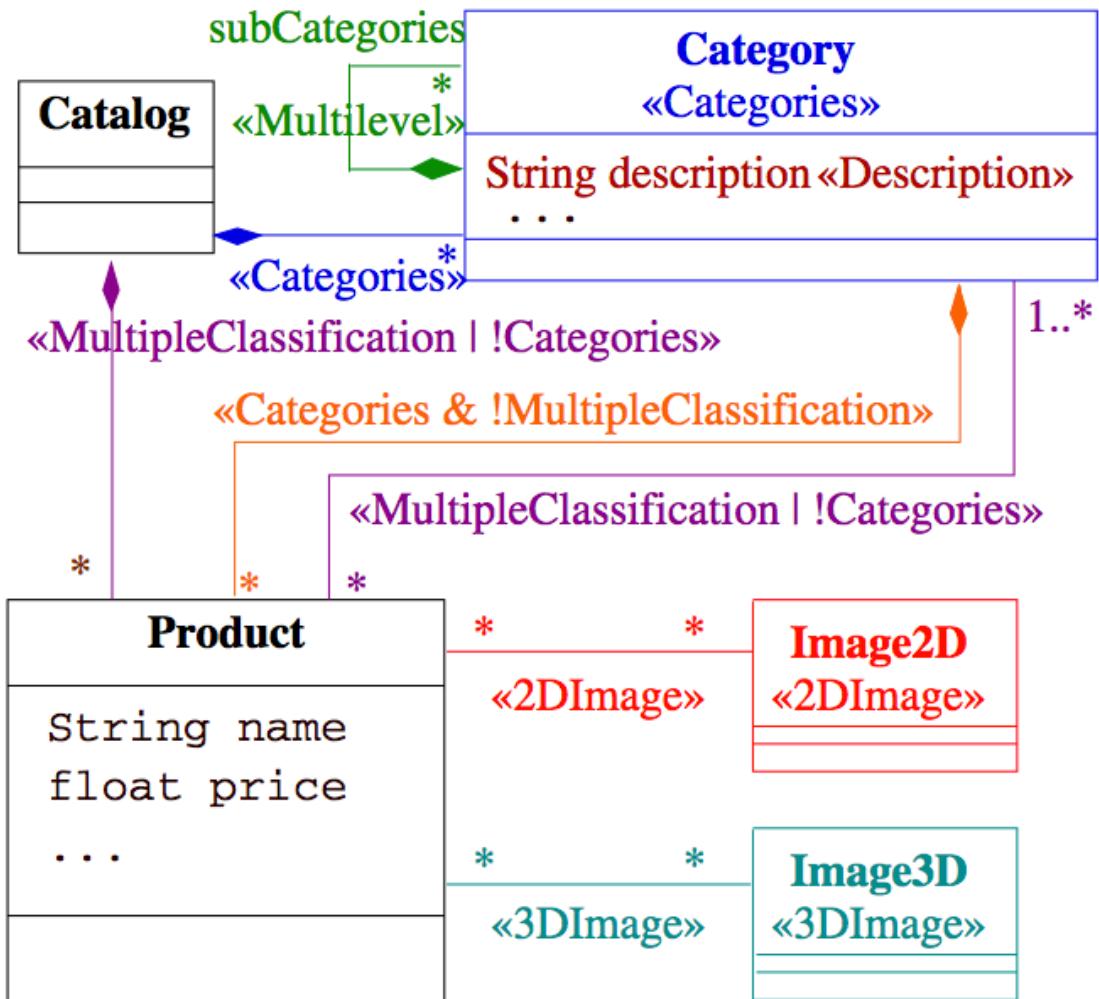
other artefacts

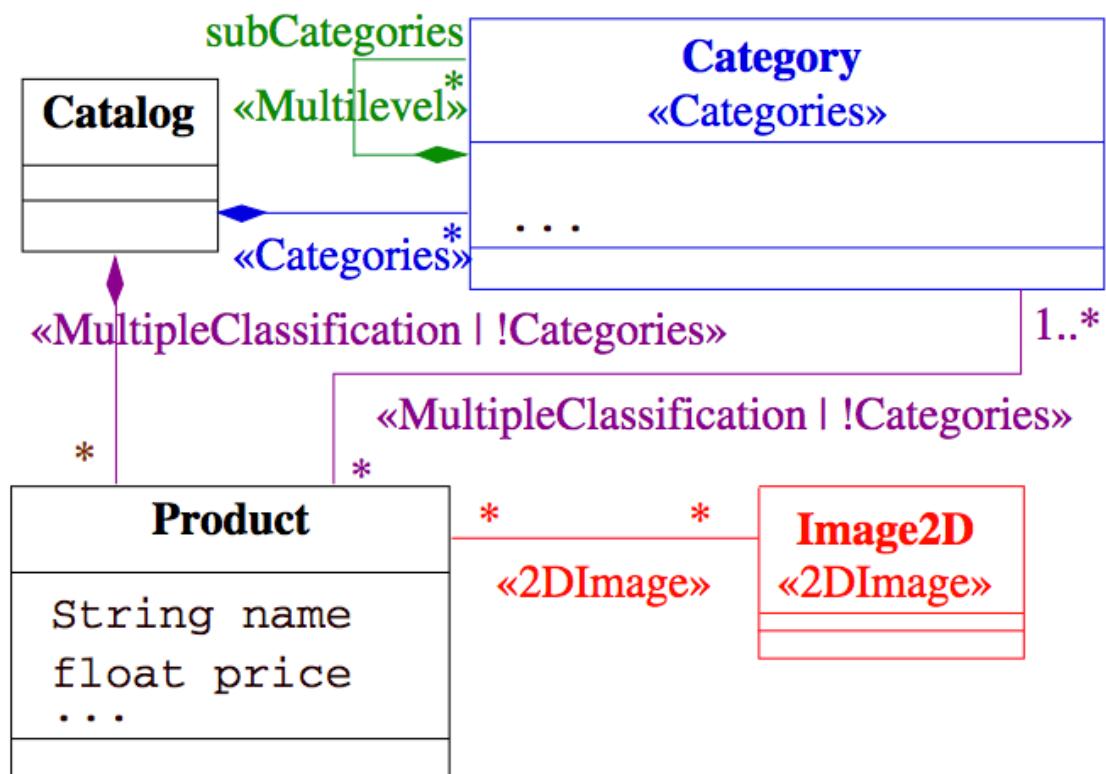
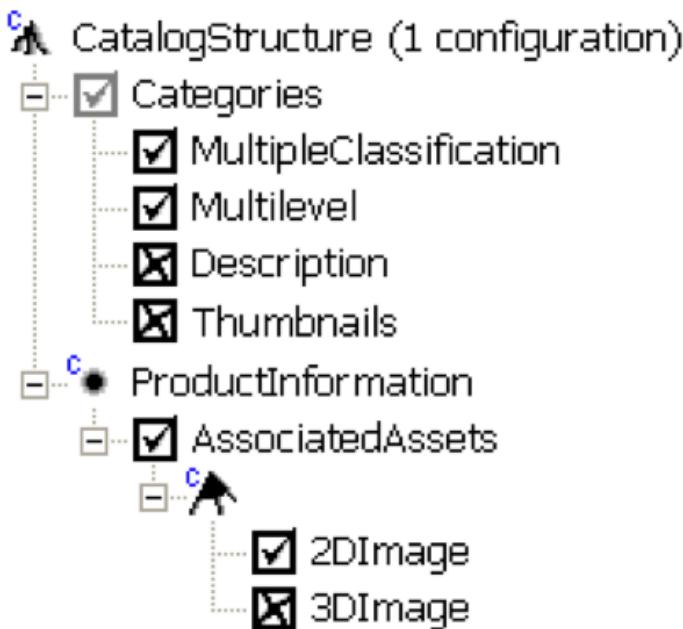
(what we can do with feature models)

**“features” in feature models:
a label**

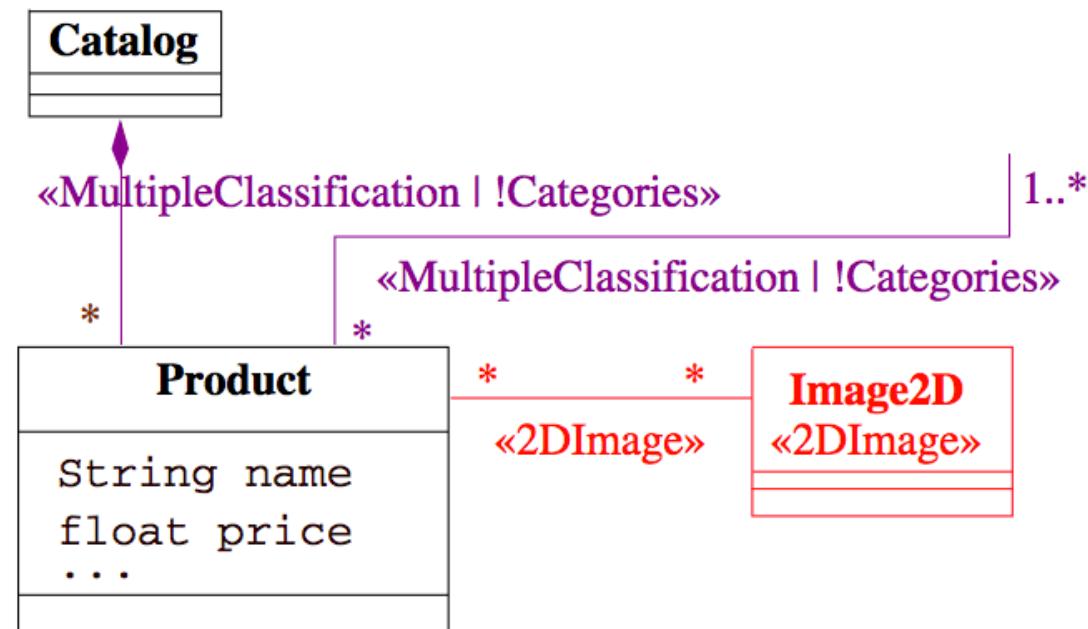
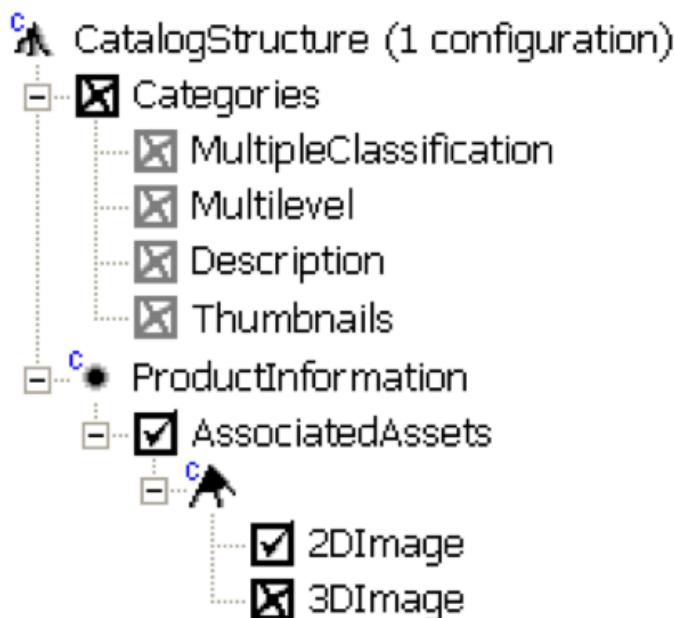
▲ CatalogStructure (52 configurations)

- Categories
 - MultipleClassification
 - Multilevel
 - Description
 - Thumbnails
- ProductInformation
 - AssociatedAssets
 - 2DImage
 - 3DImage



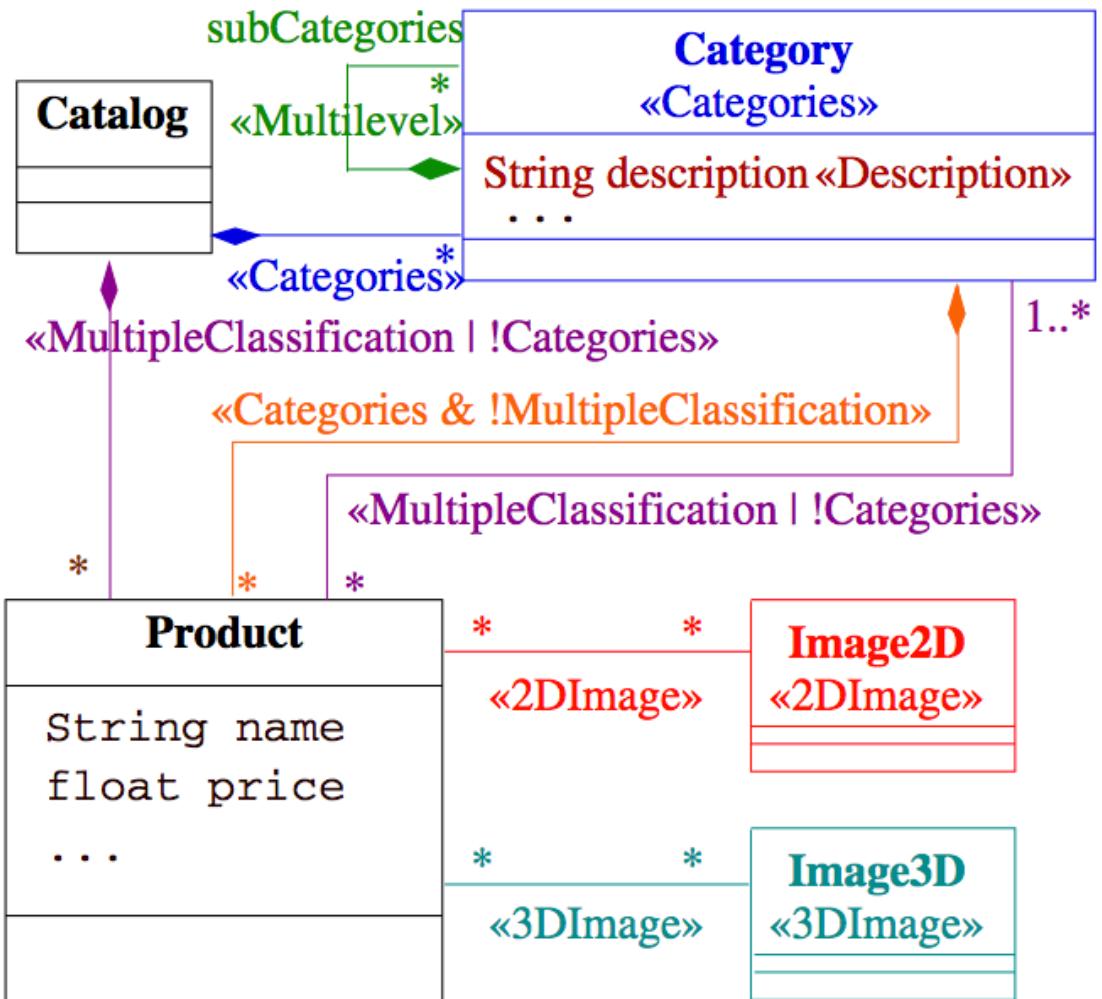
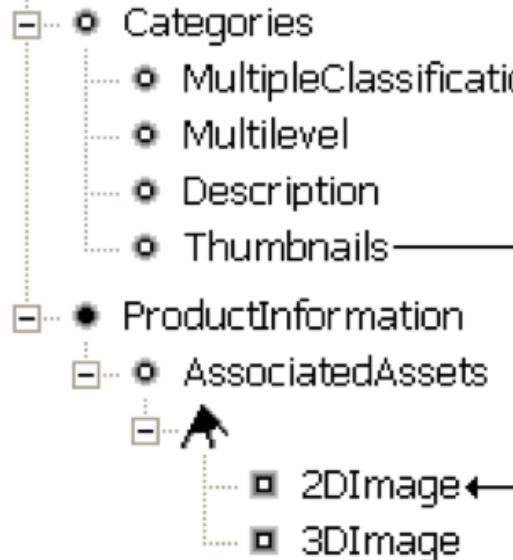


Ooops

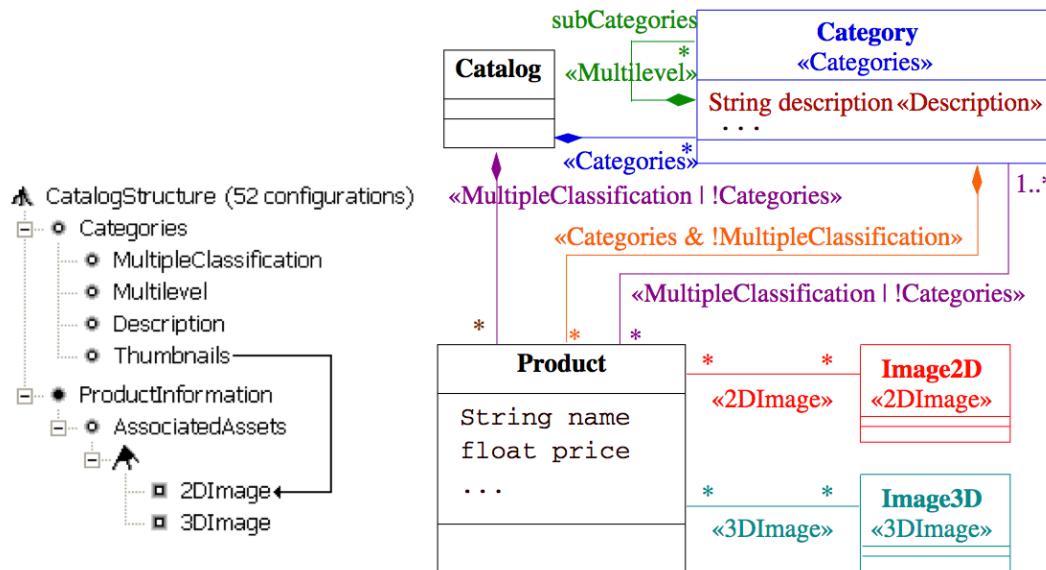


Safe composition? No!

CatalogStructure (52 configurations)



Safe composition: how does it work?



qFM

root:

child-parent:

$$= cs \wedge (ct \Rightarrow cs) \wedge (mc \Rightarrow ct) \wedge (ml \Rightarrow ct) \wedge (ds \Rightarrow ct) \wedge (tn \Rightarrow ct) \wedge (pi \Rightarrow cs) \wedge (aa \Rightarrow pi) \wedge (i2 \Rightarrow aa) \wedge (i3 \Rightarrow aa) \wedge (aa \Rightarrow choice_{1,2}(i2, i3)) \wedge (cs \Rightarrow pi) \wedge (tn \Rightarrow i2)$$

group:

mandatory:

additional:



Another propositional formula

Mapping: an example

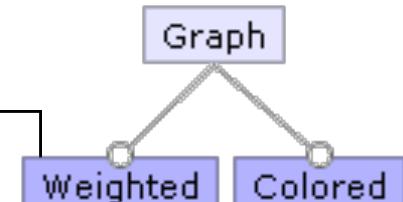
```
class Graph {  
    Vector nv = new Vector(); Vector ev = new Vector();  
    Edge add(Node n, Node m) {  
        Edge e = new Edge(n, m);  
        nv.add(n); nv.add(m); ev.add(e);  
        e.weight = new Weight();  
        return e;  
    }  
    Edge add(Node n, Node m, Weight w)  
    Edge e = new Edge(n, m);  
    nv.add(n); nv.add(m); ev.add(e);  
    e.weight = w; return e;  
}  
void print() {  
    for(int i = 0; i < ev.size(); i++) {  
        ((Edge)ev.get(i)).print();  
    }  
}
```

```
class Node {  
    int id = 0;  
    Color color = new Color();  
    void print() {  
        Color.setDisplayColor(color);  
        System.out.print(id);  
    }  
}
```

```
class Edge {  
    Node a, b;  
    Color color = new Color();  
    Weight weight = new Weight();  
    Edge(Node _a, Node _b) { a = _a; b = _b; }  
    void print() {  
        Color.setDisplayColor(color);  
        a.print(); b.print();  
        weight.print();  
    }  
}
```

```
class Color {  
    static void setDisplayColor(Color c) { ... }  
}
```

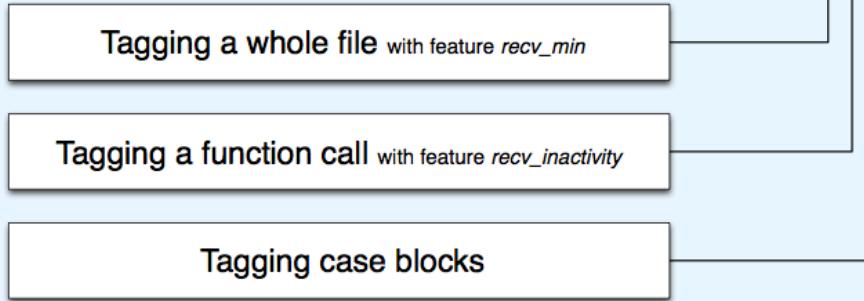
```
class Weight { void print() { ... } }
```



Tag and prune

Tag: annotate blocks of code with *features*

Prune: remove blocks tagged with non-selected features

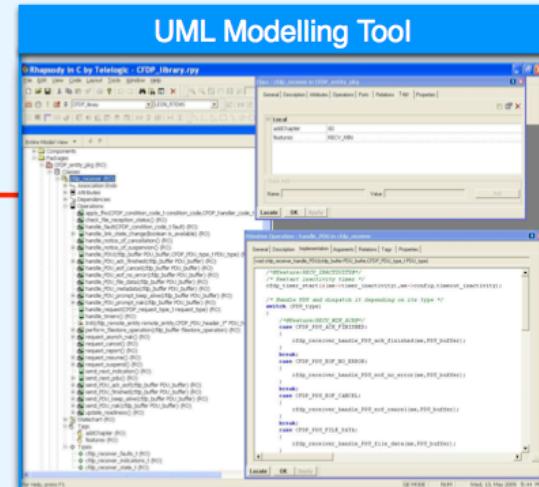
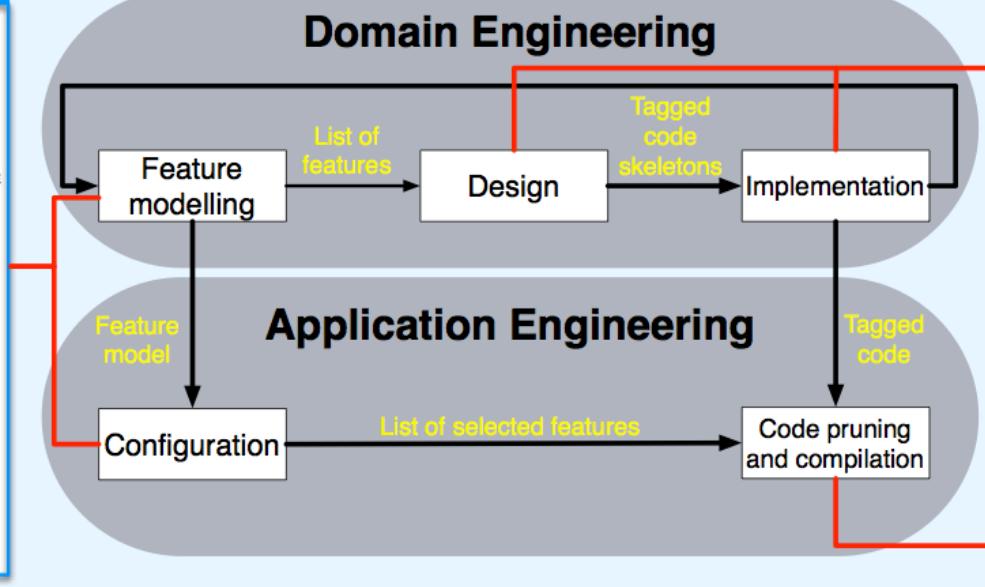
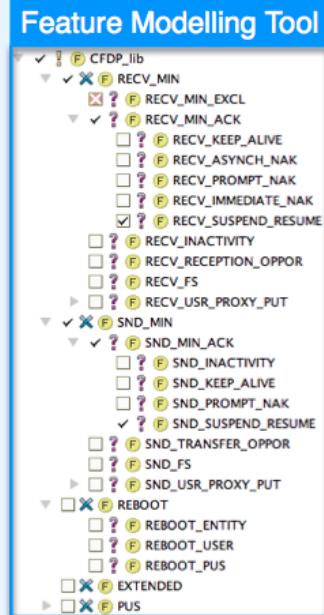


```

/*@feature:recv_min@*/ /*@file_feature@*/
void cfdp_receiver_handle_PDU(cfdp_receiver* const me, struct cfdp_buffer* PDU_buffer,
CFDP_PDU_type_t PDU_type) {
    /* Handle PDU and dispatch it depending on its type */
    switch (PDU_type) {
        /*@feature:recv_min_ack@*/
        case CFDP_PDU_ACK_FINISHED:
        {
            cfdp_receiver_handle_PDU_ack_no_error(me,PDU_buffer);
        } break;
        case CFDP_PDU_EOF_NO_ERROR:
        {
            cfdp_receiver_handle_PDU_eof_no_error(me,PDU_buffer);
        } break;
        /*@feature:recv_keep_alive@*/
        case CFDP_PDU_PROMPT_KEEP_ALIVE:
        {
            cfdp_receiver_handle_PDU_prompt_keep_alive(me,PDU_buffer);
        } break;
        /*@feature:recv_prompt_nak@*/
        case CFDP_PDU_PROMPT_NAK:
        {
            cfdp_receiver_handle_PDU_prompt_nak(me,PDU_buffer);
        } break;
        default:
        {
            /* Unexpected PDU -> discard */
        } break;
    }
}

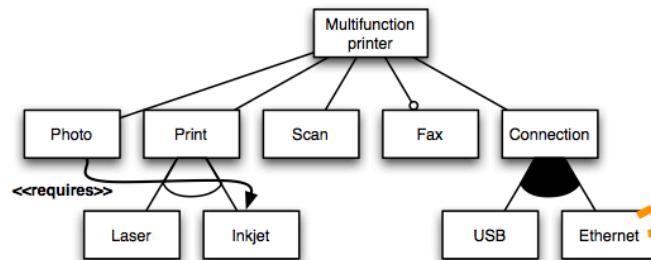
```

AST (Abstract Syntax Tree) diagram showing the structure of the annotated code.



Product Derivation

feature model

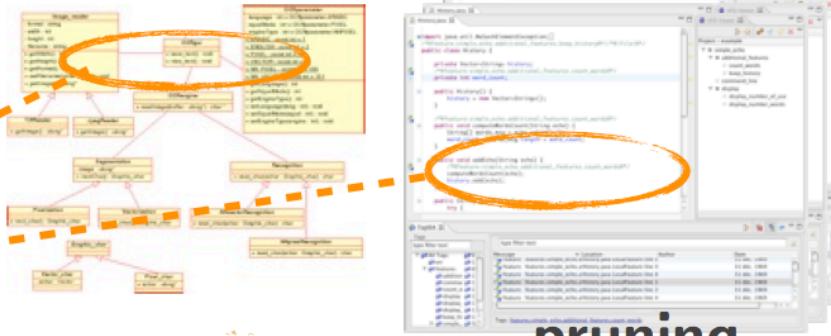


configuration

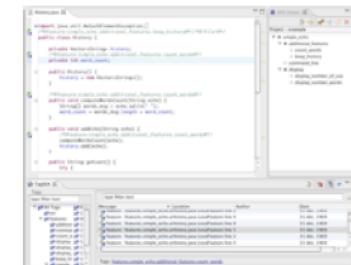
{ MP, Photo, Print, Inkjet, Scan,
Fax, Connection, USB, Ethernet }

product spec

variable model and
code assets



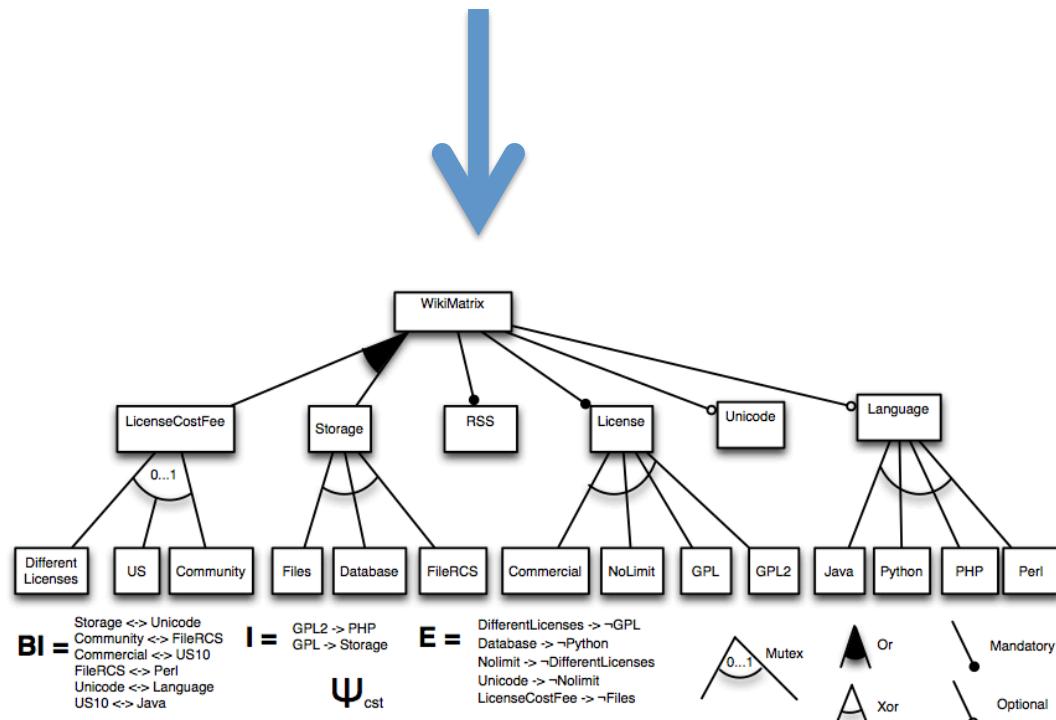
pruning,
composition,
weaving,
transformation



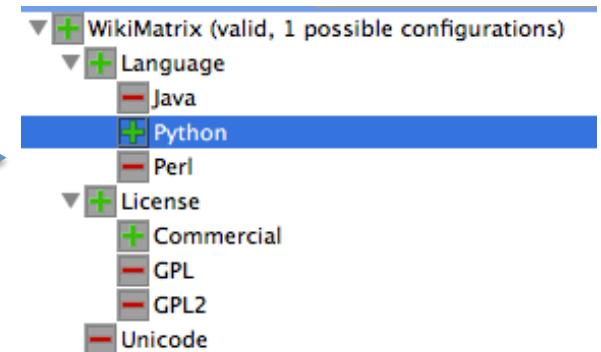
product

From Products to Feature Models

Identifier	License	Language	Storage	LicenseCostFee	RSS	Unicode
Confluence	Commercial	Java	Database	US10	Yes	Yes
PBwiki	Nolimit	No	No	Yes	Yes	No
MoinMoin	GPL	Python	Files	No	Yes	Yes
DokuWiki	GPL2	PHP	Files	No	Yes	Yes
PmWiki	GPL2	PHP	Files	No	Yes	Yes
DrupalWiki	GPL2	PHP	Database	Different Licences	Yes	Yes
TWiki	GPL	Perl	FilesRCS	Community	Yes	Yes
MediaWiki	GPL	PHP	Database	No	Yes	Yes

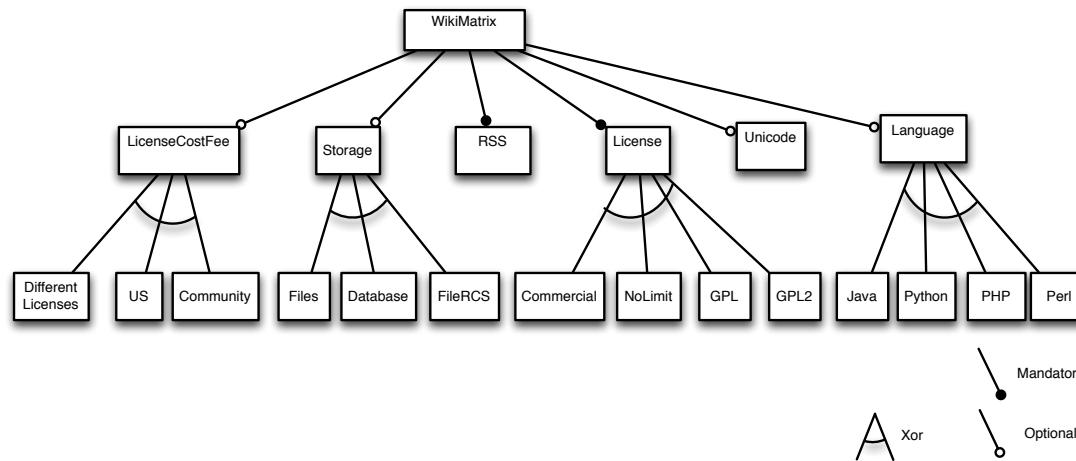


PBWORKS



Manual extraction of a feature model from product description(s) is not possible

Identifier	License	Language	Storage	LicenseCostFee	RSS	Unicode
Confluence	Commercial	Java	Database	US10	Yes	Yes
PBwiki	Nolimit	No	No	Yes	Yes	No
MoinMoin	GPL	Python	Files	No	Yes	Yes
DokuWiki	GPL2	PHP	Files	No	Yes	Yes
PmWiki	GPL2	PHP	Files	No	Yes	Yes
DrupalWiki	GPL2	PHP	Database	Different Licences	Yes	Yes
TWiki	GPL	Perl	FilesRCS	Community	Yes	Yes
MediaWiki	GPL	PHP	Database	No	Yes	Yes



640 configurations
(634 counter examples)

Exact set of configurations, each configuration corresponding to at least one product

Automation

- Each product description is encoded as a feature model

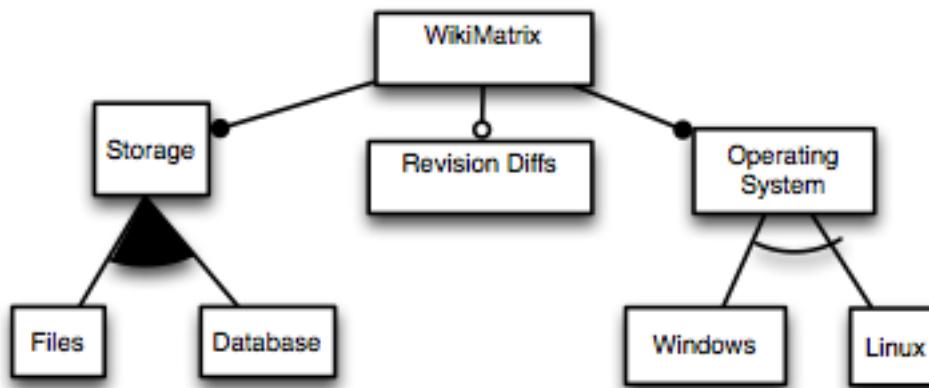
Identifier	License	Language	Storage	LicenseCostFee	RSS	Unicode	
Confluence	Commercial	Java	Database	US10	Yes	Yes	→ fm1
PBwiki	Nolimit	No	No	Yes	Yes	No	→ fm2
MoinMoin	GPL	Python	Files	No	Yes	Yes	→ fm3
DokuWiki	GPL2	PHP	Files	No	Yes	Yes	→ fm4
PmWiki	GPL2	PHP	Files	No	Yes	Yes	→ fm5
DrupalWiki	GPL2	PHP	Database	Different Licences	Yes	Yes	→ fm6
TWiki	GPL	Perl	FilesRCS	Community	Yes	Yes	→ fm7
MediaWiki	GPL	PHP	Database	No	Yes	Yes	→ fm8

- Feature models {fm1, fm2,...,fm8} are merged

Each product description is encoded as a feature model

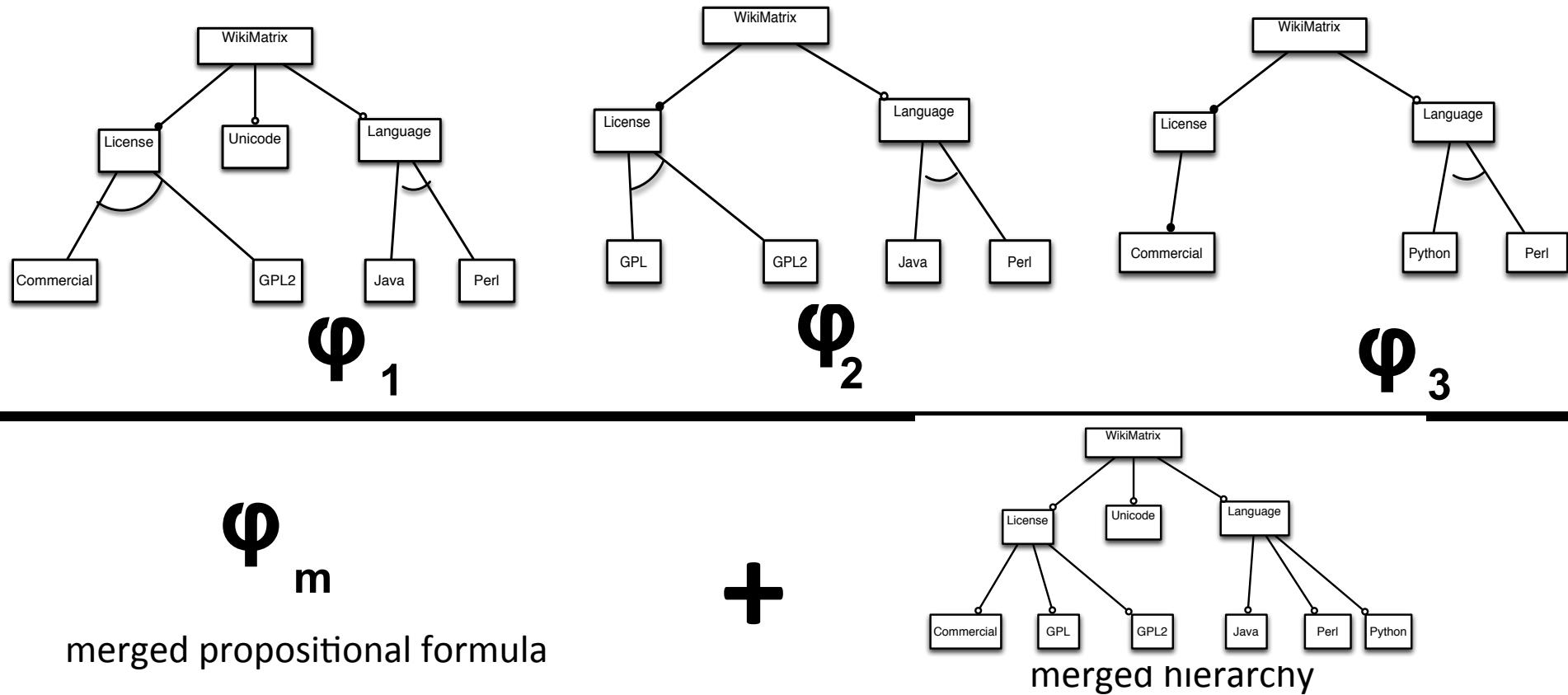
ID	Revision Diffs	Operating System	Storage	Open
...
FooWiki	Plugin	Windows; Linux	Files ; Database	No
...

(a) A wiki engine description

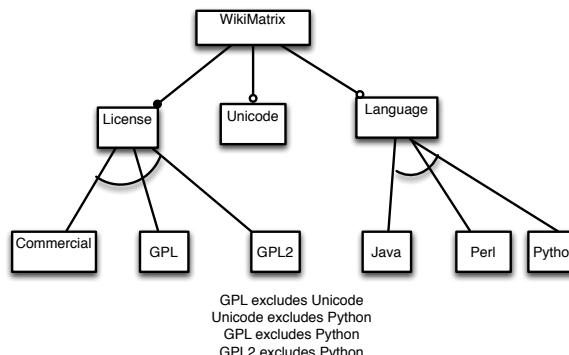


(b) Corresponding FM

Merging of feature models



Set mandatory features
 Detect Xor and Or-groups
 Compute “implies/excludes”



Feature models in the real

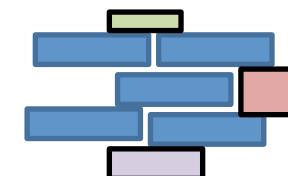
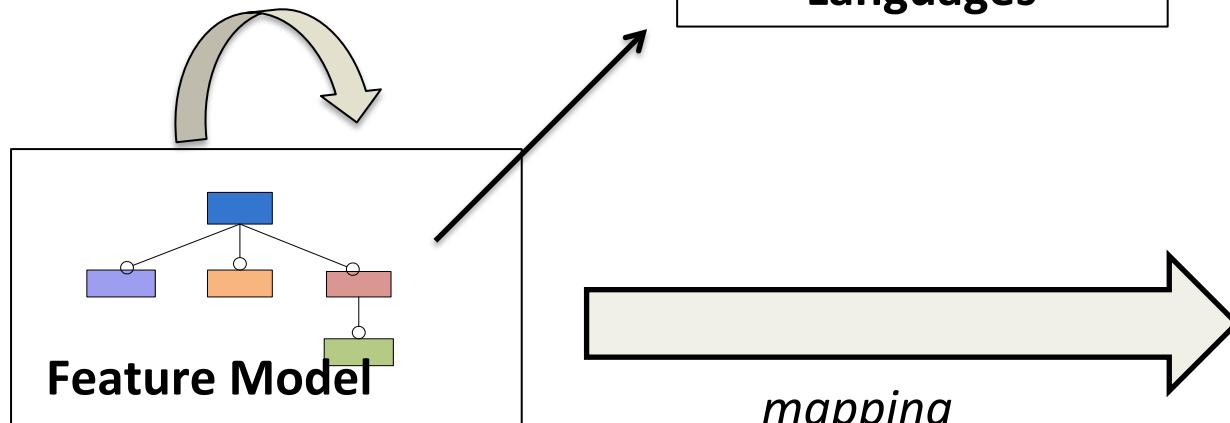
- SPLOT repository
 - more than 200 feature models reported from the literature (various domains)
- Linux feature model
 - worst case: more than 6300 features!
 - eCos, FreeBSD, BusyBox, etc.
- Automotive industry
 - thousands of features
- Wiki matrix
 - wiki engines: ~ 2000 features
- **Reverse engineering procedures are emerging**

Summary

- Variability models for mastering complexity: variability is everywhere, precise specification, automated reasoning and management
- Feature Models
 - **Formalism and theory**: modeling principles, semantics, logics
 - **Language** to elaborate/build variability models
 - **Reasoning techniques** to analyze properties of an SPL (scalable and automated way)
 - **Applications**: variability models and so what?

reasoning and management

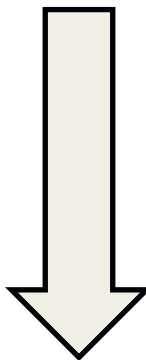
Textual or Graphical Languages



Feature Model

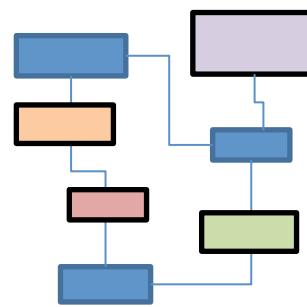
mapping

Reusable Assets
(e.g., models or source code)



Configurators

derivation



product_n