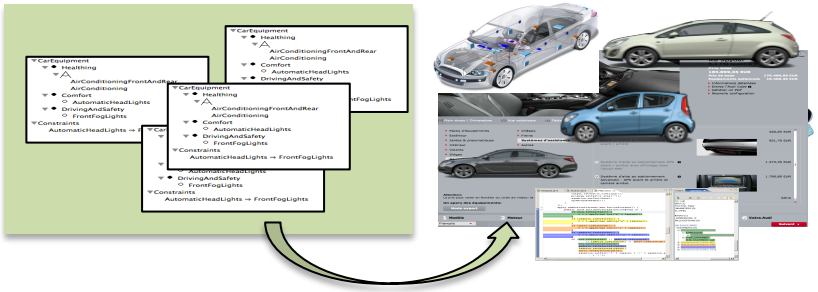


FAMiliAR

Managing Feature Models

Mathieu Acher

UMR IRISA

informatiques mathématiques Inria

UNIVERSITÉ DE RENNES 1

Learning Feature Models with

(a.k.a implementing the introductory example)

FAMiliAR

(FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)

http://familiar-project.github.com/

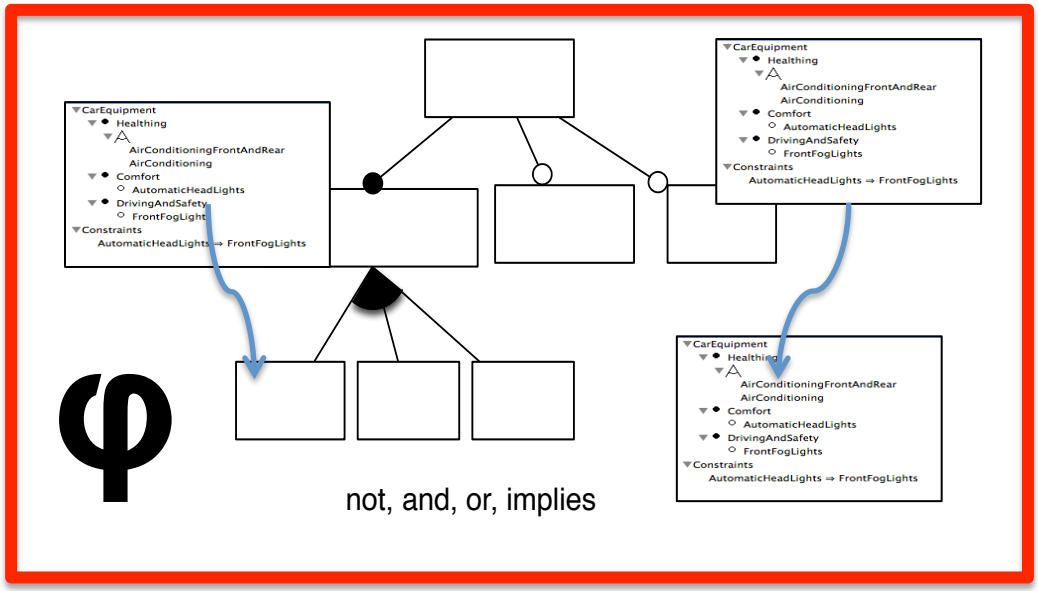
COMMON VARIABILITY LANGUAGE

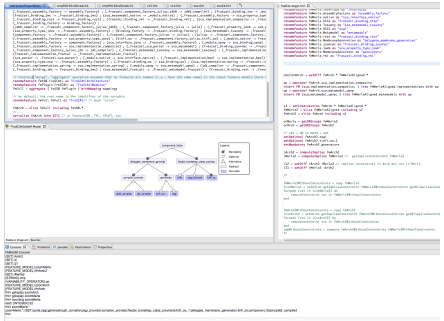
S.P.L.O.T. Software Product Lines Online Tools

IDE Feature

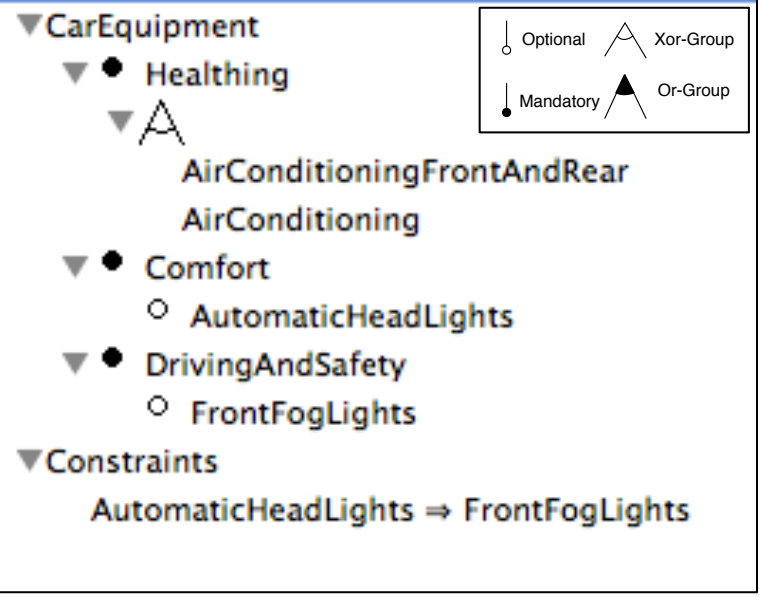
TVL

DIMACS





Mathieu Acher, Philippe Collet, Philippe Lahire, Robert B. France « A Domain-Specific Language for Large-Scale Management of Feature Models » Science of Computer Programming (SCP), 2013



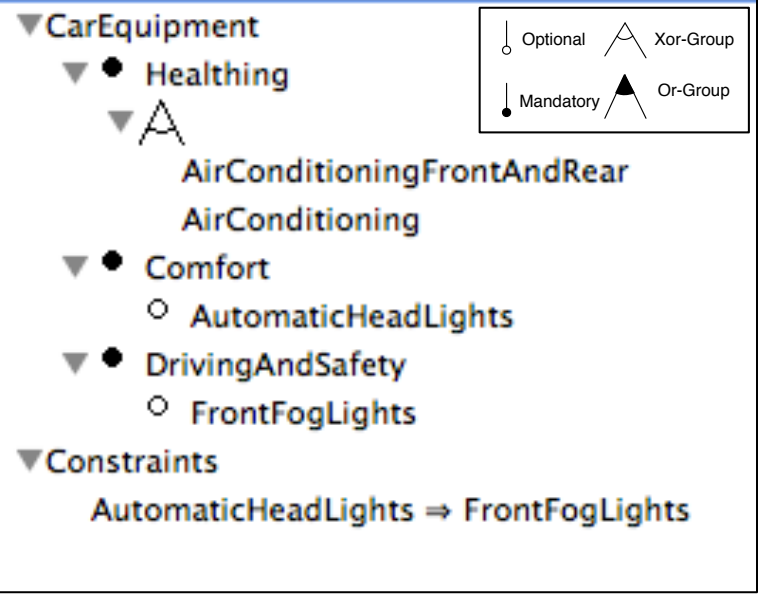

```
fml> convert fmCarEquipment into SPLOT
res1: (STRING) <feature_model name="fmCarEquipment">
<meta>
<data name="description"/>
<data name="creator"/>
<data name="address"/>
<data name="email"/>
<data name="phone"/>
<data name="website"/>
<data name="organization"/>
<data name="department"/>
<data name="date"/>
<data name="reference"/>
</meta>
<feature_tree>
:r CarEquipment(_r0)
        :m Healthing(_r1)
                :g [1,1]
                        : AirConditioningFrontAndRear(_r2)
                        : AirConditioning(_r3)
        :m DrivingAndSafety(_r4)
                :o FrontFogLights(_r5)
        :m Comfort(_r6)
                :o AutomaticHeadLights(_r7)
</feature_tree>
<constraints>
C0:~_r7 or _r5
</constraints>
</feature_model>
```

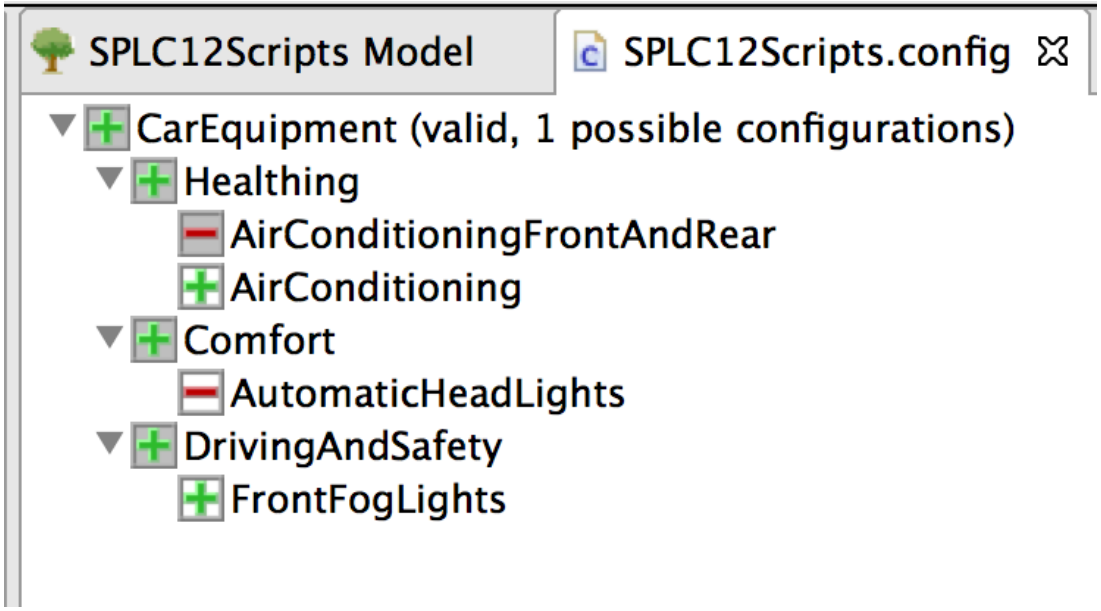
S.P.L.O.T.
Software Product Lines Online Tools

IDE
eature

```
fmCarEquipment = FM (CarEquipment : Healthing DrivingAndSafety Comfort ; // 3 mandatory features
                     Healthing : (AirConditioning|AirConditioningFrontAndRear) ; // Xor
                     DrivingAndSafety : [FrontFogLights] ; // optional
                     Comfort : [AutomaticHeadLights] ; // optional
                     // cross-tree constraints
                     AutomaticHeadLights -> FrontFogLights ; )
```

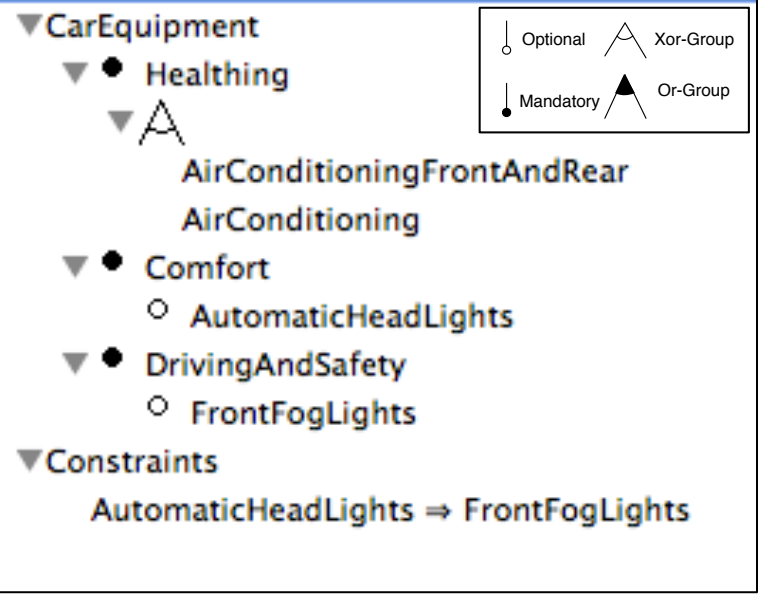
```
MacBook-Pro-de-Mathieu-3:Documents macher1$ java -Xmx1024M -jar FML-basic-1.1.jar FMLTestRepository/carEquipTuto.fml
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 1.1 (beta)
http://familiar-project.github.com/
fml> ls
(FEATURE_MODEL) fmCarEquipment
fml>
```






```
fmCarEquipment = FM (CarEquipment : Healthing DrivingAndSafety Comfort ; // 3 mandatory features
                                    Healthing : (AirConditioning|AirConditioningFrontAndRear) ; // Xor
                                    DrivingAndSafety : [FrontFogLights] ; // optional
                                    Comfort : [AutomaticHeadLights] ; // optional
                                    // cross-tree constraints
                                    AutomaticHeadLights -> FrontFogLights ; )
```

```
fml> c1 = configuration fmCarEquipment
c1: (CONFIGURATION) selected: [Healthing, CarEquipment, DrivingAndSafety, Comfort]       deselected: []
fml> select AirConditioning FrontFogLights in c1
res2: (BOOLEAN) true
fml> deselect AutomaticHeadLights in c1
res3: (BOOLEAN) true
fml> selectedF c1
res4: (SET) {Comfort;CarEquipment;Healthing;AirConditioning;DrivingAndSafety;FrontFogLights}
```



{CarEquipment, Comfort,
DrivingAndSafety,
Healthing}

X

{AirConditioning, FrontFogLights}
{AutomaticHeadLights, AirConditioning, FrontFogLights}
{AutomaticHeadLights, FrontFogLights, AirConditioningFrontAndRear}
{AirConditioningFrontAndRear}
{AirConditioning}
{AirConditioningFrontAndRear, FrontFogLights}

```
fmCarEquipment = FM (CarEquipment : Healthing DrivingAndSafety Comfort ; // 3 mandatory features
                                    Healthing : (AirConditioning|AirConditioningFrontAndRear) ; // Xor
                                    DrivingAndSafety : [FrontFogLights] ; // optional
                                    Comfort : [AutomaticHeadLights] ; // optional
                                    // cross-tree constraints
                                    AutomaticHeadLights -> FrontFogLights ; )
```

```
fml> co = cores fmCarEquipment
co: (SET) {CarEquipment;Healthing;DrivingAndSafety;Comfort}
fml> fmCarEquipment.*
res6: (SET) {DrivingAndSafety;AirConditioningFrontAndRear;Comfort;Healthing;FrontFogLights;AirConditioning;AutomaticHeadLights;CarEquipment}
fml> setDiff fmCarEquipment.* co
res7: (SET) {AutomaticHeadLights;FrontFogLights;AirConditioning;AirConditioningFrontAndRear}
fml>
```

```
res1: (DOUBLE) 6.0
```

```
// carEquipTuto.fml
fmCarEquipment = FM ("carEquip.fml")

c1 = configuration fmCarEquipment
select AirConditioning FrontFogLights in c1
deselect AutomaticHeadLights in c1
s0 = selectedF c1


s1 = configs fmCarEquipment // configuration set
n1 = size s1
n2 = counting fmCarEquipment // efficient way, se
assert (n2 eq n1)

co = cores fmCarEquipment // features necessarily
vp = setDiff fmCarEquipment.* co
//
```

Much more than that!

Let us have a deeper look

FAMiliAR

(FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)

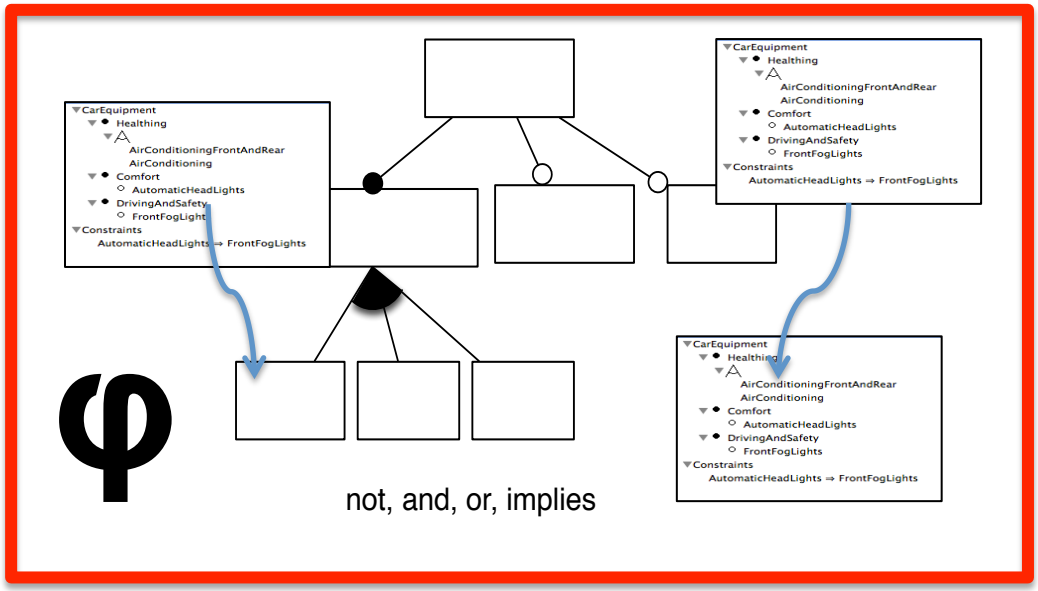
http://familiar-project.github.com/

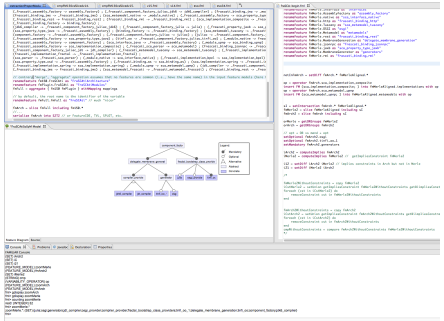
COMMON VARIABILITY LANGUAGE

S.P.L.O.T. Software Product Lines Online Tools

IDE Feature

TVL
DIMACS





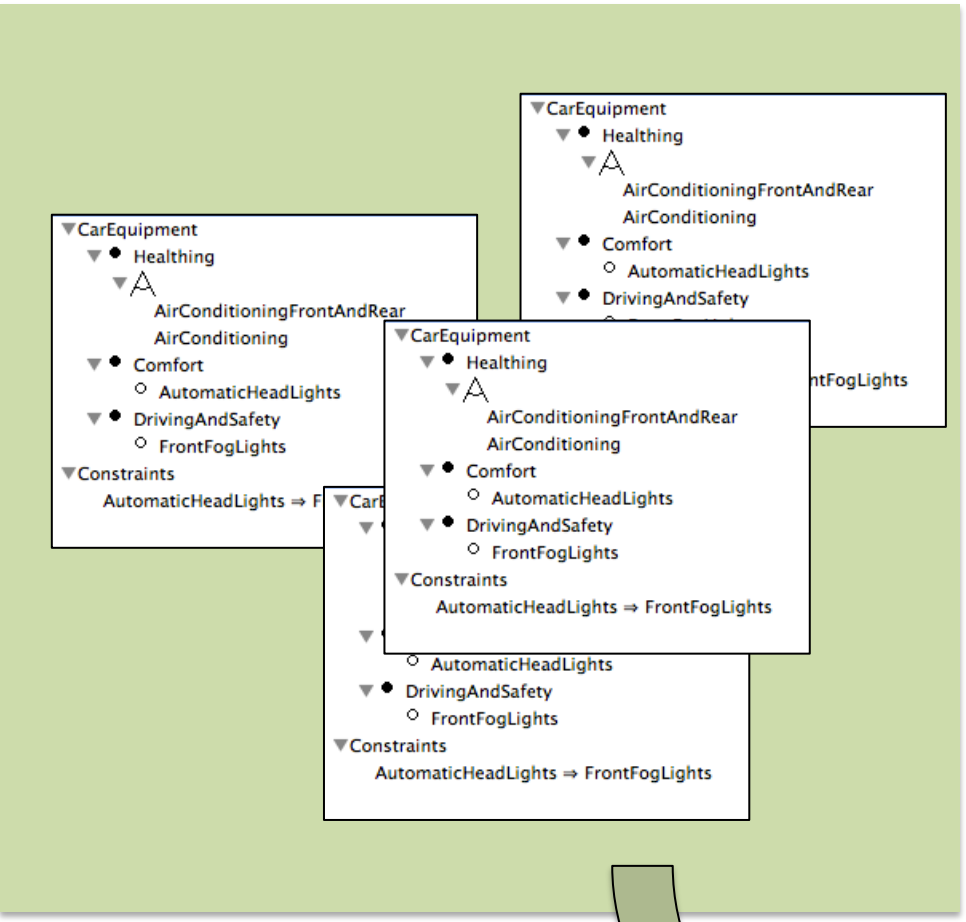
importing, exporting, composing, decomposing, editing, configuring, reverse engineering, computing "diffs", refactoring, testing, and reasoning about (multiple) variability models

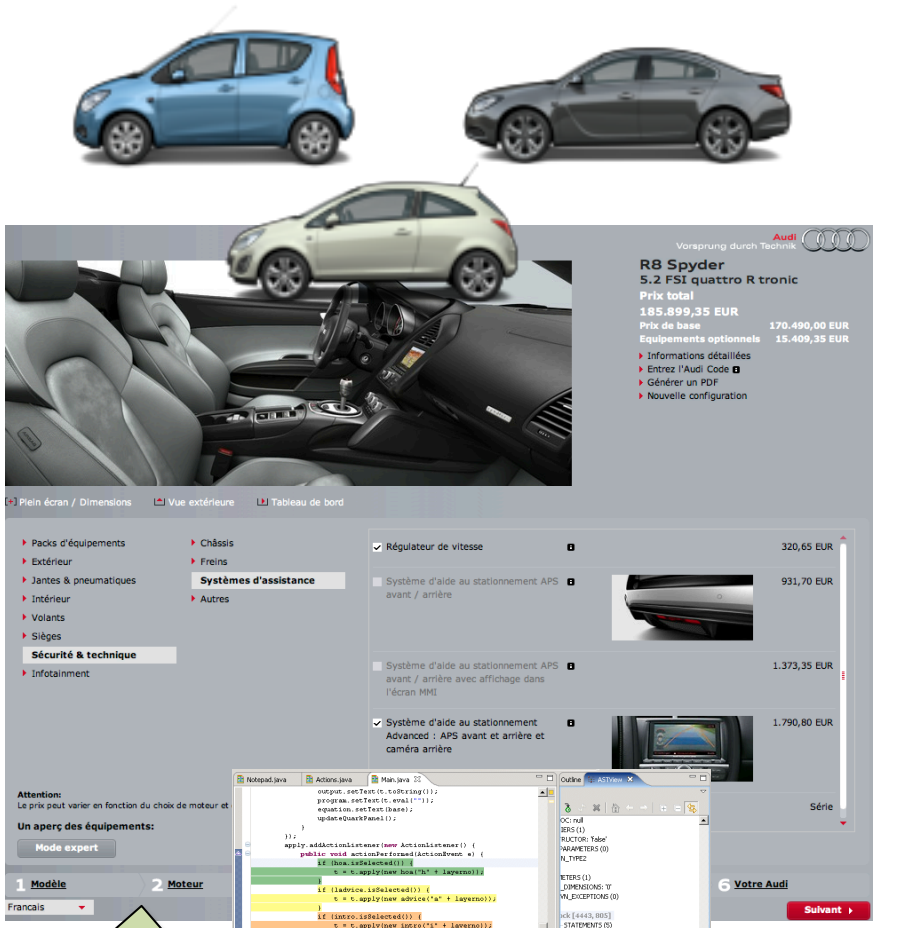
Mathieu Acher, Philippe Collet, Philippe Lahire, Robert B. France « A Domain-Specific Language for Large-Scale Management of Feature Models » Science of Computer Programming (SCP), 2013

#1 Automated Analysis

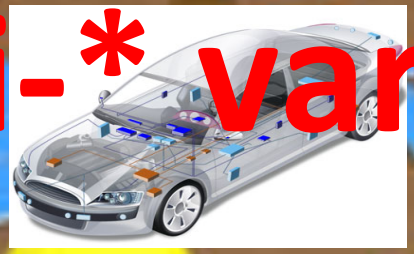
- CarEquipment
  - Healthing
    - AirConditioningFrontAndRear
    - AirConditioning
  - Comfort
    - AutomaticHeadLights
  - DrivingAndSafety
    - FrontFogLights
- Constraints
  - AutomaticHeadLights ⇒ FrontFogLights

#2 Multiple Feature Models



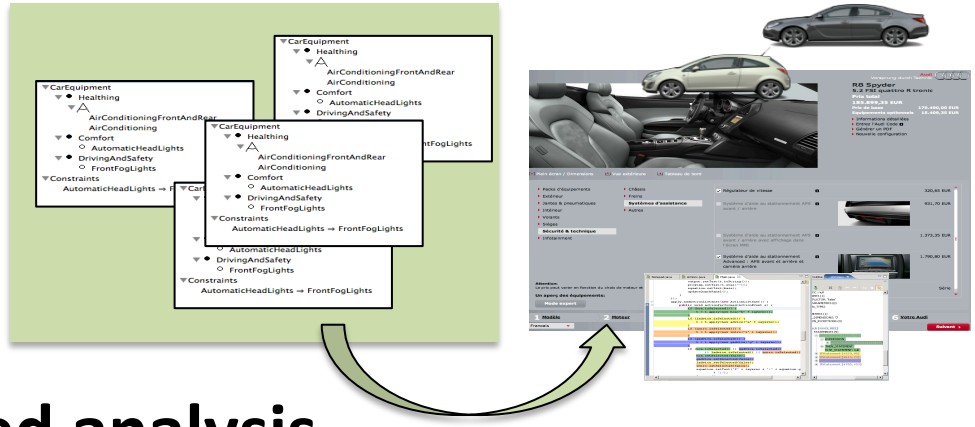


Multi-* variability



* systems, perspectives, or stakeholders

Two Key Requirements



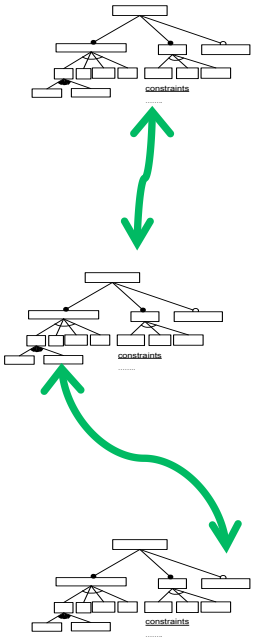
- **#1 Automated analysis**
  - Aka support to better understand and play with your feature model (TVL model)

Automated analysis of feature models 20 years later: A literature review ☆

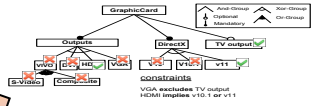
David Benavides *, Sergio Segura, Antonio Ruiz-Cortés

- **#2 Managing <u>multiple</u> feature models**
  - Composing / Decomposing / Diff and Reasoning about their relationships
  - Combining these operators

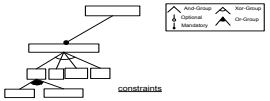
FAMiliAR language and environment




```
// foo.fml
fm1 = FM ("foo1.tvl")
fm2 = FM ("foo2.m")
fm3 = merge intersection { fm1 fm2 }
c3 = counting fm3
renameFeature fm3.TV as "OutputTV"
fm5 = aggregate { fm3 FM ("foo4.xml") }
assert (isValid fm5)
fm6 = slice fm5 including fm5.TV.*
export fm6
```

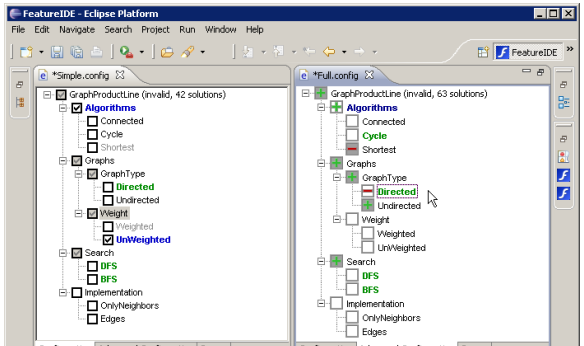


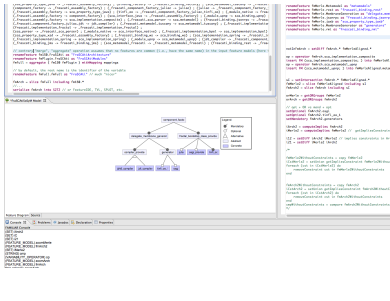
True/False
8759
"OutputTV", "TV"



Interoperability **Language facilities** **Environment**

eclipse

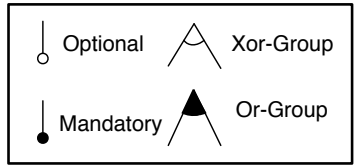




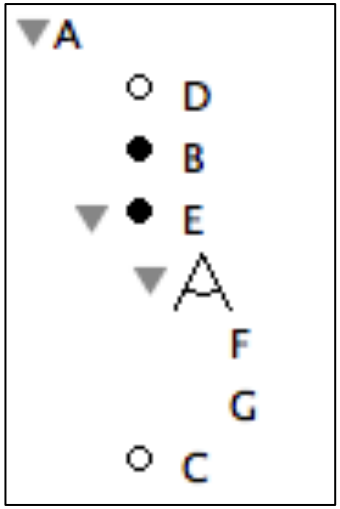
FAMiliAR features

helloworld.fml

Hello World




```
fm1 = FM ("foo1.tvl")
s = "hello world"
c = counting fm1
n = size fm1.*
println s
```




```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.6.jar helloworld.fml
hello world

FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.6 (beta)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(DOUBLE) c
(FEATURE_MODEL) fm1
(INTEGER) n
(STRING) s
fml>
```

basics2.fml

Typed language

- Domain-specific types
  - Feature Model,
  - Configuration,
  - Feature,
  - Constraint
- Other types include
  - Set
  - String
  - Boolean,
  - Enum,
  - Integer and Real.

```
fm1 = FM ("foo1.tvl")
ft1 = root fm1
ft2 = fm1.B
fts = fm1.*
n = size fts
n2 = cores fm1

cf = configuration fm1
select B in cf
deselect C in cf

cst1 = constraint (B -> !C)
addConstraint cst1 to fm1
```

- A set of operations, called **operators**, are defined for a given type.

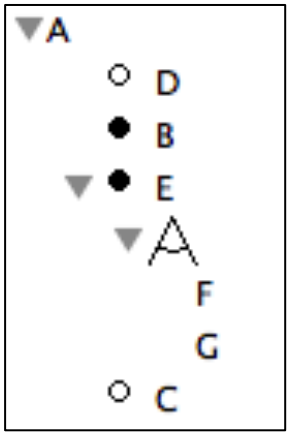
Typed language

```
fm1 = FM ("foo1.tvl")
ft1 = root fm1
ft2 = fm1.B
fts = fm1.*
n = size fts
n2 = cores fm1

cf = configuration fm1
select B in cf
deselect C in cf

cst1 = constraint (B -> !C)
addConstraint cst1 to fm1
```

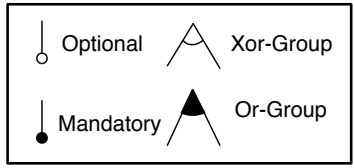
```
fml> ls
(FEATURE_MODEL) fm1
(SET) fts
(FEATURE) ft2
(CONSTRAINT) cst1
(INTEGER) n
(CONFIGURATION) cf
(SET) n2
(FEATURE) ft1
```

Typed language

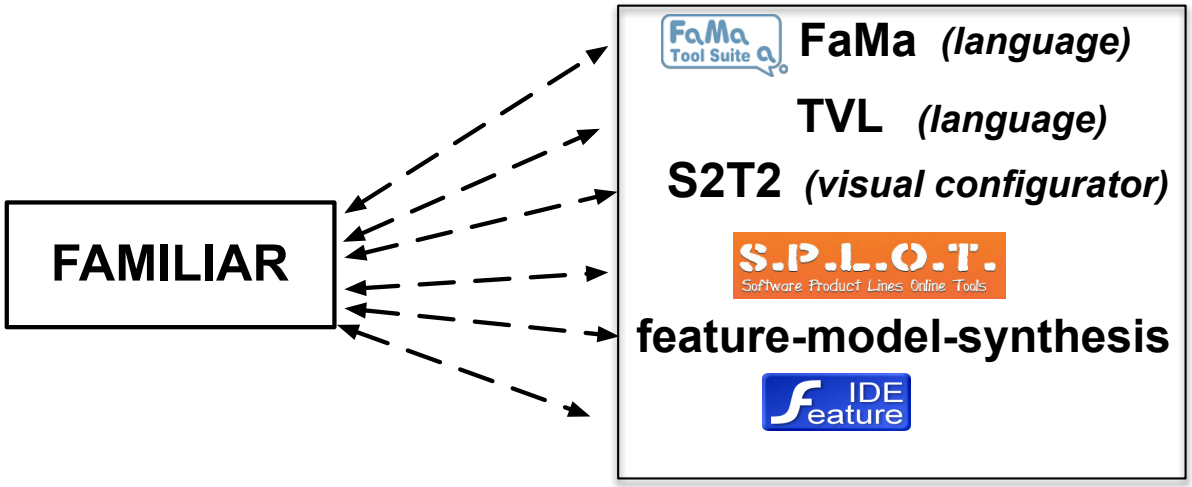
```
fm1 = FM ("foo1.tvl")
ft1 = root fm1
ft2 = fm1.B
fts = fm1.*
n = size fts
n2 = cores fm1

cf = configuration fm1
select B in cf
deselect C in cf

cst1 = constraint (B -> !C)
addConstraint cst1 to fm1
```




```
fml> ft1
ft1: (FEATURE) A
fml> ft2
ft2: (FEATURE) B
fml> fts
fts: (SET) {G;E;D;F;C;A;B}
fml> n
n: (INTEGER) 7
fml> n2
n2: (SET) {E;A;B}
fml> cf
cf: (CONFIGURATION) selected: [E, A, B]          deselected: [C]
fml> cst1
cst1: (CONSTRAINT) (B -> !C)
fml> fm1
fm1: (FEATURE_MODEL) A: [D] B E [C] ;
E: (F|G) ;
(B -> !C);
```



Importing/Exporting feature models



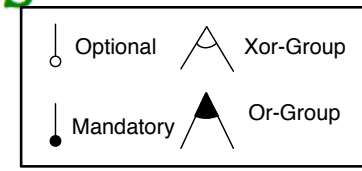
Internal notation *or* by "filename extensions"

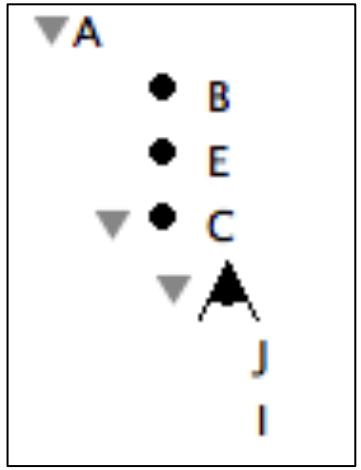
```
fm1 = FM ("foo1.tvl")
fm2 = FM (A : [B] [C] D ; )
fm3 = FM ("foo2.m")
serialize fm2 into SPLOT // export
```

ftAccessors.fml

Feature Accessors (1)

```
fm1 = FM (A : B [C] ; B : E F ; C : (I|J) ; )

r1 = root fm1
s = children r1
s1 = children fm1.A
assert (s eq s1) // equality of the two sets
ft1 = parent fm1.F
str1 = name ft1
ft2 = parent F // parent fm1.F

// another FM
fm2 = FM (A : B C E ; C : (I|J)+ ; )
ft3 = fm2.B
ft4 = name B // ambiguity
```



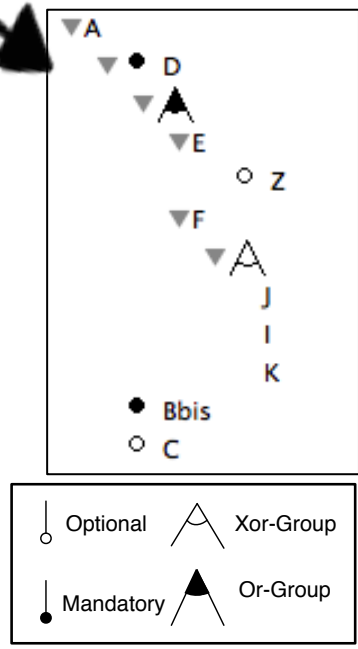


ftAccessors2.fml

Other constructs

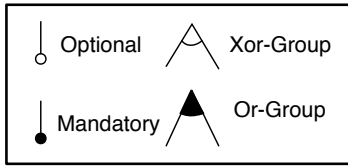
```
fm1 = FM (A: B [C] D; D : (E|F)+; F : (I|J|K); E : [Z]; )
fm1bis = copy fm1 // save the original version

renameFeature fm1.B as "Bbis"
s1 = fm1.* // set of features of fm1
foreach (ft1 in s1) do
        println ft1
end
```

```
acher-scr:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar ftAccessors2.fml
Bbis

D

E

A

Z

I

C

J

K

F

FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.5 (bet
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> exit
Bye, FAMILIAR user!
```

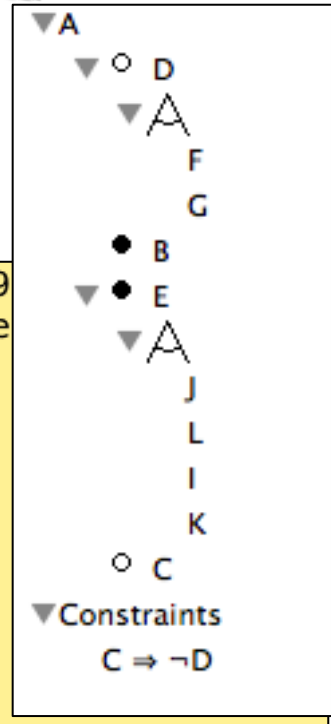


conf.fml

Configuration



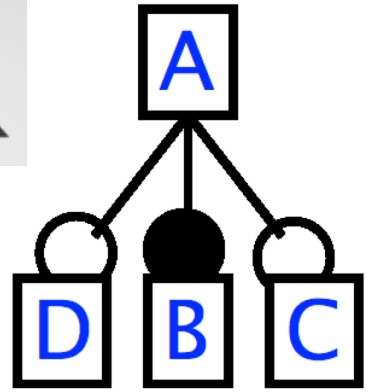

```
fm1 = FM (A: B [C] [D] E; D : (F|G) ; E : (I|J|K|L) ; C -> !D ; )
c1 = configuration fm1
select C in c1
sc1 = selectedF c1 // accessors
cFM1 = asFM c1 // configuration and FM: back!
```



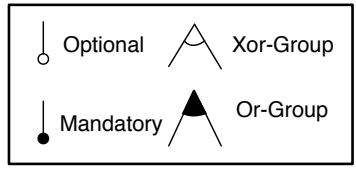

```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Re
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cFM1
cFM1: (FEATURE_MODEL) A: B E C ;
E: (J|L|I|K) ;
E;
A;
B;
C;
fml> fm1
fm1: (FEATURE_MODEL) A: [D] B E [C] ;
D: (F|G) ;
E: (J|L|I|K) ;
(C -> !D);
```

A ^
A ⇔ B ^
C => A ^
D => A

FAMiliAR



FM



φ

```
fml> fm1 = FM ("output/fm1.tvl")
root A {
    group [ 3..3 ] {
        opt D {
        },
        B {
        },
        opt C {
        }
    }
}
fm1: (FEATURE_MODEL) A: [D] B [C] ;
```

```
fm1bis = FM ("foo3.dimacs")
fm1bisbis = FM ("foo3.constraints")
```

```
fml> c1 = cores fm1
c1: (SET) {B;A}
fml> c1bis = cores fm1bis
```

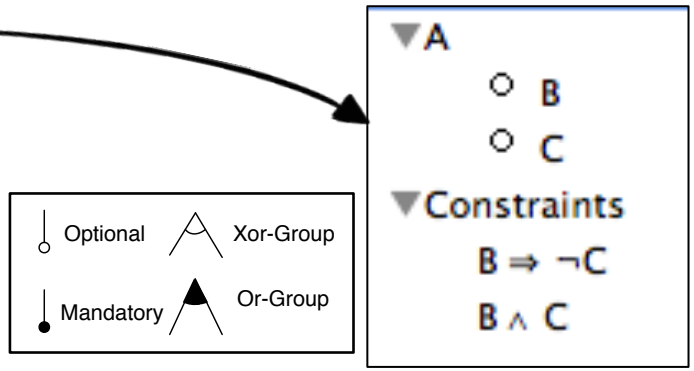
```
fml> compare fm1 fm1bis
res7: (STRING) REFACTORING
fml> compare fm1bis fm1bisbis
res8: (STRING) REFACTORING
```

```
fml> c1 eq c1bisbis
res6: (BOOLEAN) true
```

operatorsFM.fml

Operations for Feature Models (1)

```
fm1 = FM (A : [B] [C] ; B -> !C ; B and C ; )
b1 = isValid fm1
```

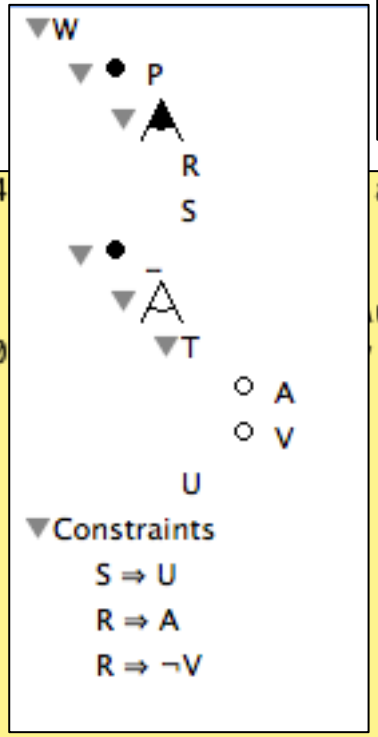


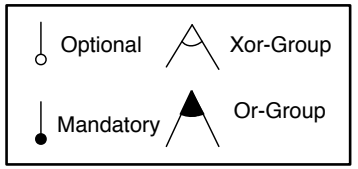

```
acher-scr:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar operatorsFM.f
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoni
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(FEATURE_MODEL) fm1
(BOOLEAN) b1
fml> b1
b1: (BOOLEAN) false
fml> configs fm1
res0: (SET) {}
```

operatorsFM2.fml

Operations for Feature Models (2)

```
1 fm1 = FM (W : P (T|U); P : (R|S)+ ; T : [V] [A] ; R -> !V ; S -> U ; R -> A ; )
2 b1 = isValid fm1
3 s1 = configs fm1
4 c1 = counting fm1
5 dfm1 = deads fm1
6 println "cores: ", cores fm1
7 fo1 = falseOptionals fm1
```



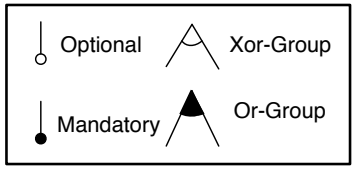



```
acher-scr:FML-scripts macher$ java -jar -Xmx1024                    ar operatorsFM2.fml
cores: {P;W}

FAMILIAR (for FeAture Model scrIpt Language for                     utomatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 60
https://nyx.unice.fr/projects/familiar/
fml> ls
(SET) fo1
(SET) dfm1
(SET) s1
(DOUBLE) c1
(BOOLEAN) b1
(FEATURE_MODEL) fm1
fml> c1
c1: (DOUBLE) 2.0
fml> fo1
fo1: (SET) {A}
fml> dfm1
dfm1: (SET) {V}
```

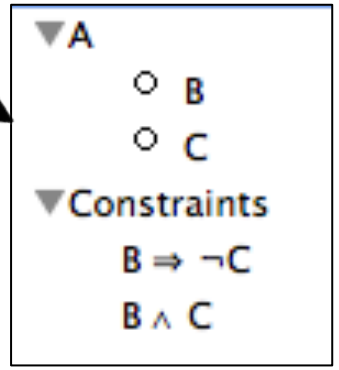
φ

operatorsFM3.fml

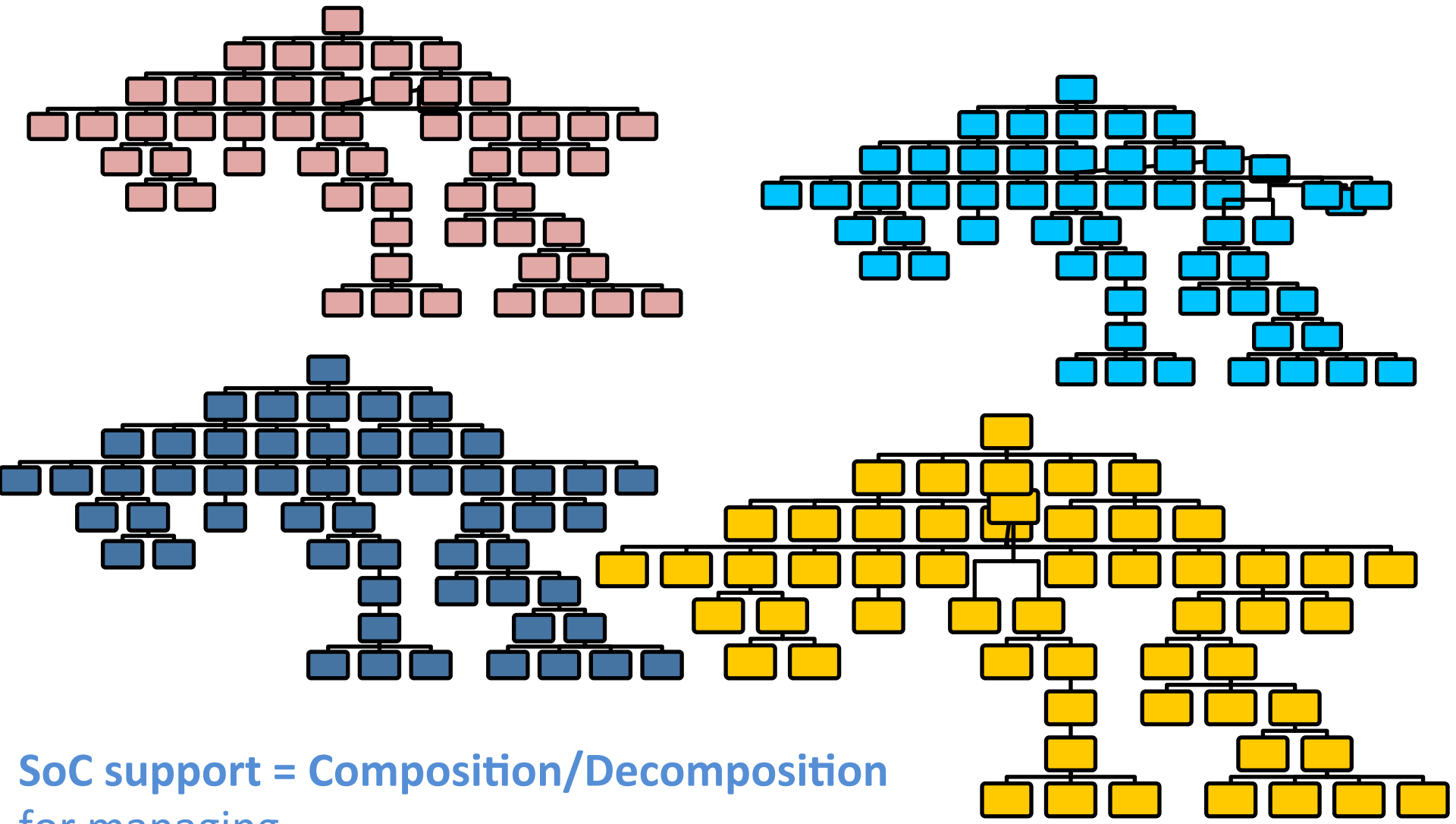
Operations for Feature Models (3)




```
fm1 = FM (A : [B] [C] ; B -> !C ; B and C ; )
b1 = isValid fm1

csts1 = constraints fm1
foreach (cst in csts1) do
        println "removing constraint... ", cst
        removeConstraint cst in fm1
        c = counting fm1
        println "now the number of valid configurations is... ", c
end
```




```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar operatorsFM3.fml
removing constraint... (B & C)

now the number of valid configurations is... 3.0

removing constraint... (B -> !C)

now the number of valid configurations is... 4.0
```

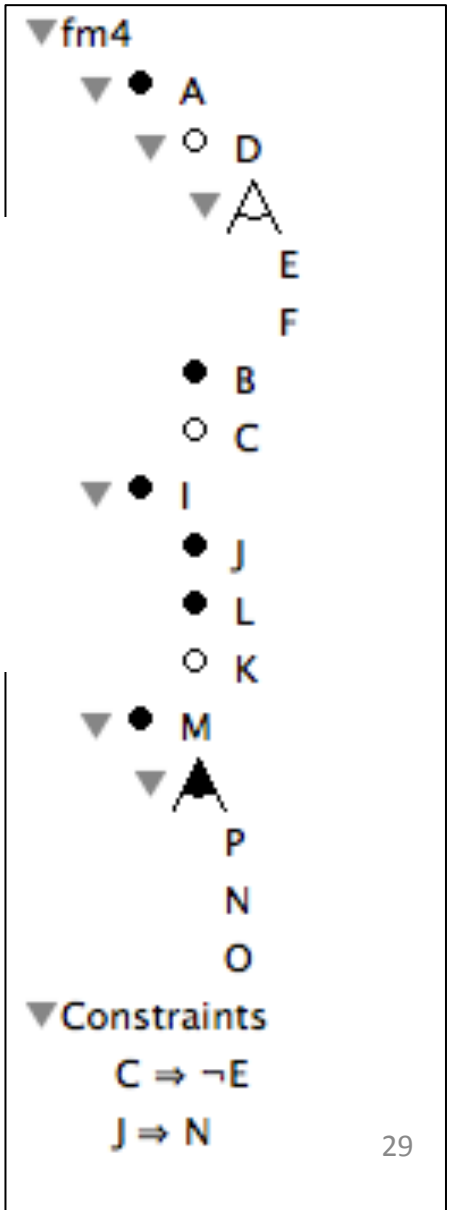


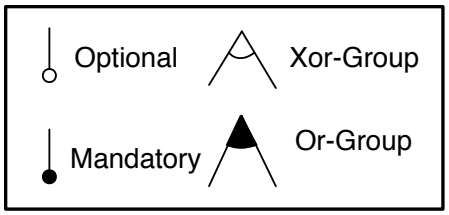
SoC support = Composition/Decomposition
for managing
large, complex and multiple
feature models

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.

Composing Feature Models (1)

```
fm1 = FM (A : B [C] [D] ; D : (E|F) ; C -> !E; )
fm2 = FM (I : J [K] L ; )
fm3 = FM (M : (N|O|P)+ ; )
cst = constraints (J implies N ; )

// equivalent to aggregate { fm1 fm2 fm3 }
fm4 = aggregate fm* withMapping cst
```





aggregate1.fml

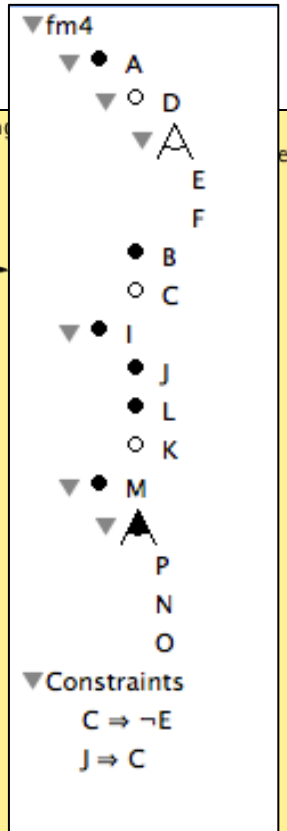
Composing Feature Models (2)

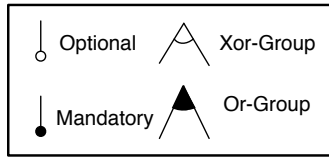
```
fm1 = FM (A : B [C] [D] ; D : (E|F) ; C -> !E; )
fm2 = FM (I : J [K] L ; )
fm3 = FM (M : (N|O|P)+ ; )
cst = constraints (J implies C ; )

// equivalent to aggregate { fm1 fm2 fm3 }
fm4 = aggregate fm* withMapping cst

// composition sometimes leads to "anomalies"
dfm4 = deads fm4
```

```
fm1 = FM (A : B [C] [D] ; D : (E|F) ; C -> !E; )
fm2 = FM (I : J [K] L ; )
fm3 = FM (M : (N|O|P)+ ; )
cst = constraints (J implies N ; )

// equivalent to aggregate { fm1 fm2 fm3 }
fm4 = aggregate fm* withMapping cst
```

Previous version

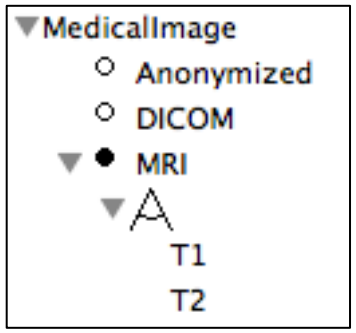
```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar ag
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cores fm4
res0: (SET) {C;fm4;A;J;I;B;L;M}
fml> falseOptionals fm4
res1: (SET) {F;C}
fml> operator fm4.C
res2: (VARIABILITY_OPERATOR) OPTIONAL
fml> operator fm4.F
res3: (VARIABILITY_OPERATOR) ALTERNATIVE
fml> sibling fm4.F
res4: (SET) {E}
fml> deads fm4
res5: (SET) {E}
fml> operator fm4.E
res6: (VARIABILITY_OPERATOR) ALTERNATIVE
fml> fm4
fm4: (FEATURE_MODEL) fm4: A I M ;
A: [D] B [C] ;
I: J L [K] ;
M: (P|N|O)+ ;
D: (E|F) ;
(C -> !E);
(J -> C);
```

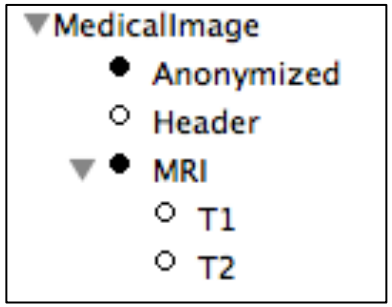


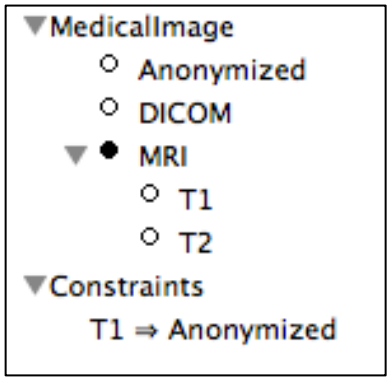


mergeMI.fml

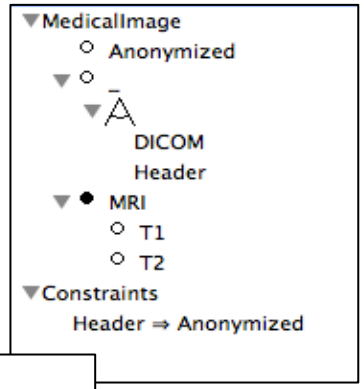
Composing Feature Models (3)

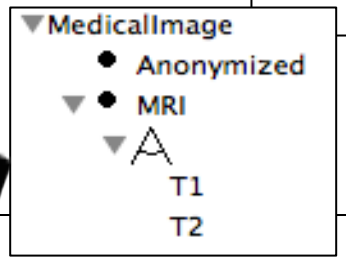







```
fmsupp1 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : (T1|T2) ;  )
fmsupp2 = FM (MedicalImage : Anonymized MRI [Header] ; MRI : [T1] [T2] ; )
fmsupp3 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : [T1] [T2] ; T1 -> Anonymized; )

s1 = configs fmsupp1
s2 = configs fmsupp2
s3 = configs fmsupp3

s123 = setUnion s3 setUnion s1 s2

fmSupp = merge sunion fmsupp*

assert (size s123 eq counting fmSupp)

fmCommon = merge intersection { fmsupp1 fmsupp2 }
sC = configs fmCommon
sC2 = setIntersection s1 s2
```





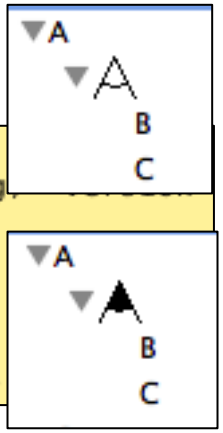
Mathieu Acher, Philippe Collet, Philippe Lahire, Robert B. France « Comparing Approaches for Implementing Feature Model Composition » ECMFA'10

compare.fml

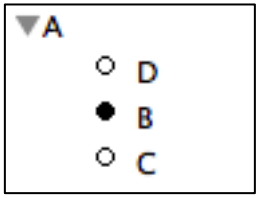
Comparing Feature Models

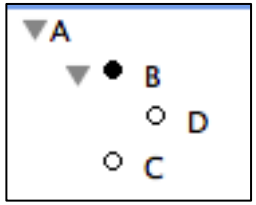
```
fm0 = FM (A: (B|C) ; )
fm1 = FM (A: (B|C)+ ; )
```

```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cmp23
cmp23: (STRING) REFACTORING
fml>
```

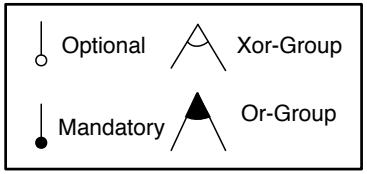



```
assert (cmp10 eq GENERALIZATION)

// example taken from Automated Analysis of Feature Models
fm2 = FM (A: B [C] [D]; )
fm3 = FM (A: B [C]; B : [D]; )
cmp23 = compare fm2 fm3
```





	No Products Added	Products Added
No Products Deleted	Refactoring	Generalization
Products Deleted	Specialization	Arbitrary Edit



see also Thuem, Kastner and Batory, ICSE'09

Combining operators: an example

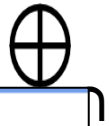
Merge Intersection: Available Suppliers

- Registration
  - Affine
  - Header
    - A
      - v3
      - v1
  - Modality
    - CT
  - Format
    - A
      - DICOM
      - Nifti

- Registration
  - Affine
  - Header
    - A
      - v2
      - v1
  - Modality
    - MRI
  - Format
    - A
      - Nifti
      - GE

- Registration
  - Rigid
  - Header
    - A
      - v3
      - v2
  - Modality
    - A
      - PET
      - MRI
  - Format
    - Nifti

Suppliers?
Products?



- Registration
  - Header
    - v1
  - Modality
    - CT
  - Format
    - A
      - DICOM
      - Nifti

- Registration
  - Header
    - v1
  - Modality
    - A
      - CT
      - MRI
  - Format
    - A
      - DICOM
      - Nifti
- Constraints
  - DICOM ⇒ ¬MRI

A customer has some requirements

suppliersExample0.fml

In FAMILIAR

```
REGsupp1 = FM (Registration : Header Format Modality [Affine] ;
                                                    Header : (v1|v3);
                                                    Format : (DICOM|Nifti) ;
                                                    Modality : CT; )

REGsupp2 = FM (Registration : Header [Affine] Format Modality ;
                                                    Header : (v1|v2);
```

```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.6.jar suppliersExample0.fml
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.6 (beta)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(FEATURE_MODEL) REGsupp3
(FEATURE_MODEL) REGsupp2p
(FEATURE_MODEL) REGrequired
(FEATURE_MODEL) REGsupp3p
(FEATURE_MODEL) REGsupp1p
(FEATURE_MODEL) REGsupp2
(FEATURE_MODEL) REGsupp1
fml> REGsupp3p
REGsupp3p: (FEATURE_MODEL) False
fml> REGsupp1p
REGsupp1p: (FEATURE_MODEL) Registration: Header Modality Format ;
Header: v1 ;
Modality: CT ;
Format: (DICOM|Nifti) ;
```

```
REGsupp1p = merge intersection { REGrequired REGsupp1 }
REGsupp2p = merge intersection { REGrequired REGsupp2 }
REGsupp3p = merge intersection { REGrequired REGsupp3 }
```

Merge Union: Availability Checking

Registration
- Affine
- Header
  - v3
  - v1
  - odality
  - CT
  - rmat
  - DICOM
  - Nifti

Registration
- Affine
- Header
  - v2
  - v1
- Modality
  - MRI
- Format
  - Nifti
  - GE

Registration
- Rigid
- Header
  - v3
  - v2
- Modality
  - PET
  - MRI
- Format
  - Nifti

Registration
- Header
  - v3
  - v2
  - v1
- Modality
  - PET
  - CT
  - MRI
- _
  - Affine
  - Rigid
- Format
  - DICOM
  - Nifti
  - GE

Constraints
- GE ⇒ MRI
- GE ⇒ ¬v3
- CT ⇒ ¬v2
- Rigid ⇒ Nifti
- DICOM ⇒ CT
- v1 ⇒ ¬Rigid
- CT ⇒ ¬Rigid
- v1 ⇒ ¬PET
- PET ⇒ ¬Affine

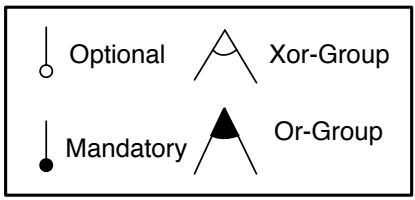
Yes!

Can suppliers provide *all* products?

“compare”

Registration
- Header
  - v1
- Modality
  - CT
  - MRI
- Format
  - DICOM
  - Nifti

Constraints
- DICOM ⇒ ¬MRI

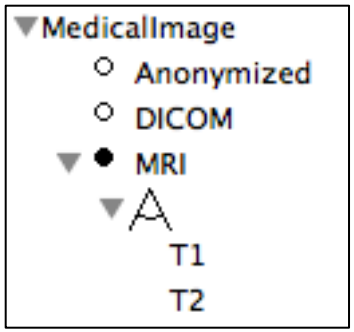


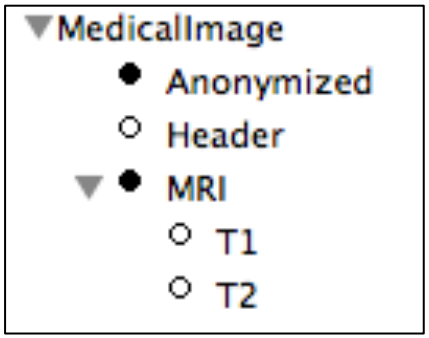
suppliersExample.fml

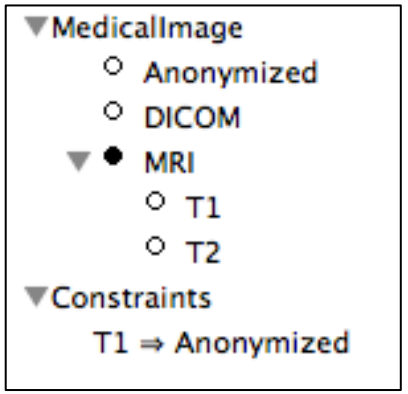
In FAMILIAR

```
REGsupp1 = FM (Registration : Header Format Modality [Affine] ;
                                                    Header : (v1|v3);
                                                    Format : (DICOM|Nifti) ;
                                                    Modality : CT; )

REGsupp2 = FM (Registration : Header [Affine] Format Modality ;
                                                    Header : (v1|v2);
                                                    Format : (Nifti|GE) ;
                                                    Modality : MRI; )

REGsupp3 = FM (Registration : Header [Rigid] Format Modality ;
                                                    Header : (v2|v3) ;
                                                    Format : Nifti ;
                                                    Modality : (MRI|PET); )


REGrequired = FM (Registration : Header Format Modality ;
                                                    Header : v1 ; //v3;
                                                    Format : (DICOM|Nifti) ;
                                                    Modality: (MRI|CT);
                                                    !DICOM or !MRI;
                                                    )

REGmspl = merge sunion REGsupp*          // merge all FMs whose variable identifier starts w

cmp = compare REGrequired REGmspl
//missingSPL = merge diff { REGrequired REGmspl }
```

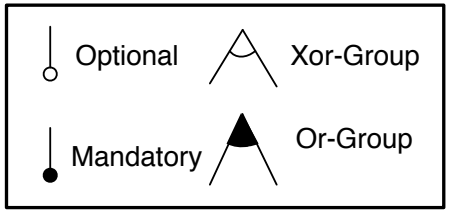
Merging operation: implementation issues

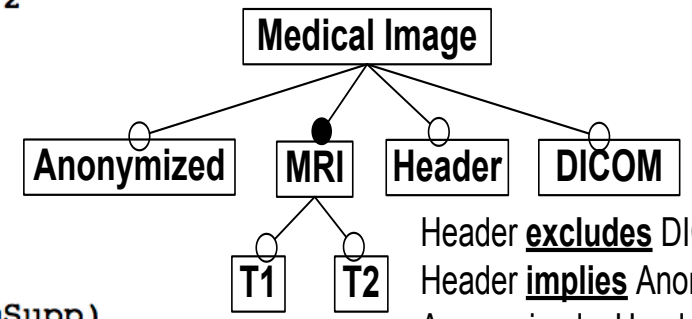







```
fmsupp1 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : (T1|T2) ;  )
fmsupp2 = FM (MedicalImage : Anonymized MRI [Header] ; MRI : [T1] [T2] ; )
fmsupp3 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : [T1] [T2] ; T1 -> Anonymized; )

// computing the union of sets of configurations like this is COSTLY
s1 = configs fmsupp1
s2 = configs fmsupp2
s3 = configs fmsupp3

s123 = setUnion s3 setUnion s1 s2

// you WONT scale
//...

fmSupp = merge sunion fmsupp*

assert (size s123 eq counting fmSupp)
```





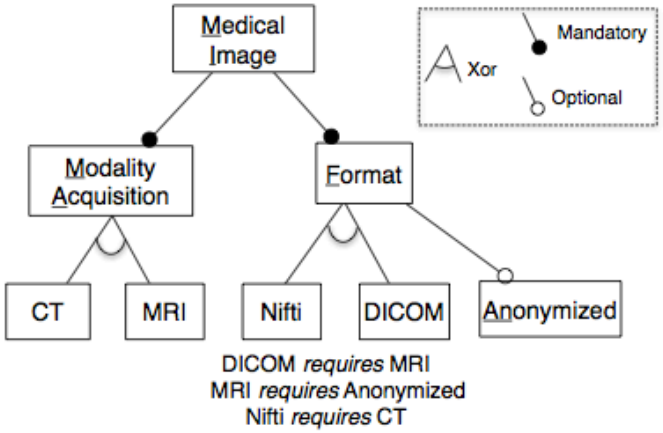
Header **excludes** DICOM
Header **implies** Anonymized
Anonymized v Header v ~DICOM v ~T1 v ~T2
Anonymized v Header v DICOM v ~T1 v ~T2

Merging operation: semantic issues (2)

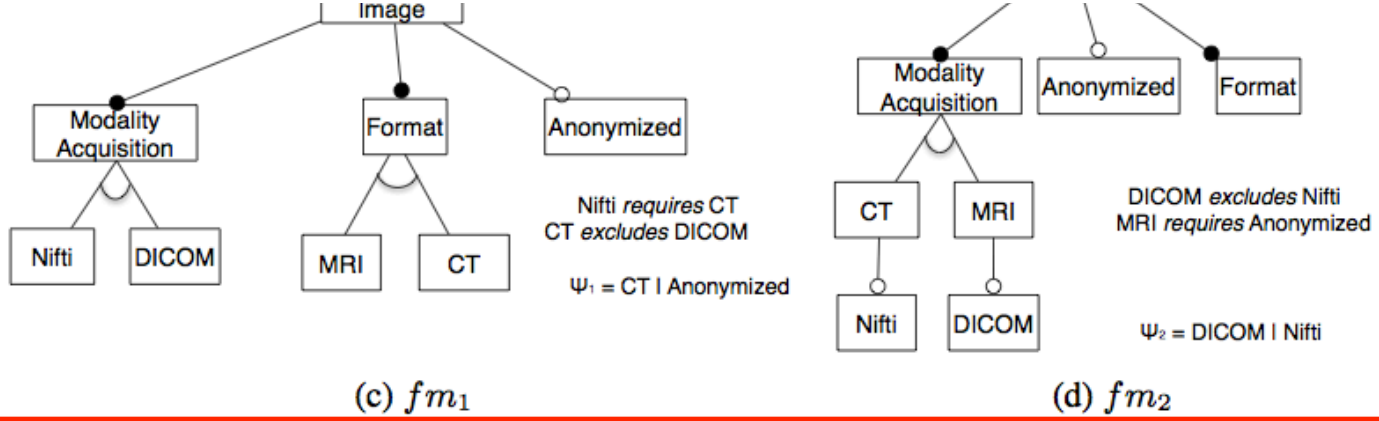
φ

$$s_0 = \{$$
$$\{MI, MA, F, CT, Nifti\},$$
$$\{MI, MA, F, CT, Nifti, AN\},$$
$$\{MI, MA, F, DICOM, MRI, AN\}$$
$$\}$$

Union
Intersection
Diff



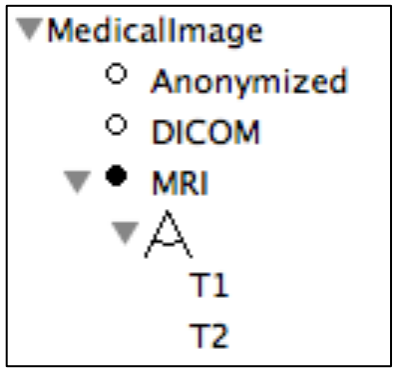
How to synthesise a feature model that represents the union of input sets of configurations?



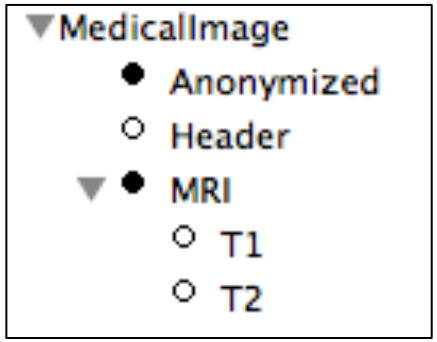
(c) fm_1 (d) fm_2

Fig. 2: For a given set of configurations, three possible yet different FMs ($s_0 = [\![fm_0]\!] = [\![fm_1]\!] = [\![fm_2]\!]$)

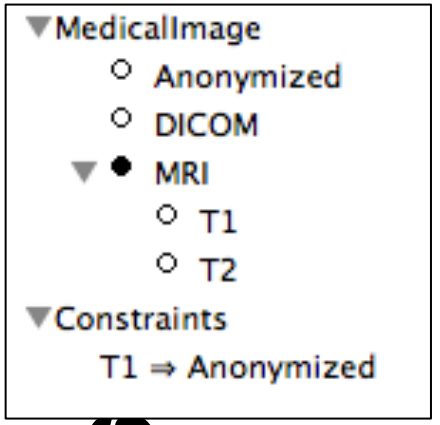
Merging operation: algorithm



φ_1



φ_2



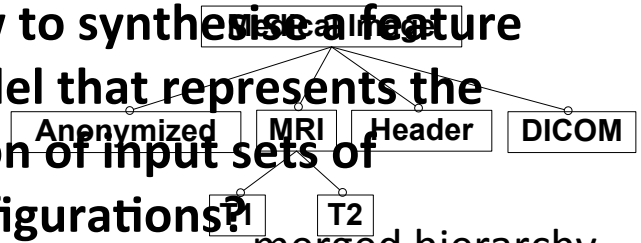
φ_3

φ_{123}

merged propositional formula

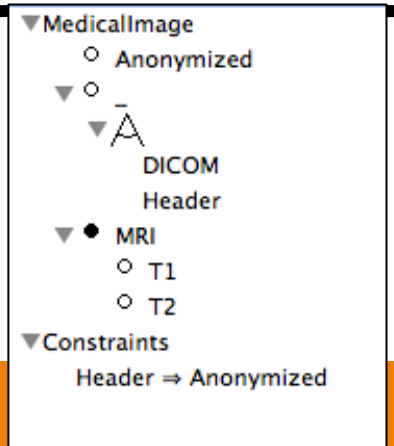
+

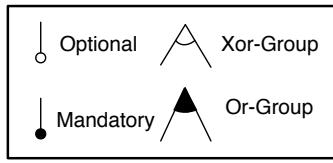
How to synthesise a feature model that represents the union of input sets of configurations?



merged hierarchy

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



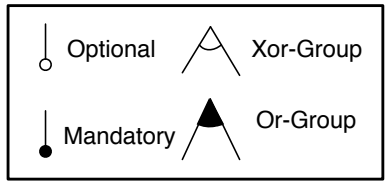


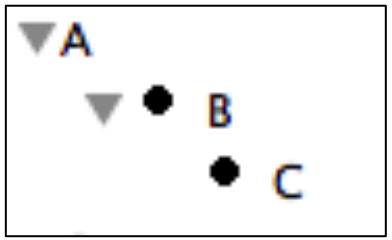
see also [Czarnecki SPLC'07 or SPLC'12]

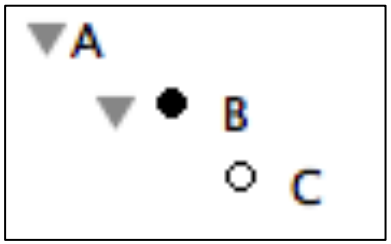
mergeNonPC.fml

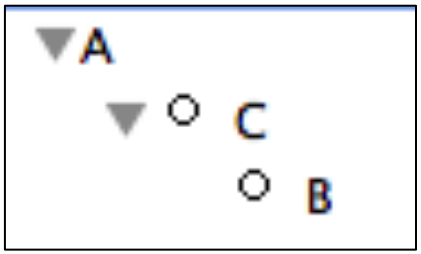
Merging operation: back to hierarchy

```
fm1 = FM (A : B ; B : C ;)
fm2 = FM (A : B ; B : [C] ;)
fm3 = FM (A : [C] ; C : [B] ; )
fm4 = merge sunion { fm1 fm2 fm3 }
```









> configs fm4
> res12: (SET) {{C;A};{A;B};{A};{A;B;C}}

?

Related Works

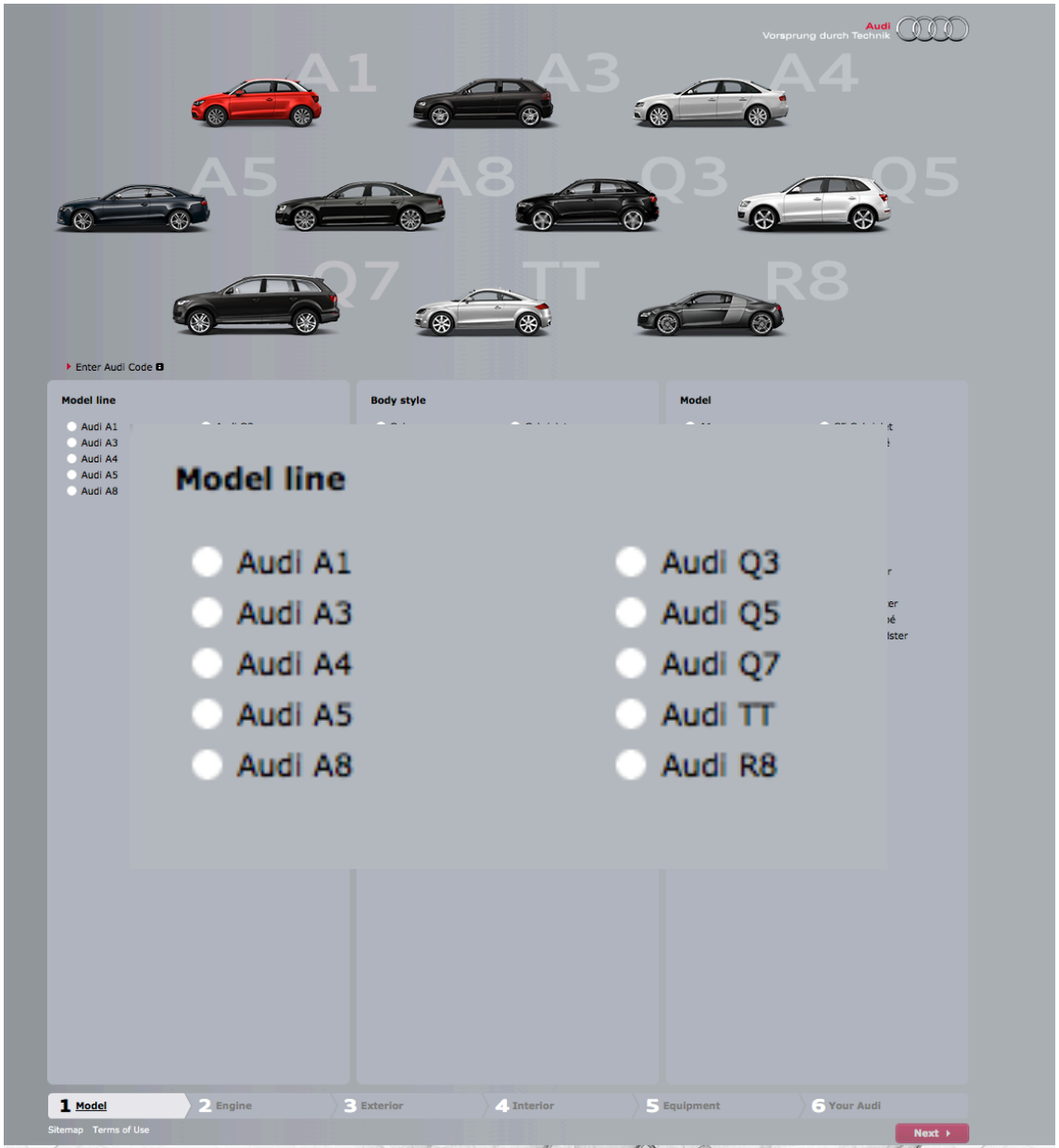
- Well-defined semantics
- Guarantee semantics properties by construction
- More compact feature models than reference-based techniques [Schobbens et al., 2007], [Hartmann et al., 2007]
  - Easier to understand
  - Easier to analyze (e.g., compare with another)
- Applicable to any propositional feature models
  - Full support of propositional constraints
  - Different hierarchies [Van Den Broek et al., SPLC'2010/2012]
- Syntactical strategies fail [Alves et al., 2006], [Segura et al., 2007]

see also [Acher et al., ECMFA'10 / MODELS'13]

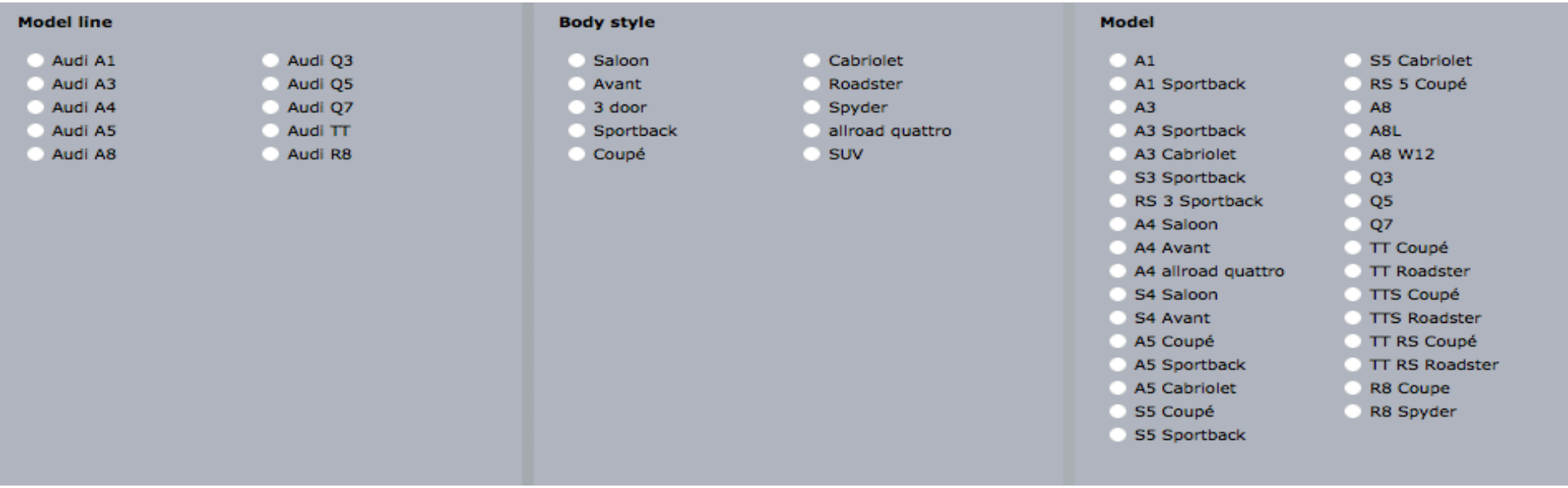
Another application of composing feature models

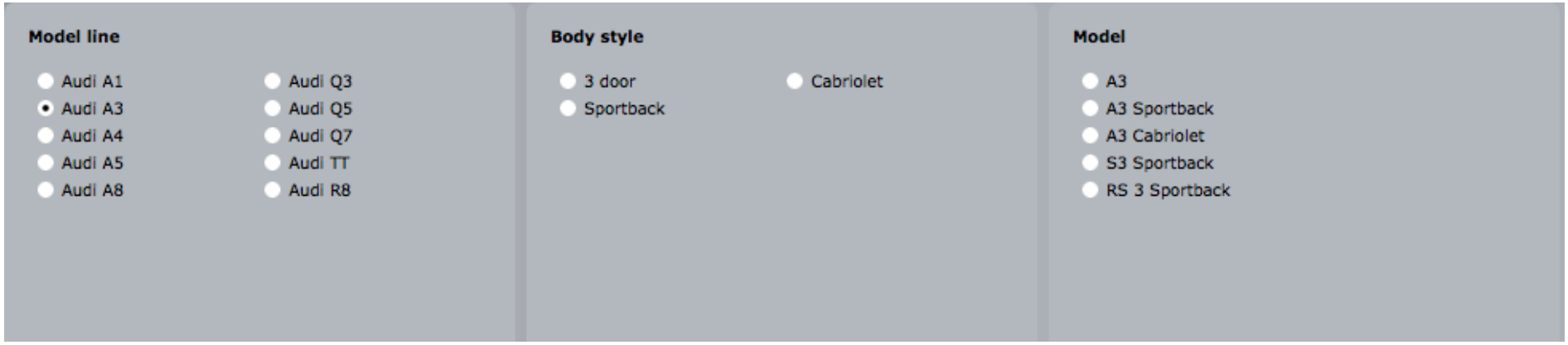
(purpose: automated synthesis of feature models)

Problem: multiple „car models“



Problem: multiple „car models“





Problem: multiple „car models“

Model line

- ○ Audi A1
- ◉ Audi A3
- ○ Audi A4
- ○ Audi A5
- ○ Audi A8
- ○ Audi Q3
- ○ Audi Q5
- ○ Audi Q7
- ○ Audi TT
- ○ Audi R8

Body style

- ○ 3 door
- ○ Sportback
- ○ Cabriolet

Model

- ○ A3
- ○ A3 Sportback
- ○ A3 Cabriolet
- ○ S3 Sportback
- ○ RS 3 Sportback

Model line

- ○ Audi A1
- ○ Audi A3
- ◉ Audi A4
- ○ Audi A5
- ○ Audi A8
- ○ Audi Q3
- ○ Audi Q5
- ○ Audi Q7
- ○ Audi TT
- ○ Audi R8

Body style

- ○ Saloon
- ○ Avant
- ○ allroad quattro

Model

- ○ A4 Saloon
- ○ A4 Avant
- ○ A4 allroad quattro
- ○ S4 Saloon
- ○ S4 Avant

Model line

- ◉ Audi A1
- ○ Audi A3
- ○ Audi A4
- ○ Audi A5
- ○ Audi A8
- ○ Audi Q3
- ○ Audi Q5
- ○ Audi Q7
- ○ Audi TT
- ○ Audi R8

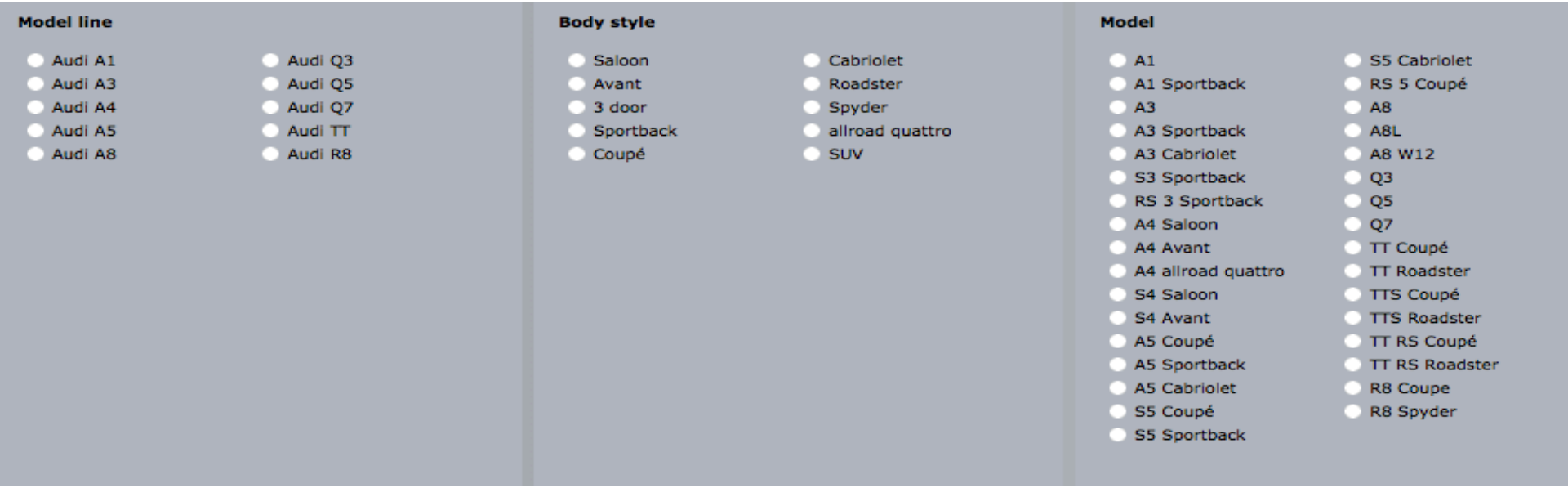
Body style

- ○ 3 door
- ○ Sportback

Model

- ○ A1
- ○ A1 Sportback

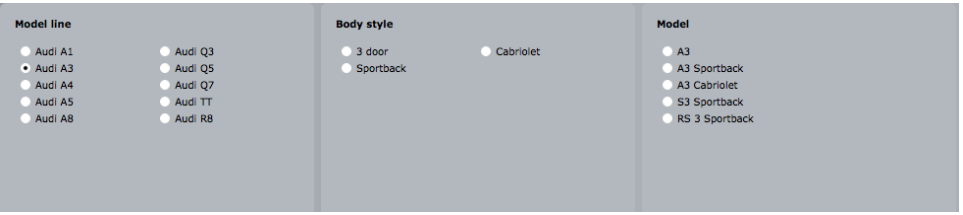
Problem: multiple „car models“

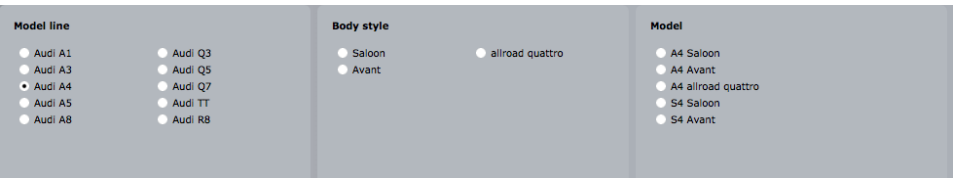


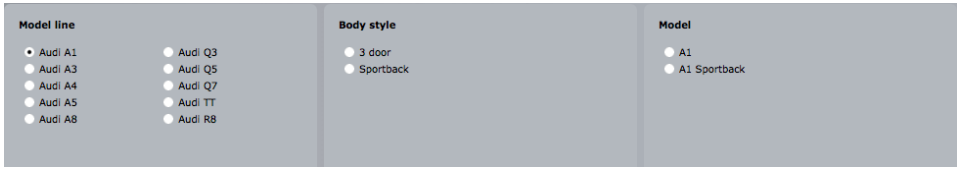
#1 – top-down: specify constraints (e.g., excludes) of all model lines upfront

Two modeling approaches

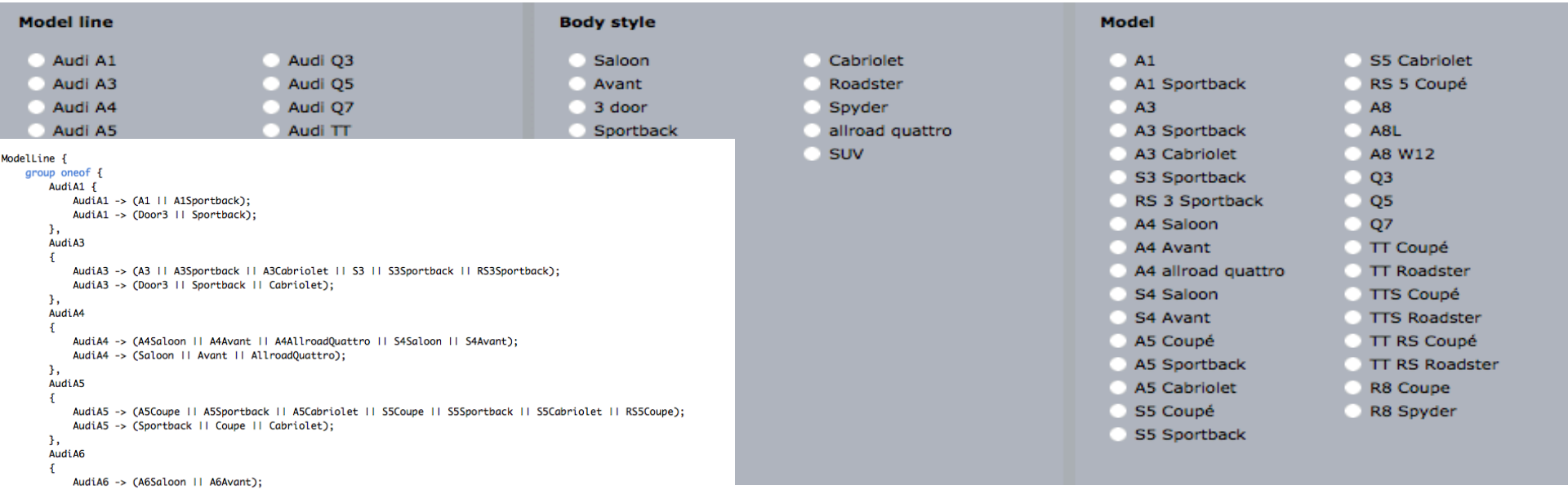
#2 – bottom-up: elaborate a feature model for each model line and merge them



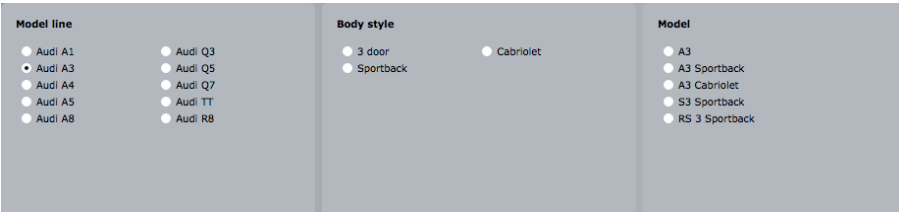


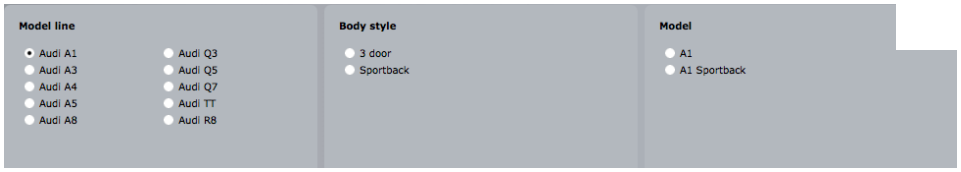


#1 top-down



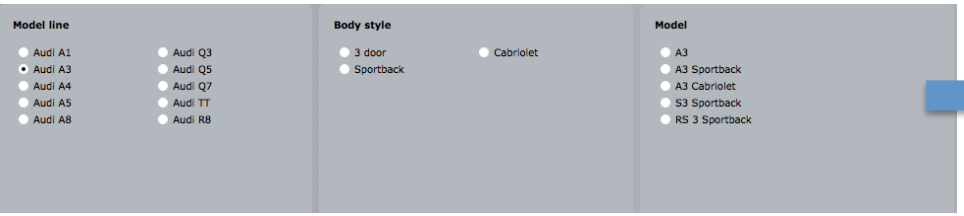

```
ModelLine {
    group oneof {
        AudiA1 {
            AudiA1 -> (A1 || A1Sportback);
            AudiA1 -> (Door3 || Sportback);
        },
        AudiA3
        {
            AudiA3 -> (A3 || A3Sportback || A3Cabriolet || S3 || S3Sportback || RS3Sportback);
            AudiA3 -> (Door3 || Sportback || Cabriolet);
        },
        AudiA4
        {
            AudiA4 -> (A4Saloon || A4Avant || A4AllroadQuattro || S4Saloon || S4Avant);
            AudiA4 -> (Saloon || Avant || AllroadQuattro);
        },
        AudiA5
        {
            AudiA5 -> (A5Coupe || A5Sportback || A5Cabriolet || S5Coupe || S5Sportback || S5Cabriolet || RS5Coupe);
            AudiA5 -> (Sportback || Coupe || Cabriolet);
        },
        AudiA6
        {
            AudiA6 -> (A6Saloon || A6Avant);
            AudiA6 -> (Saloon || Avant);
        },
```

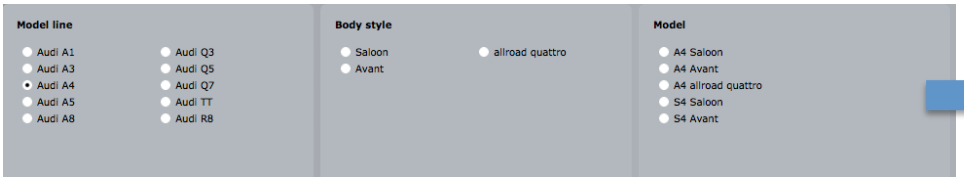


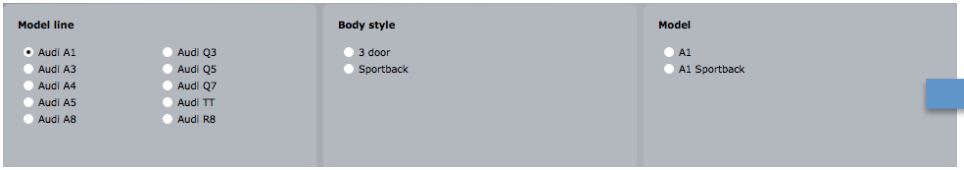



```
BodyStyle {
    group oneof {
        Saloon
        {
            Saloon -> (A4Saloon || S4Saloon || A6Saloon || A8 || A8L || A8W12);
        },
        Avant
        {
            Avant -> (A4Avant || S4Avant || A6Avant);
        },
        Door3
        {
            Door3 -> (A1 || A3 || S3);
        },
        Sportback
        {
            Sportback -> (A1Sportback || A3Sportback || S3Sportback || RS3Sportback || A5Sportback || S5Sportback || A7Spor
        },
        Coupe
        {
            Coupe -> (A5Coupe || S5Coupe || RS5Coupe || TTCoupe || TTSCoupe || TTRSCoupe || R8Coupe);
        },
```

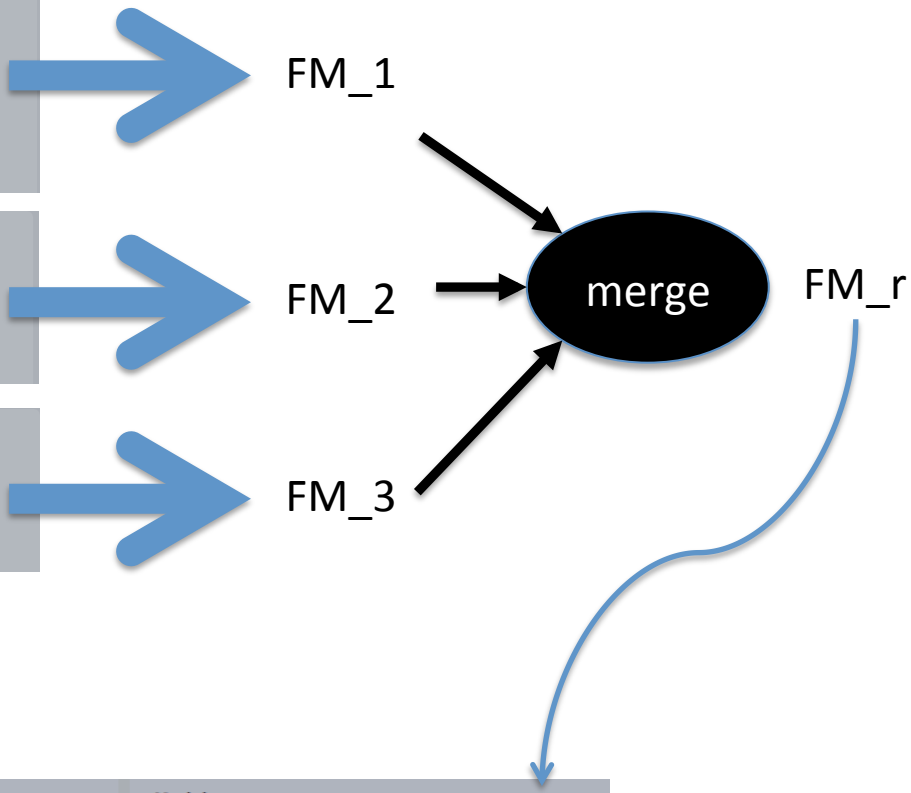
#1 bottom-up


Model line
Audi A1
Audi A3
Audi A4
Audi A5
Audi A8
Audi Q3
Audi Q5
Audi Q7
Audi TT
Audi R8
Body style
3 door
Sportback
Cabriolet
Model
A3
A3 Sportback
A3 Cabriolet
S3 Sportback
RS 3 Sportback


Model line
Audi A1
Audi A3
Audi A4
Audi A5
Audi A8
Audi Q3
Audi Q5
Audi Q7
Audi TT
Audi R8
Body style
Saloon
Avant
allroad quattro
Model
A4 Saloon
A4 Avant
A4 allroad quattro
S4 Saloon
S4 Avant

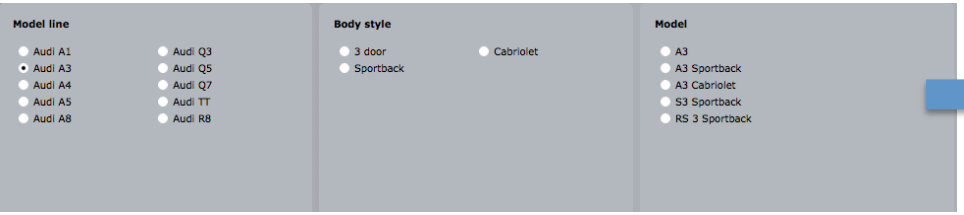

Model line
Audi A1
Audi A3
Audi A4
Audi A5
Audi A8
Audi Q3
Audi Q5
Audi Q7
Audi TT
Audi R8
Body style
3 door
Sportback
Model
A1
A1 Sportback

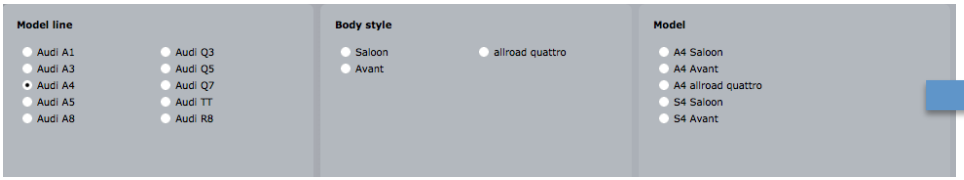

Model line
Audi A1
Audi A3
Audi A4
Audi A5
Audi A8
Audi Q3
Audi Q5
Audi Q7
Audi TT
Audi R8
Body style
Saloon
Avant
3 door
Sportback
Coupé
Cabriolet
Roadster
Spyder
allroad quattro
SUV
Model
A1
A1 Sportback
A3
A3 Sportback
A3 Cabriolet
S3 Sportback
RS 3 Sportback
A4 Saloon
A4 Avant
A4 allroad quattro
S4 Saloon
S4 Avant
A5 Coupé
A5 Sportback
A5 Cabriolet
S5 Coupé
S5 Sportback
S5 Cabriolet
RS 5 Coupé
A8
A8L
A8 W12
Q3
Q5
Q7
TT Coupé
TT Roadster
TTS Coupé
TTS Roadster
TT RS Coupé
TT RS Roadster
R8 Coupe
R8 Spyder

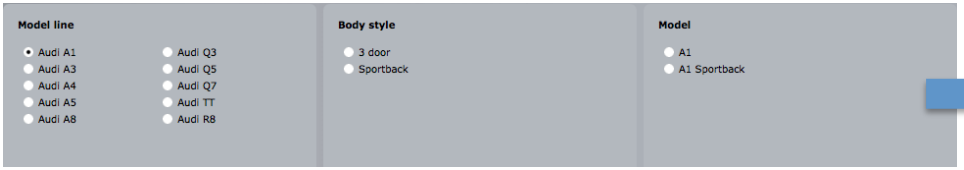

FM_1
FM_2
FM_3
merge
FM_r

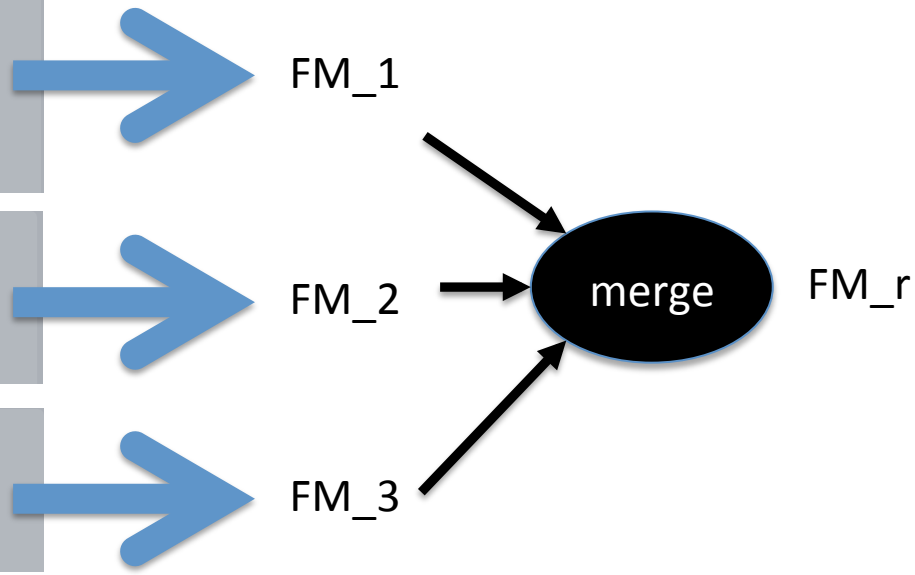
audiMerge.fml

#1 bottom-up (FAMILIAR)







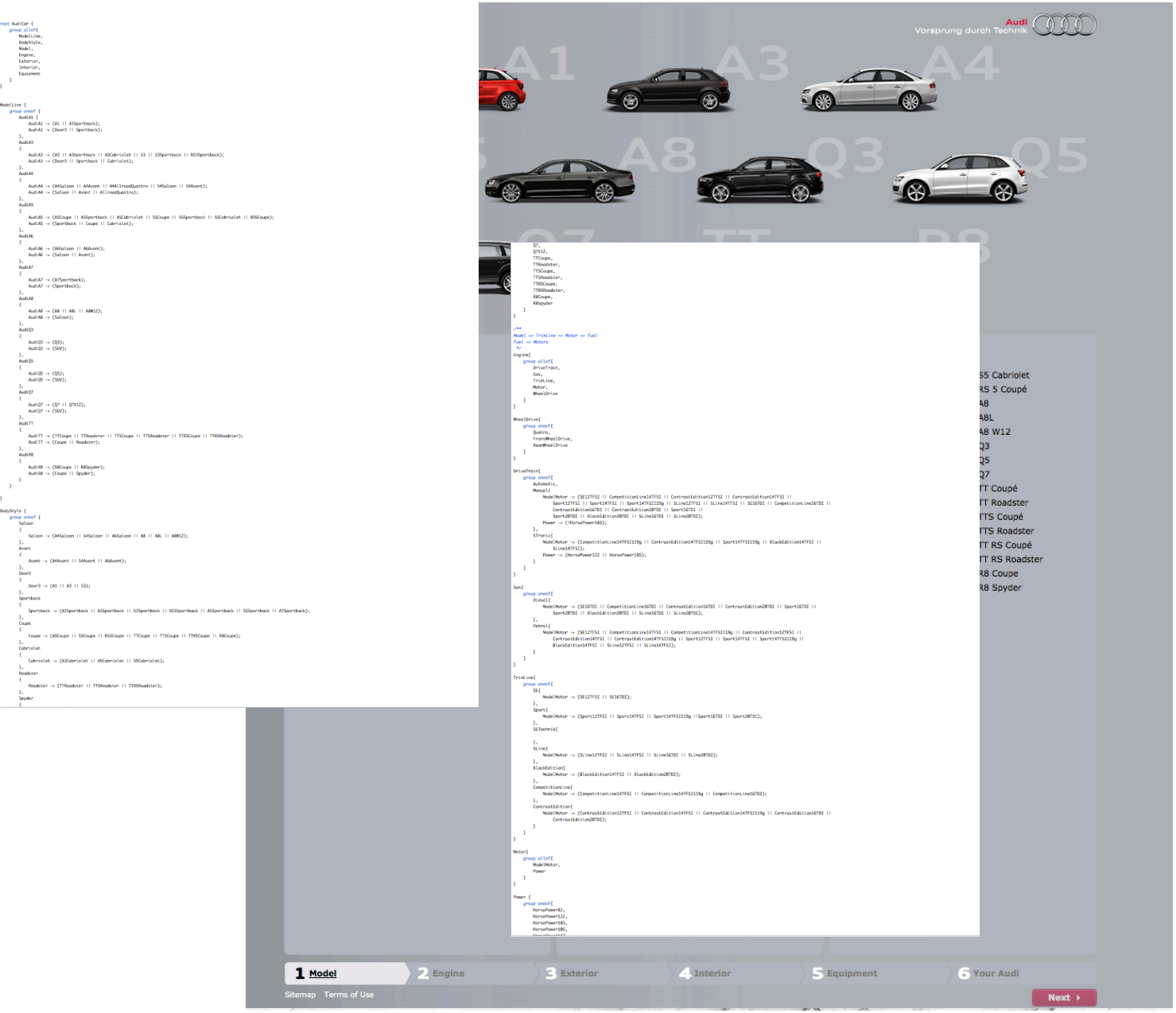



```
fml> fmAudiS = merge sunion { fm1 fm2 fm3 }
fmAudiS: (FEATURE_MODEL) Audi: ModelLine BodyStyle ;                    9.9.4 (beta)
ModelLine: (A1|A4|A3) ;
BodyStyle: (Saloon|Door3)? (Cabriolet|AllroadQuattro)? (Sportback|Avant)? ;
(A4 -> !Sportback);
(A1 -> !Avant);
(A1 -> !Saloon);
(A3 -> !Saloon);
(A4 -> !Cabriolet);
(A1 -> !AllroadQuattro);
(A1 -> !Cabriolet);
(A4 -> !Door3);
(A3 -> !Avant);
(A3 -> !AllroadQuattro);
```

Decomposition support

(and its combination with other operators)

Building “views” of a feature model

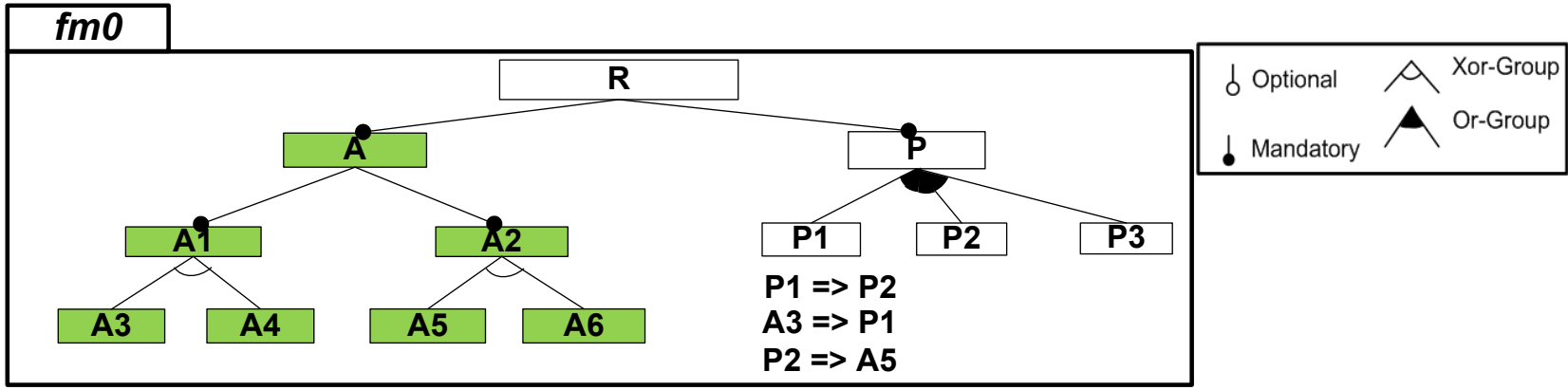


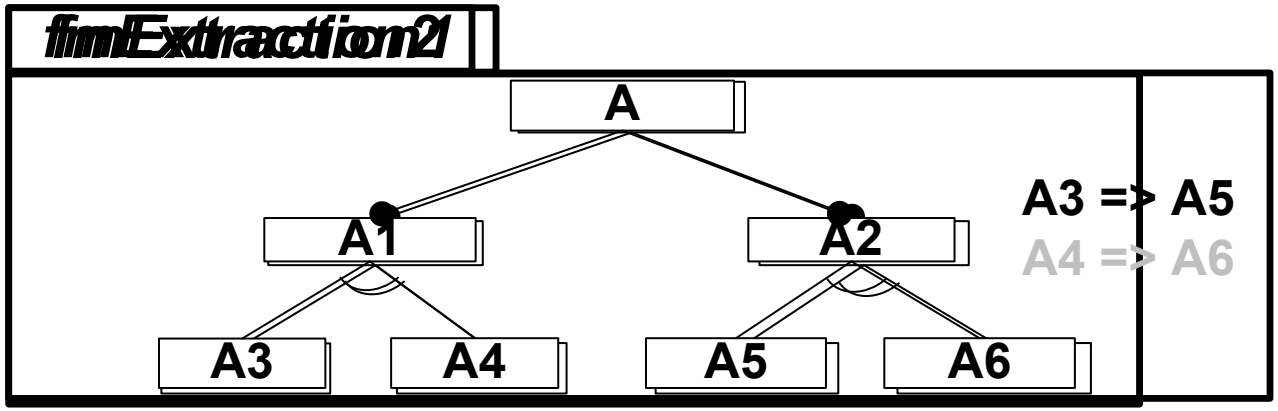
Building "views" of a feature model

- 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



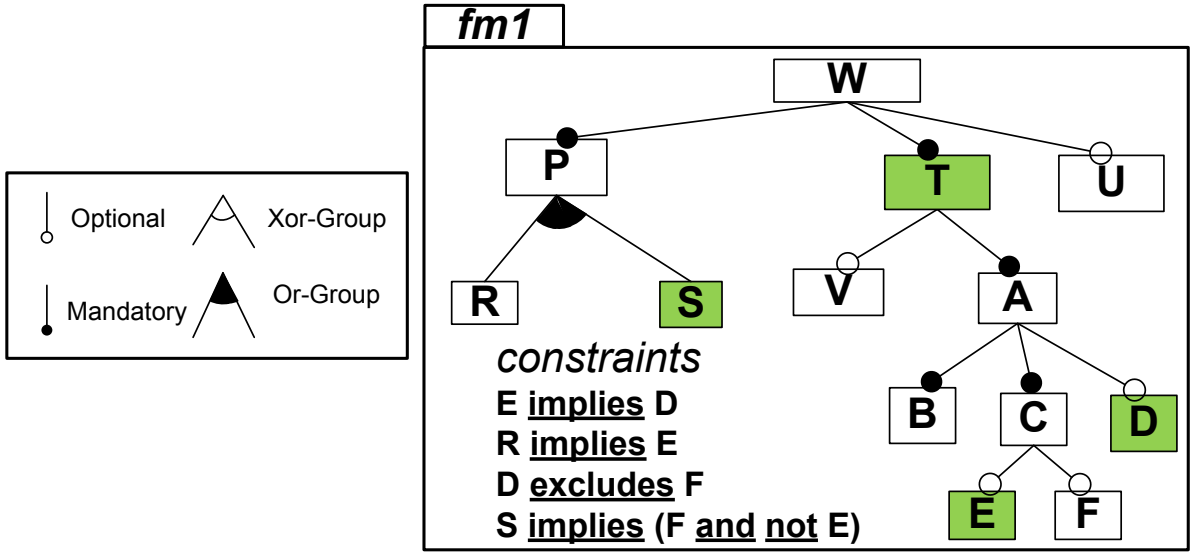


Problem: You can select **A3** without **A5**

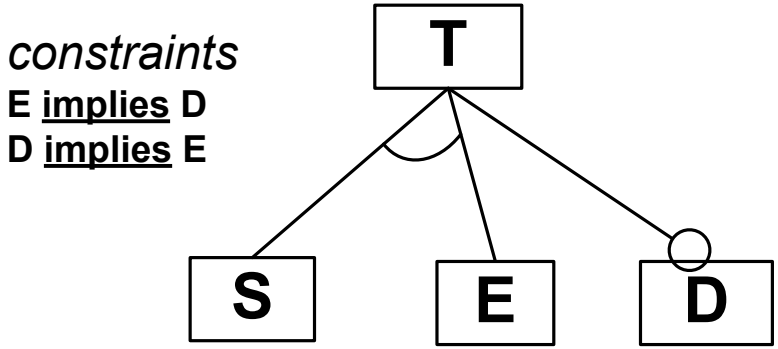
Hierarchy and Configuration matter!

Slicing Operator

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

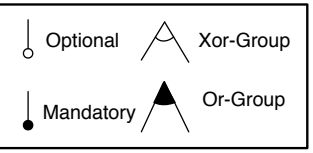


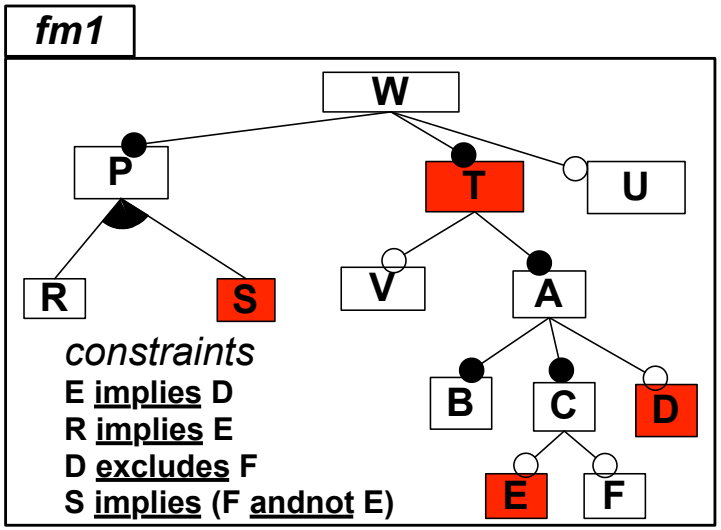
slice : a new feature model, representing a projected set of configurations



Slicing operator: going into details

projected set of configurations



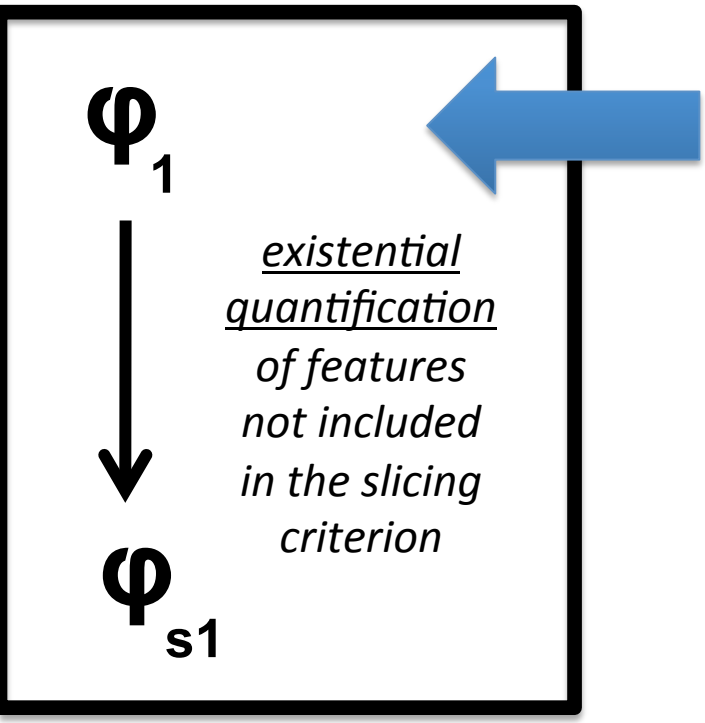


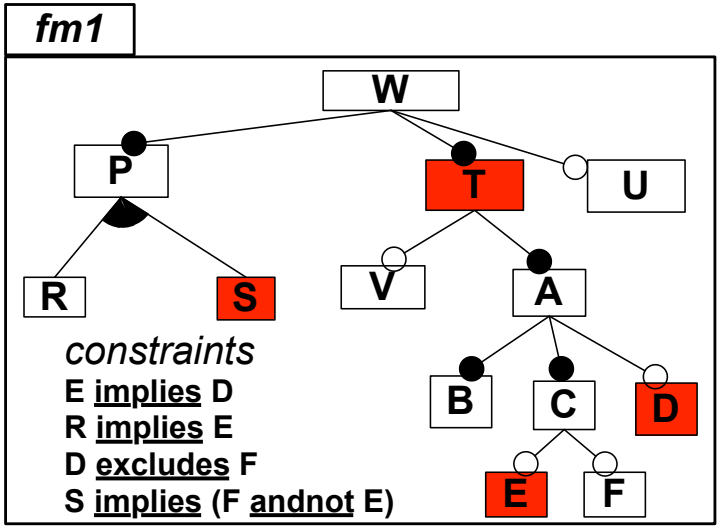
fm1 = {
{A,B,C,D,E,P,R,T,U,W},
{A,B,C,E,P,S,T,U,W},
{A,B,C,D,E,P,R,T,W},
{A,B,C,E,P,S,T,V,W},
{A,B,C,E,P,S,T,U,V,W},
{A,B,C,E,P,S,T,W},
{A,B,C,D,E,P,R,T,V,W},
} }

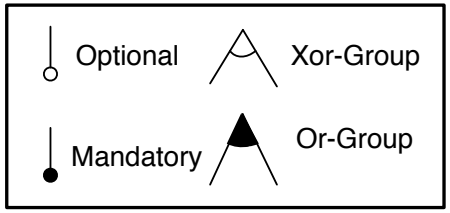
fm1p = {
{D,E,T},
fm1p = {
{S,T},
{D,E,T},
{D,E,T},
{S,T},
{S,T},
{S,T},
{S,T},
{D,E,T}
}

Slicing operator: going into details

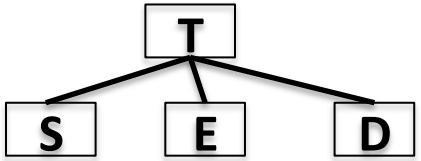
synthesizing the corresponding feature model



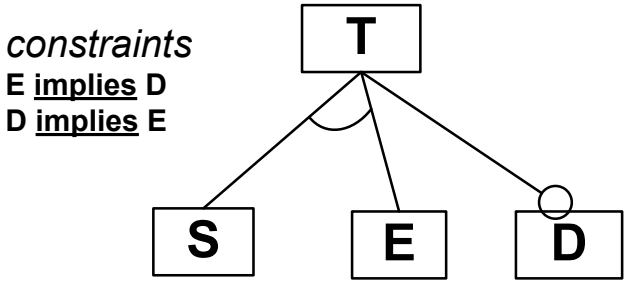




+



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

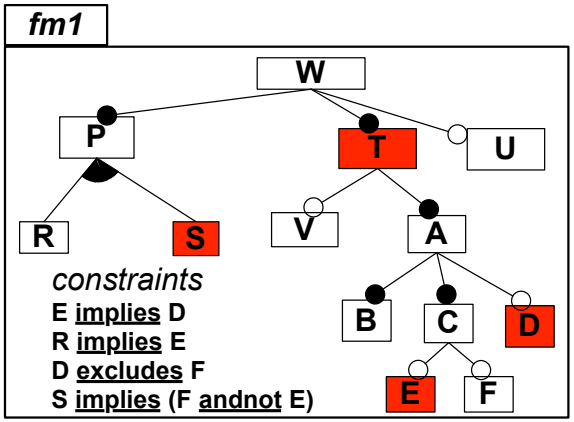


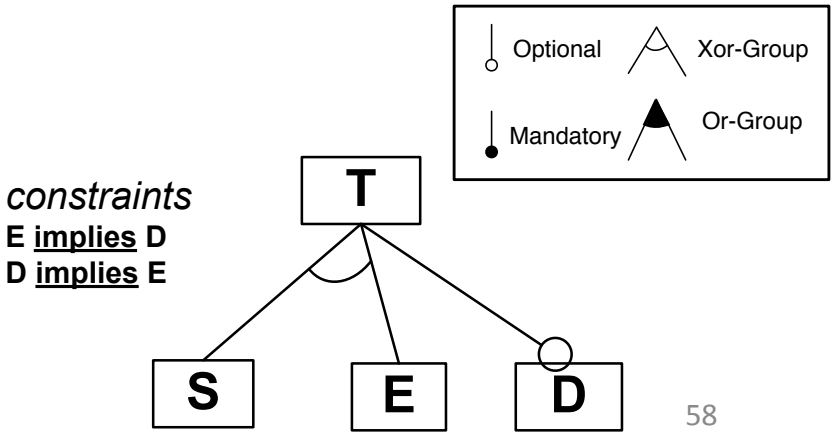
Mathieu Acher, Philippe Collet, Philippe Lahire, Robert B. France « Separation of Concerns in Feature Modeling: Support and Applications » AOSD'12

slicingOp2.fml

Slicing operator with FAMILIAR (1)

```
fm1 = FM (W : P T [U] ; T : [V] A ;
                          A : B C [D] ;
                           C : [E] [F] ;
                           P : (R|S)+ ;
                           E implies D ; R implies E ;
                           S implies (F and !E) ; D implies !F ; )

fm2 = slice fm1 including { S T E D }
fm2bis = slice fm1 excluding { W P R V A B C F U }

cmp = compare fm2 fm2bis
assert (cmp eq REFACTORING)
```





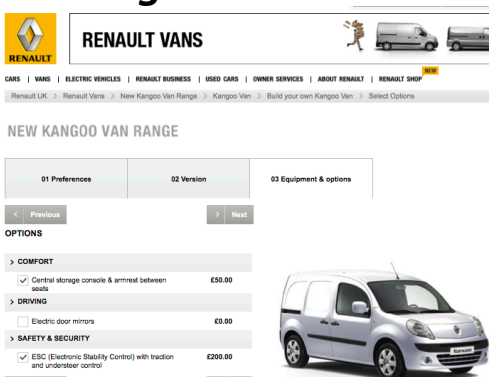
slicingOp.fml

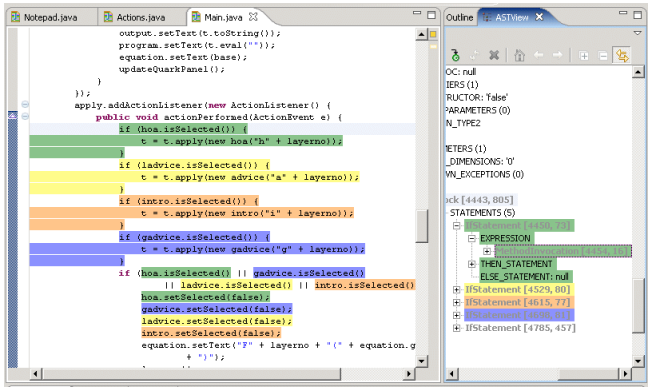
Slicing with FAMILIAR (2)

```
fm1 = FM (W : P T [U] ; T : [V] A ;
                          A : B C [D] ;
                           C : [E] [F] ;
                           P : (R|S)+ ;
                           E implies D ; R implies E ;

                           S implies (F and !E) ; D implies !F ; )

fm2 = slice fm1 including fm1.A.* ++ { fm1.A }

fm3 = slice fm1 including fm1.P.* ++ { fm1.P }
//fm3bis = slice fm1 including { fm1.P fm1.R fm1.S } // equivalent to fm3

fm4 = slice fm1 including { fm1.E fm1.D fm1.F }

fts5 = { fm1.P fm1.W } ++ fm1.P.*
fm5 = slice fm1 including fts5
```

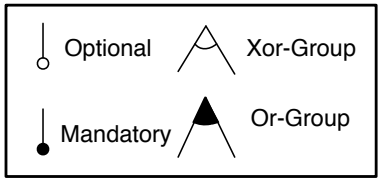
From marketing, customers, product management

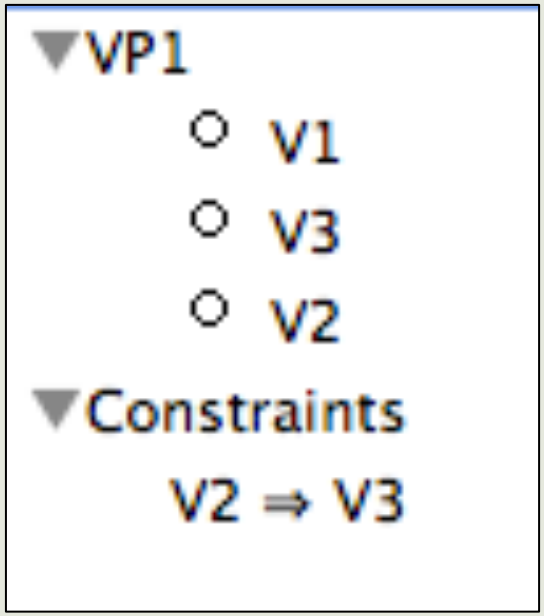
From existing software assets (technical variability)





Metzger, Heymans et al. "Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis" (RE'07)



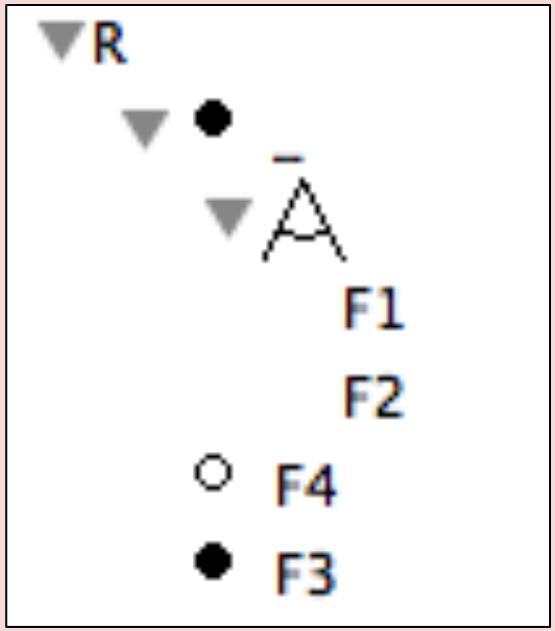


From marketing, customers, product management

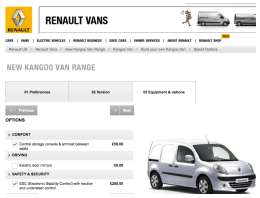
usefulness

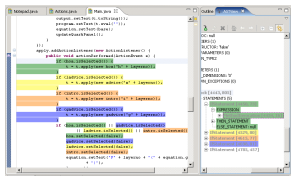
V1 ⇔ f1
V2 ⇔ f2
V3 ⇔ f3

realizability

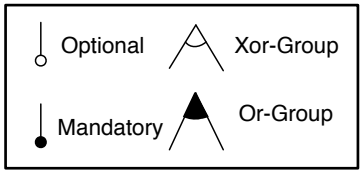


From existing software assets

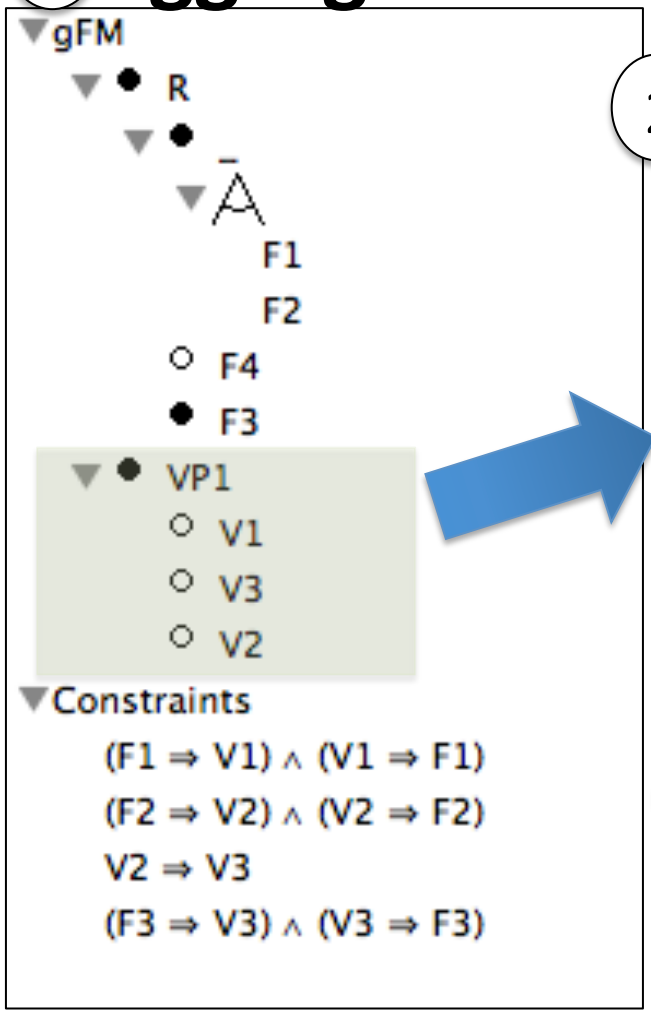




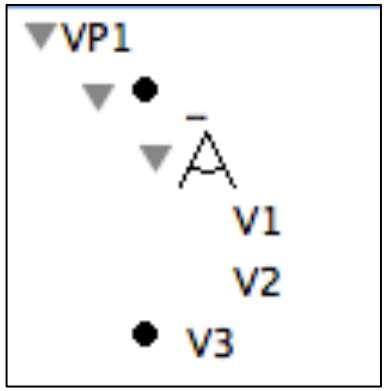
Realizability checking



1 **aggregate**



2 **slice** (“realizable part”)



φ

3 **compare**

{{V1,V3,V2,VP1},
{V1,VP1},
{V3,VP1},
{VP1}}

4 **merge diff**

(“unrealizable products”)

Mathieu Acher, Philippe Collet, Philippe Lahire, Robert B. France « Separation of Concerns in Feature Modeling: Support and Applications » AOSD’12

realizibility.fml

With FAMILIAR

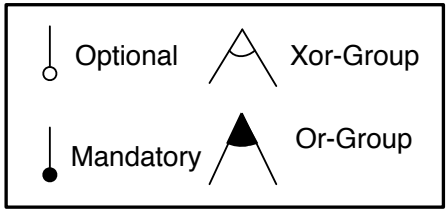
```
/*
* Metzger et al. 2007, RE'07
* Disambiguating the .....
* Figure 1, Section 3
*/
```

```
fmSoftware = FM (R : (F1|F2) F3 [F4] ; )
```

```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.6.jar realizability.fml
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 0.9.9.6
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(FEATURE_MODEL) gFM
(FEATURE_MODEL) fmSoftware
(FEATURE_MODEL) fmPLDiff
(FEATURE_MODEL) fmPLPrime
(FEATURE_MODEL) fmPL
(SET) xLink
fml> configs fmPLDiff
res1: (SET) {{VP1;V1};{VP1;V3};{VP1};{V3;V1;VP1;V2}}
```

Advanced topics

Revisiting Merge: Aggregate + Slice



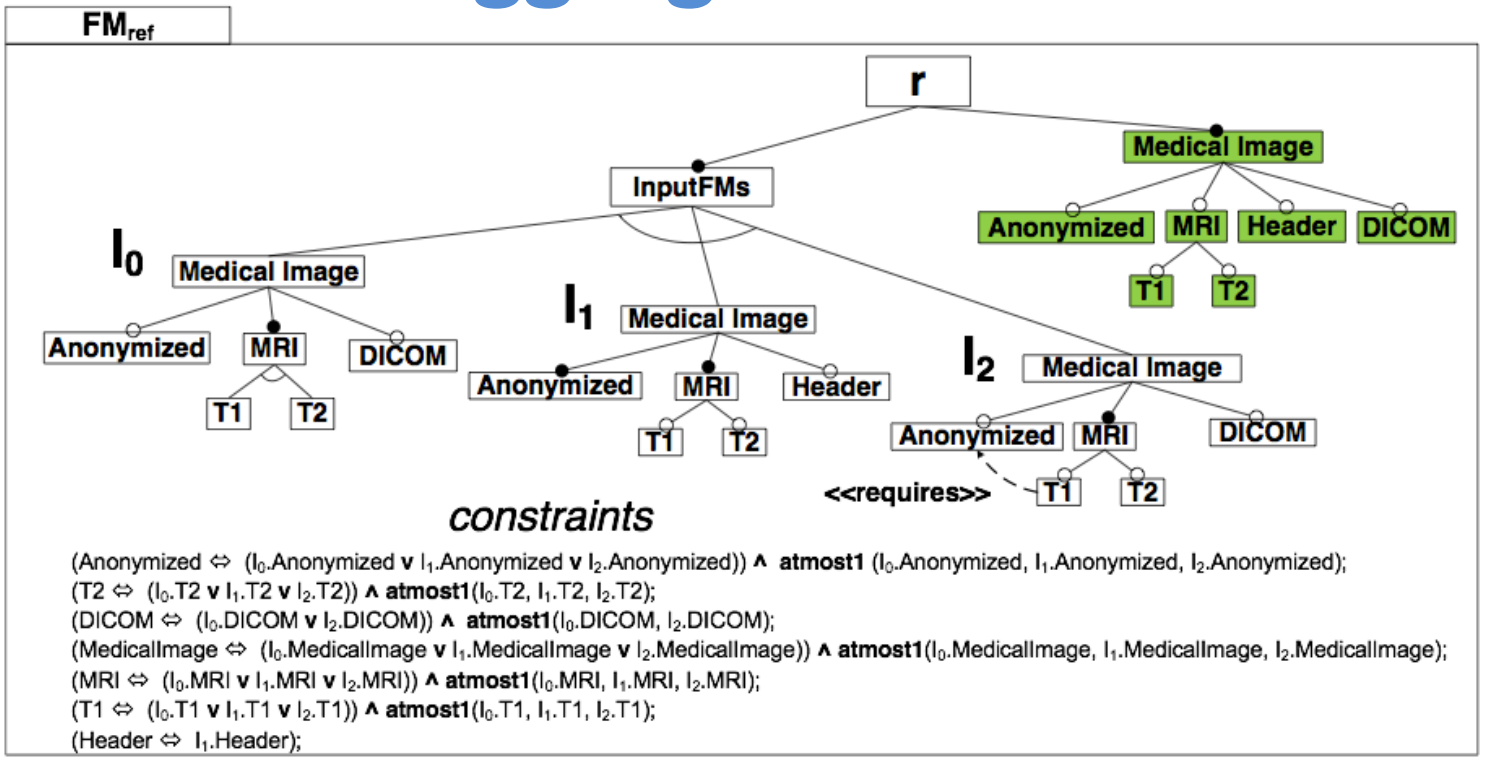
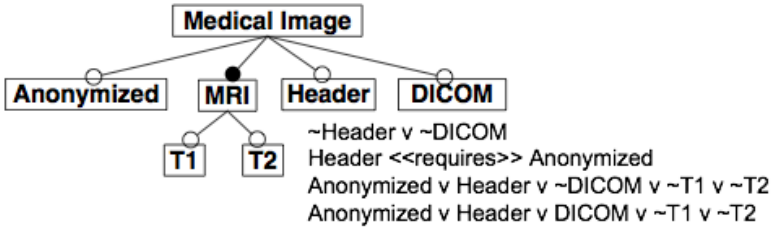


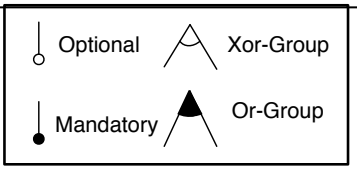
Figure 7.10: Merge of three feature models, I_0, I_1 and I_2 using the slicing operator



mergeWithAggregateMI.fml

Revisiting Aggregate, Merge and Slice:

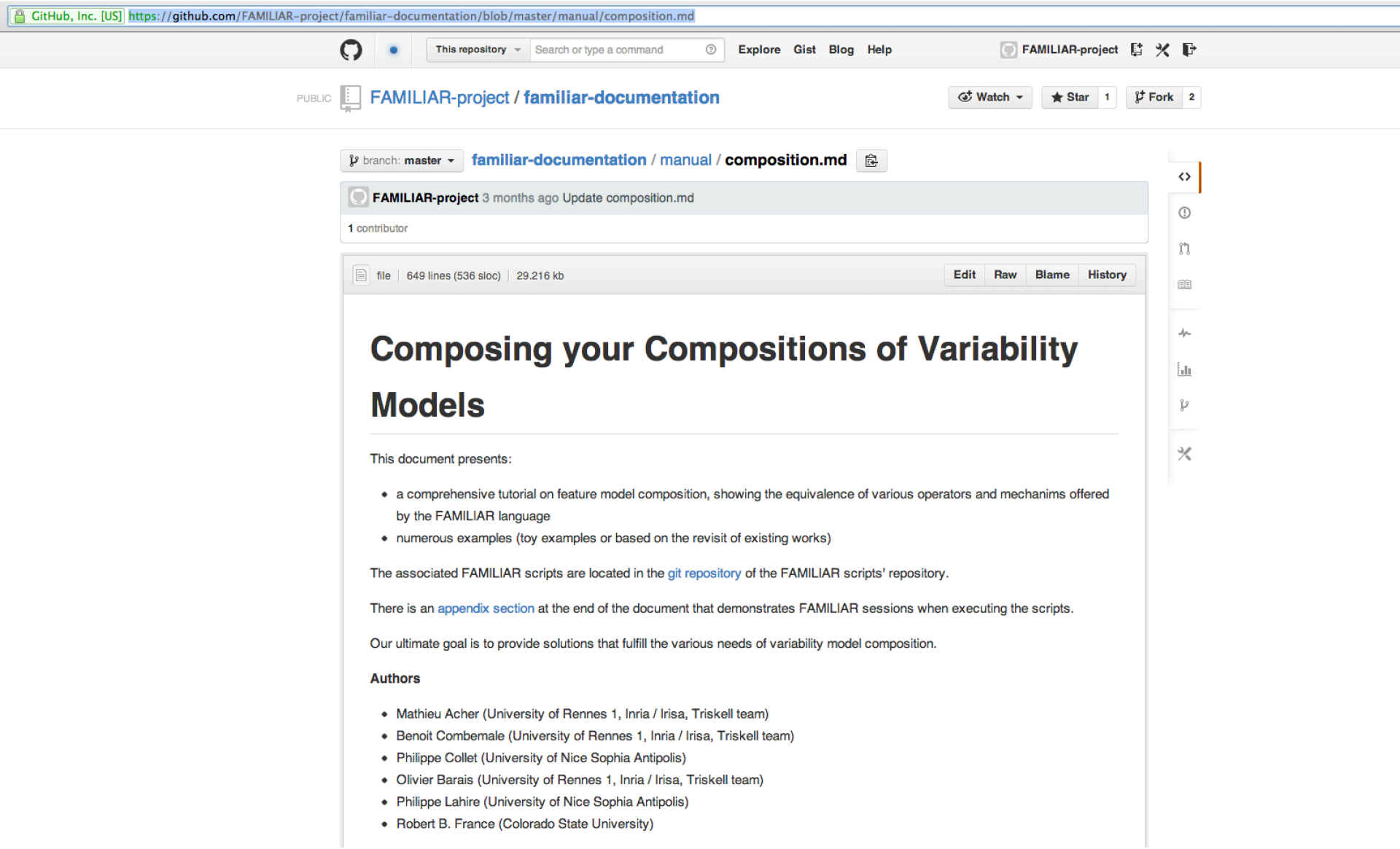
```
fmsupp1 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : (T1
fmsupp2 = FM (MedicalImage : Anonymized MRI [Header] ; MRI : [T1]
fmsupp3 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : [T1

fmSupp = merge sunion fmsupp*
fmSuppAgg = aggregateMerge sunion fmsupp*

fmSuppAggSlice = slice fmSuppAgg including fmSupp.*
```




```
fml> compare fmSuppAggSlice fmSupp
res4: (STRING) REFACTORING
```

- MergeCST
  - MedicalImage
    - DICOM
    - Anonymized
    - Header
    - MRI
      - T1
      - T2
  - InputFMs
    - MedicalImage1000
      - DICOM1000
      - Anonymized1000
      - MRI1000
        - T11000
        - T21000
    - MedicalImage1001000
      - Anonymized1001000
      - MRI1001000
        - T21001000
        - T11001000
      - DICOM1001000
    - MedicalImage2001000
      - MRI2001000
        - T12001000
        - T22001000
      - Header2001000
      - Anonymized2001000
- Constraints
  - (Anonymized1000 ∨ Anonymized1001000 ∨ Anonyr
  - (MedicalImage2001000 ∨ MedicalImage1000 ∨ Medi
  - (DICOM1001000 ∨ DICOM1000 ⇒ DICOM) ∧ (DICOM
  - (T21000 ∨ T21001000 ∨ T22001000 ⇒ T2) ∧ (T2 ⇒
  - T11001000 ⇒ Anonymized1001000
  - (Header2001000 ⇒ Header) ∧ (Header ⇒ Header200
  - (T12001000 ∨ T11000 ∨ T11001000 ⇒ T1) ∧ (T1 ⇒
  - (MRI1001000 ∨ MRI2001000 ∨ MRI1000 ⇒ MRI) ∧ (M

Mathieu Acher, Benoit Combemale, Philippe Collet, Olivier Barais, Philippe Lahire, Robert B. France « Composing your Compositions of Variability Models » MODELS'13



Mathieu Acher, Benoit Combemale, Philippe Collet, Olivier Barais, Philippe Lahire, Robert B. France « Composing your Compositions of Variability Models » MODELS'13

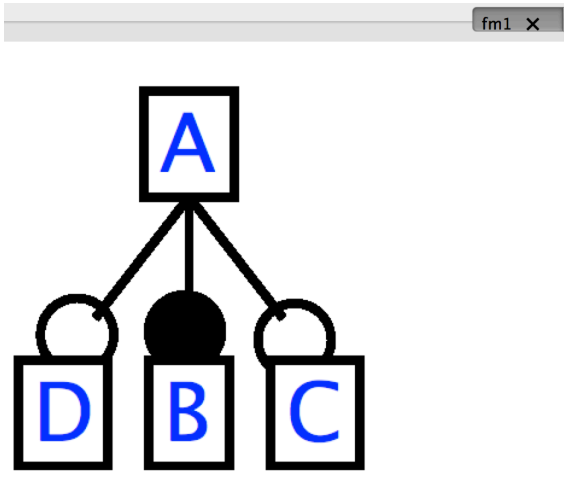
A ^
A ⇔ B ^
C => A ^
D => A

Feature Model Synthesis Problem

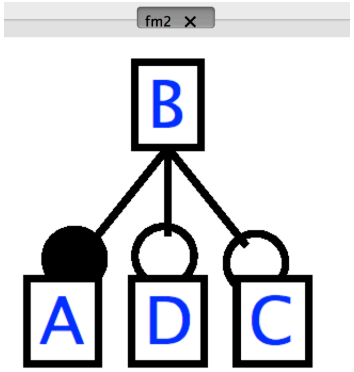
[Czarnecki et al., SPLC'07]
[She et al., ICSE'11]
[Andersen et al., SPLC'12]

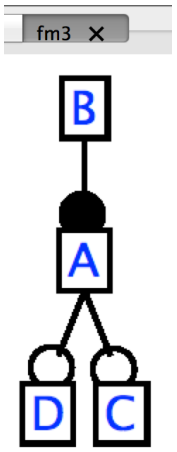
φ

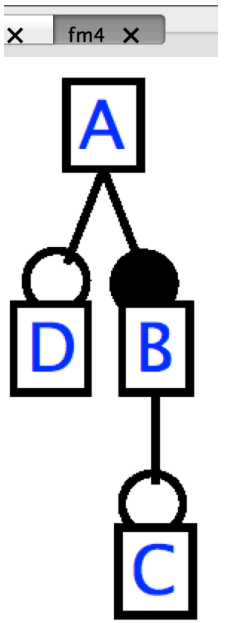
FAMiliAR



FM

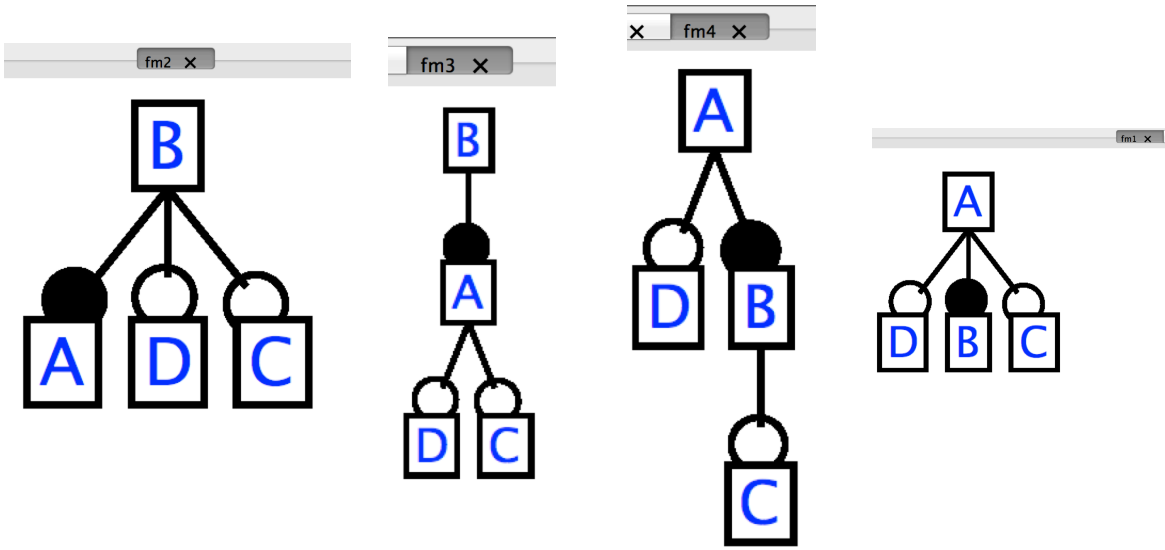







```
fm2 = FM (B : A [C] [D] ; )
fm3 = FM (B : A ; A : [C] [D] ; )
fm4 = FM (A : B [D] ; B : [C] ; )
fm5 = FM (A : B [C] ; B : [D] ; )

b12 = compare fm1 fm2
b13 = compare fm1 fm3
b14 = compare fm1 fm4
b15 = compare fm1 fm5
assert (b12 eq REFACTORING)
```

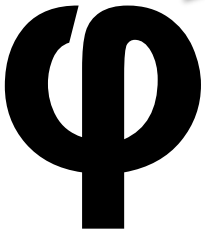
« How to synthesise an
accurate (w.r.t. the set of constraints/configurations)
meaningful (maintainable by a user), and
unique
feature model? »



http://familiar-project.github.com/

FAMiliAR

```
fm1 = FM (A : B [C] [D] ; )
fm1bis = FM ("foo3.dimacs")
fm1bisbis = FM ("foo3.constraints")

fm2 = ksynthesis fm1bis with hierarchy= A : B C D ;
fm3 = ksynthesis fm1bisbis with hierarchy= B : A ; A : C D ;
```



(SAT solvers or Binary Decision Diagrams)

The knowledge can be:

inconsistent (e.g., root feature specified is not possible)
consistent and **incomplete** (i.e., synthesis algorithm needs additional information)
consistent, **« partial »** (e.g., not all the hierarchy is specified) and actually complete

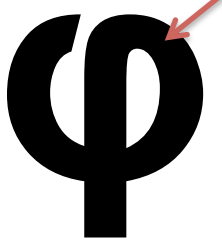
Mathieu Acher, Patrick Heymans, Anthony Cleve, Jean-Luc Hainaut, Benoit Baudry « Support for Reverse Engineering and Maintaining Feature Models » VaMoS'13

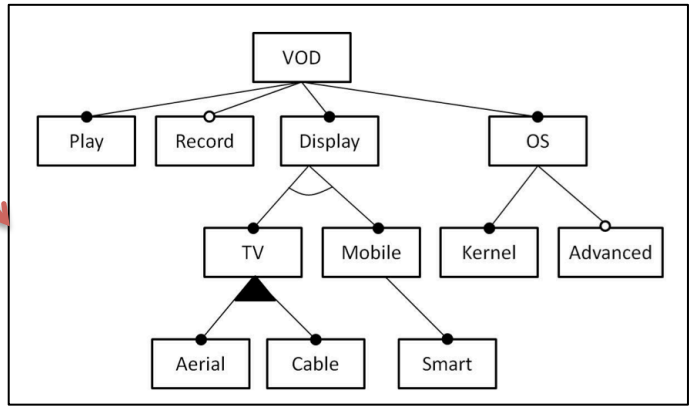
#1 Reverse Engineering Scenarios

- [Haslinger et al., WCRE'11], [Acher et al., VaMoS'12]

P	V	P	R	D	O	T	M	S	K	Ad	Ae	C
P1	✓	✓	✓	✓	✓	✓			✓	✓	✓	
P2	✓	✓	✓	✓	✓	✓			✓		✓	
P3	✓	✓		✓	✓	✓			✓	✓	✓	
P4	✓	✓		✓	✓	✓			✓		✓	
P5	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓
P6	✓	✓	✓	✓	✓	✓			✓	✓		✓
P7	✓	✓	✓	✓	✓	✓			✓		✓	✓
P8	✓	✓	✓	✓	✓	✓			✓			✓
P9	✓	✓		✓	✓	✓			✓	✓	✓	✓
P10	✓	✓		✓	✓	✓			✓	✓		✓
P11	✓	✓		✓	✓	✓			✓		✓	✓
P12	✓	✓		✓	✓	✓			✓			✓
P13	✓	✓	✓	✓	✓		✓	✓	✓	✓		
P14	✓	✓	✓	✓	✓		✓	✓	✓			
P15	✓	✓		✓	✓		✓	✓	✓	✓		
P16	✓	✓		✓	✓		✓	✓	✓			

```
// from product descriptions to feature models
// typically something generated by VariCell (see VaMoS'12 paper or the dedicated web page)
fm_1 = FM (VOD: R Ae T D P Ad K V O ; )
fm_2 = FM (VOD: R Ae T D P K V O ; )
fm_3 = FM (VOD: Ae T D P Ad K V O ; )
fm_4 = FM (VOD: T Ae D P V K O ; )
fm_5 = FM (VOD: R T Ae D P Ad K V O C ; )
fm_6 = FM (VOD: R T D P Ad V K O C ; )
fm_7 = FM (VOD: R T Ae D P V K O C ; )
fm_8 = FM (VOD: R T D P K V O C ; )
fm_9 = FM (VOD: Ae T D P Ad V K O C ; )
fm_10 = FM (VOD: T D P Ad K V O C ; )
fm_11 = FM (VOD: Ae T D P V K O C ; )
fm_12 = FM (VOD: T D P K V O C ; )
fm_13 = FM (VOD: R S D P Ad V K O M ; )
fm_14 = FM (VOD: R S D P K V O M ; )
fm_15 = FM (VOD: S D P Ad V K O M ; )
fm_16 = FM (VOD: S D P V K O M ; )

// fmR represents the union of configurations/products
// characterized by fm_1, fm_2, ..., fm_16
fmR = merge sunion fm_*

fmR2 = ksynthesis fmR with hierarchy= VOD : V P R D O ; O : K Ad ; D : T M ; T : Ae C ; M : S ;
```

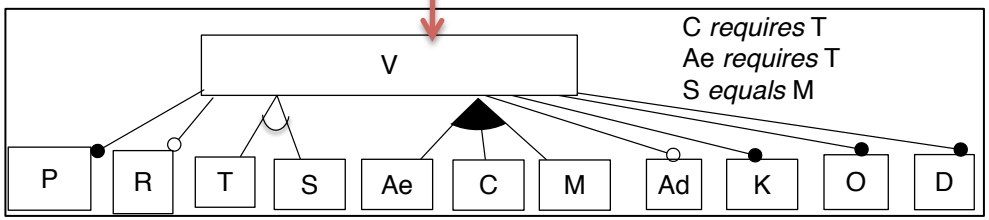


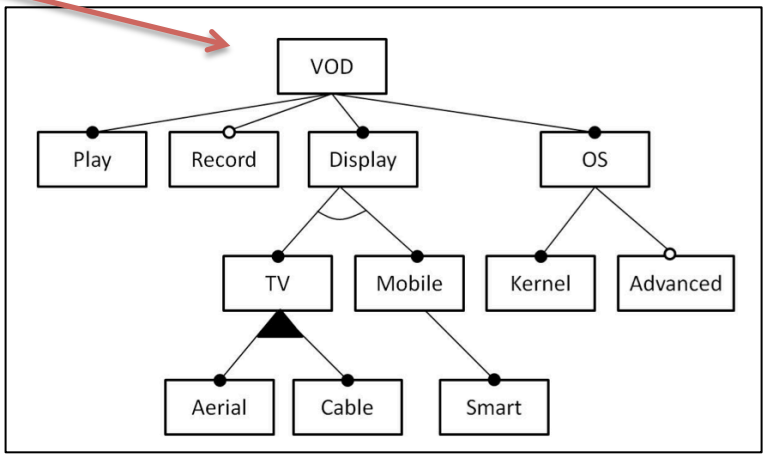


#2 Refactoring

- [Alves et al., GPCE'06], [Thuem et al., ICSE'09]

```
// refactoring!
fmR2 = ksynthesis fmR with hierarchy= VOD : V P R D O ; O : K Ad ; D : T M ; T : Ae C ; M : S ;
```





φ

```
fml> compare fmR fmR2
res0: (STRING) REFACTORING
```

#3 Re-Engineering Feature Models of S.P.L.O.T. repository

- For each FM we execute the following FAMILIAR script…

```
MacBook-Pro-de-Mathieu-3:FMLTestRepository macher1$ java -Xmx1024M -jar ../FML-1.0.3.jar
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 1.0.3 (beta)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> fm1 = FM ("output/fm1.xml")
res0: (FEATURE_MODEL) A: [D] B [C] ;
fm1: (FEATURE_MODEL) A: [D] B [C] ;
fml> fm2 = ksynthesis fm1 with hierarchy= hierarchy fm1
```

- … And we «compare» syntactically fm1 and fm2
  - semantical comparison is not needed: we know that they are refactoring by construction (good test case though ;-))
- Results:
  - 147 synthesised FMs (69 %) were exactly the same as input FMs ;
  - 40 synthesised FMs (19%) were corrections of input FMs ;
  - 24 synthesised FMs (12%) were different (knowledge needed)
    - another set of cross-tree constraints was synthesised.
    - feature group conflicts in six cases

Specification of the hierarchy is the main issue

#1 Breathing knowledge into feature model synthesis

formal specification (consistency and completeness)

concrete syntax and tooling suport

#2 Practical applications

reverse engineering, refactoring/re-engineering of feature models

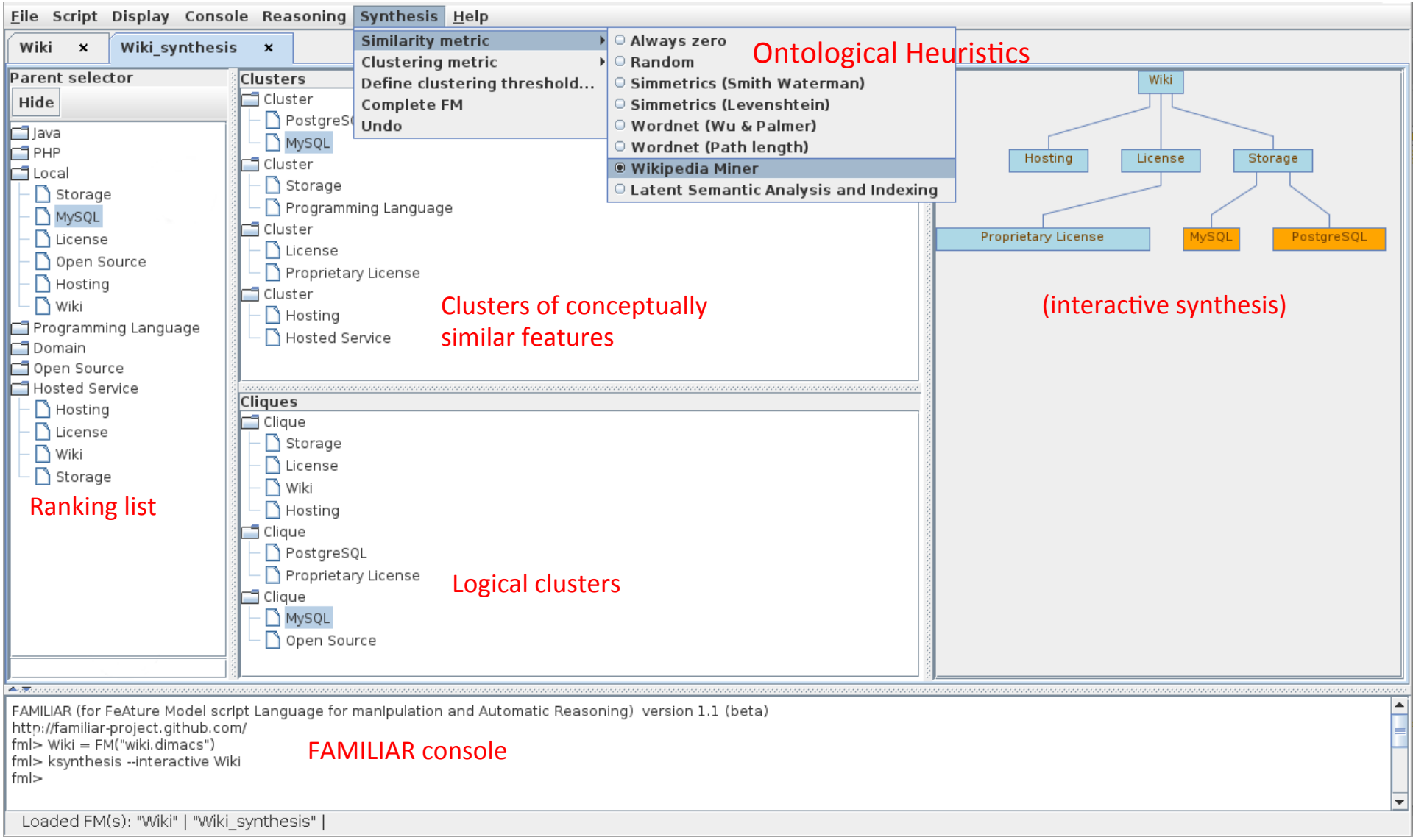
« Give me a formula
and some knowledge,

Automated support is highly needed (ongoing work)

I will synthesise
an accurate,
meaningful,
unique
feature model »

http://familiar-project.github.com/

State-of-the-art support for assisting users:
http://tinyurl.com/OntoFMExperiments



« Breathing Ontological Knowledge into Feature Model Management »
Guillaume Bécan, Mathieu Acher, Benoit Baudry, Sana Ben Nasr

300+ Products Comparison Matrices in Wikipedia

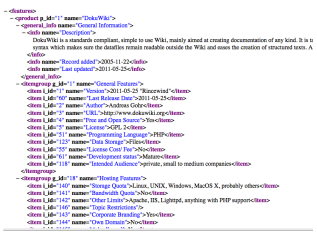
Features [edit]

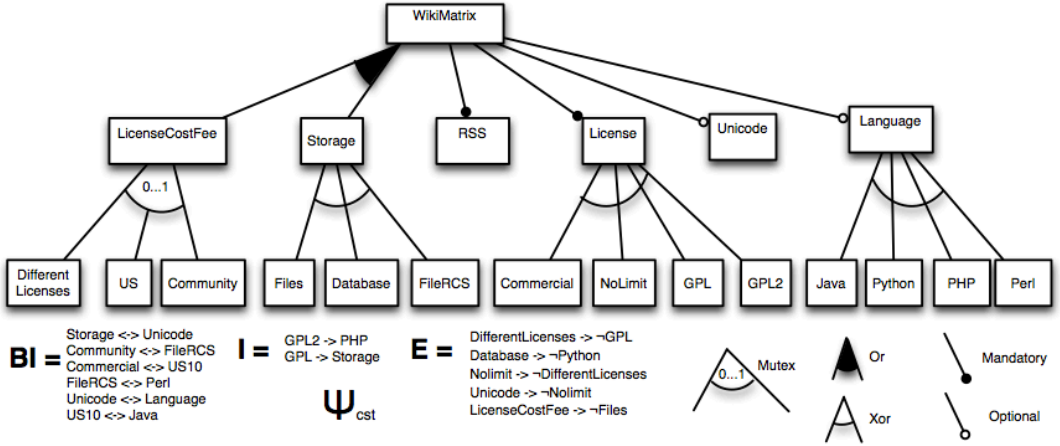
Service name	Automatic forwarding	E-mail client access[14]	client E-mail for other server	Integration with IM service	Domain Name customization	Interface script technique	Virus scanning	Filters out emails with executable attachments	Custom "From:" Address	Address modifiers (subaddressing)	Server hosted public keyring (for encryption)	Shared key encryption
AOL Mail	No	Yes (POP3, IMAP, SMTP)	Yes[9]	AOL Instant Messenger	No[1]	JavaScript/ Ajax	Yes (McAfee)	No		?	No	No
Bigfoot Communications	Premium account only	Yes (POP3, IMAP, SMTP)	Yes (POP3 only)	XMPP	Yes	HTML/ JavaScript/ CSS/Ajax	Yes	?	Yes	?	No	No
FastMail.FM	Paid accounts only	Yes (IMAP)[7]	Paid accounts (POP3, Hotmail)	XMPP	Enhanced and group (Business/ Family) accounts	HTML/ JavaScript/ CSS/Ajax (Optional user supplied custom css+JavaScript)	Paid accounts (ClamAV)	Configurable with user filtering rules	Yes Editable while composing + saved personalities	Yes userid followed by "+" followed by tag available on all accounts; subdomain addressing (e.g. <tag>@<userid> .fastmail.fm) available on paid accounts	No	No
Gmail	Yes	Yes (POP3, IMAP) SSL/TLS supported SMTP restricted[18]	Yes (POP3 only)	Google Talk[beta] (XMPP), AOL Instant Messenger	Yes (Google Apps $5.00 monthly/ $50.00 annually)	HTML/ JavaScript/ Ajax[2]	Yes	Yes, including .exe files in zip files	Yes, limited. Feature restricted to preconfigured addresses. Addresses not registered in advance with Gmail are blocked.	Yes userid followed by "+" followed by tag	No	No
GMX Mail	No	Yes (POP3, IMAP[17], SMTP) SSL/TLS supported	Yes (POP3 only)	XMPP	Yes	HTML/ JavaScript/ Ajax	Yes (McAfee and Symantec)	Yes, including .exe files in zip files	Yes	No	No	No
Hushmail	No	Extra cost[6]	?	No	$1.99/$3.99 monthly through Hushmail Business	Java or HTML	No but the text based email stops most viruses, PGP encryption	?	Yes, unlimited email aliases for Premium members	?	Yes	Yes
Mail.com	No	Yes (POP3, IMAP, SMTP) SSL/TLS supported	Yes (POP3 only)	Google Talk (XMPP)	No	HTML/ JavaScript/ Ajax[2]	Yes	Yes, including .exe files in zip files	Yes	No	No	No
Mail.ru	Yes	Yes (POP3, IMAP)	Yes (POP3 only)	custom	Yes (Free, Mail.ru for Business supports up to 5000 mailboxes)	HTML/ Ajax (Beta)	Yes (Kaspersky)	?	Yes	?	No	No
Outlook.com	Yes	Partial (POP3, SMTP)[3]	Yes (POP3 only)	Windows Live Messenger, Google Hangouts, Facebook Chat	Yes[4]	HTML/ JavaScript/ CSS/Ajax	Yes	Yes, including .exe files in zip files	Yes	userid followed by "+" followed by tag[36]	No	No
Rackspace	Yes	Yes (POP3, IMAP) SSL/TLS supported SMTP restricted	?	No	Yes	HTML/ JavaScript/ Ajax	Yes	Yes	?	No	No	No
rediff	No	Plus members only	?	Rediff Bol	Yes	JavaScript/ Ajax[2]	Yes	?	Yes	?	No	No
Runbox	Yes	Yes (IMAP, POP, SMTP) SSL/TLS supported	Yes (POP3, Hotmail, Gmail) SSL/TLS supported	XMPP, Google Talk, AOL Instant Messenger, MSN, ICQ, IRC[37]	Yes	HTML/ JavaScript/ CSS/Ajax	Yes (ClamAV)	If discarded by virus scanner	Yes Selectable while composing; saved identities	Yes userid followed by "+" or "-" followed by tag	No	No

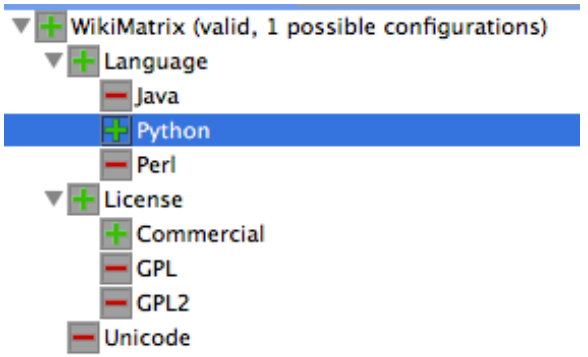
« From Comparison Matrix to Variability Model: The Wikipedia Case Study »
Nicolas Sannier, Mathieu Acher, and Benoit Baudry (ASE'2013)

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

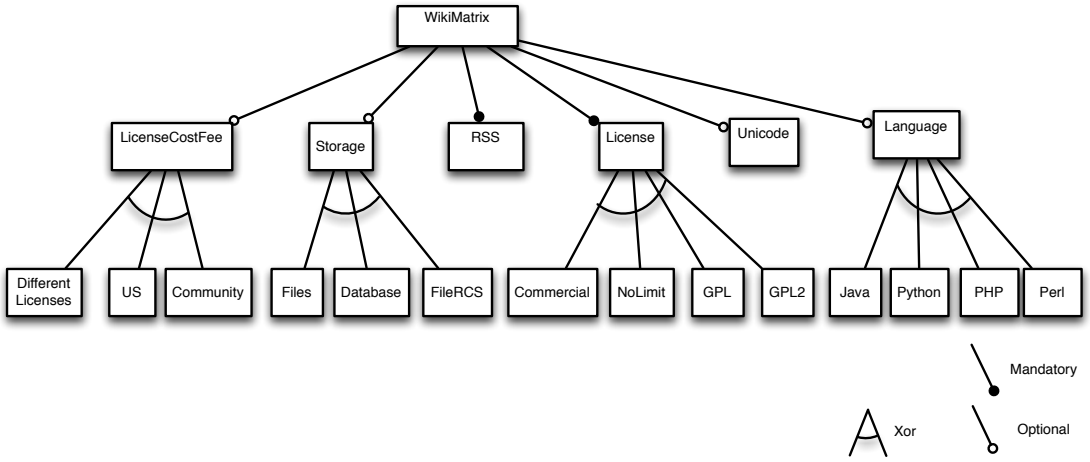






Manual extraction of a feature model from product description(s) is not possible

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

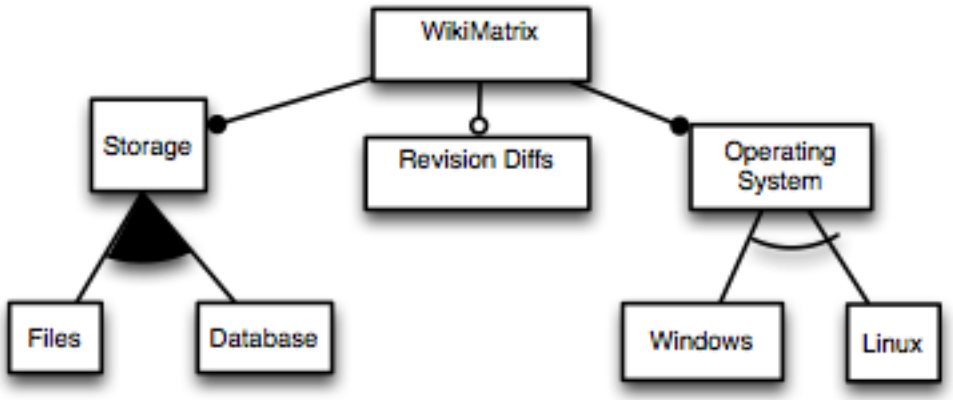
Identifier	License	Language	Storage	LicenseCostFee	RSS	Unicode	
Confluence	Commercial	Java	Database	US10	Yes	Yes	→ fm1
PBwiki	Nolimit	No	No	Yes	Yes	No	→ fm2
MoinMoin	GPL	Python	Files	No	Yes	Yes	→ fm3
DokuWiki	GPL2	PHP	Files	No	Yes	Yes	→ fm4
PmWiki	GPL2	PHP	Files	No	Yes	Yes	→ fm5
DrupalWiki	GPL2	PHP	Database	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

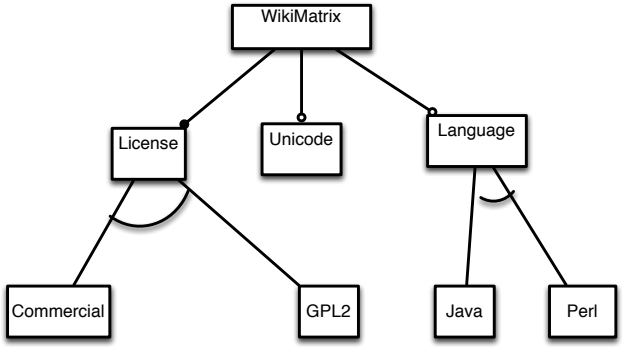
ID	Revision Diffs	Operating System	Storage	Open
...	...	...	...	...
FooWiki	Plugin	Windows; Linux	Files ; Database	No
...	...	...	...	...

(a) A wiki engine description

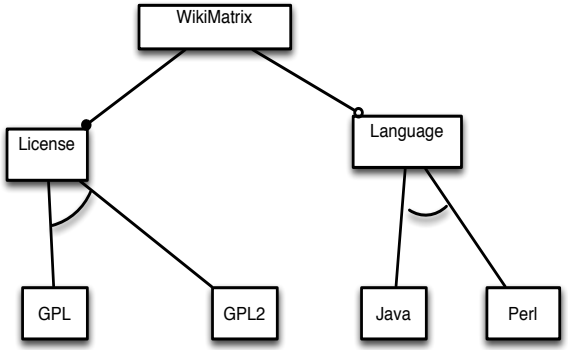


(b) Corresponding FM

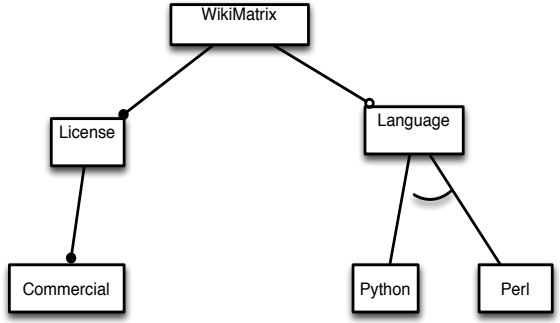
Merging of feature models



φ_1



φ_2

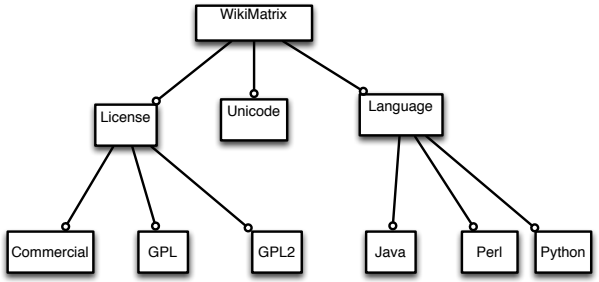


φ_3

φ_m

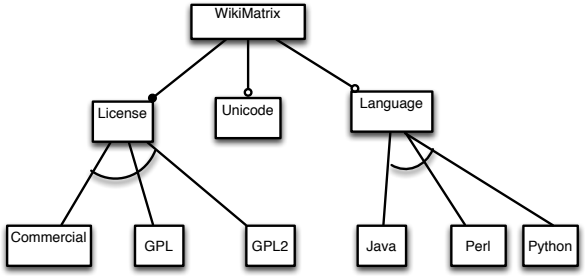
merged propositional formula

+



merged hierarchy

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



GPL excludes Unicode
Unicode excludes Python
GPL excludes Python
GPL2 excludes Python

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**

(ongoing) Comprehensive model-based product line support

Reverse engineering

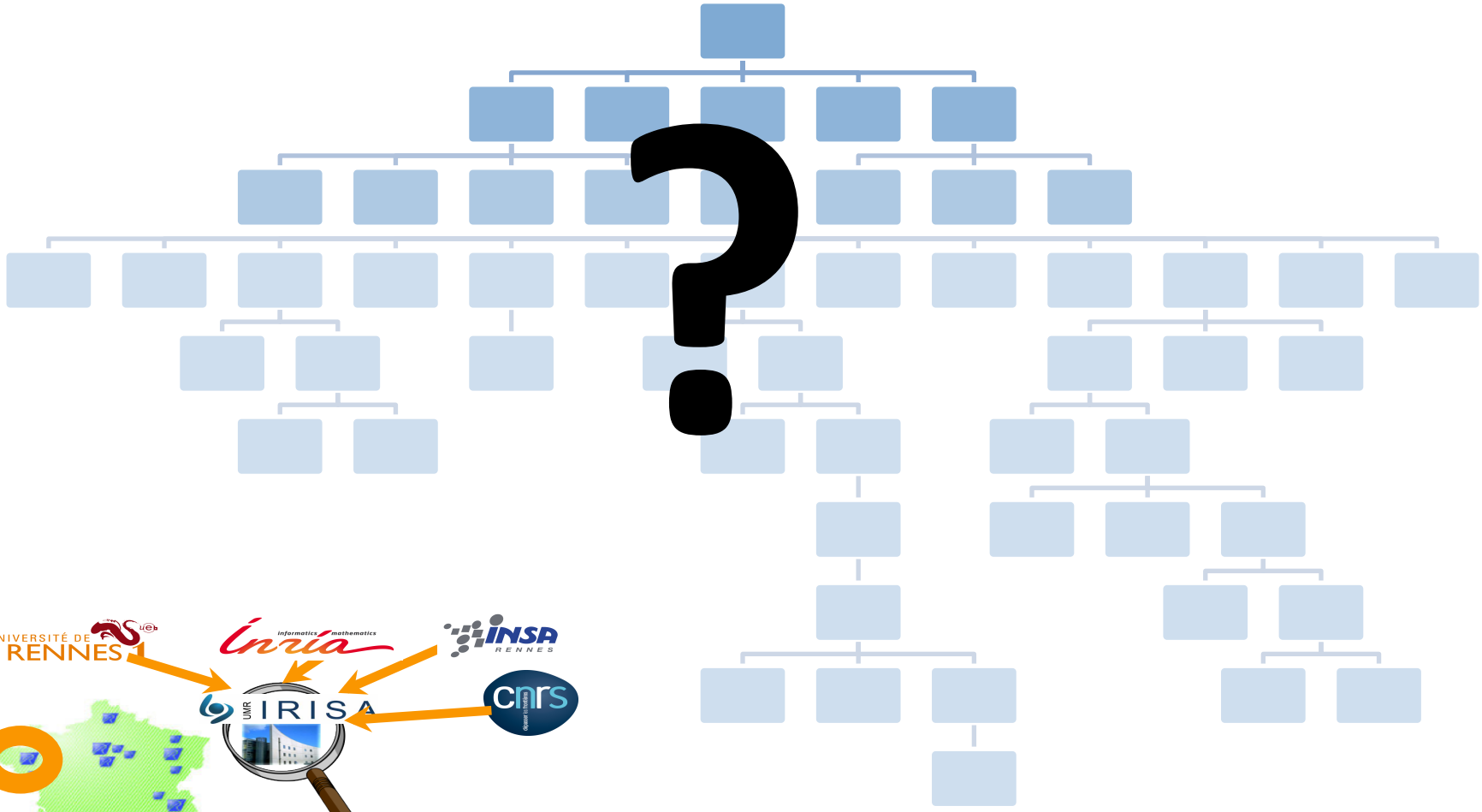
Automated Analysis

Languages, API/DSLs

Evaluation **(European projects, long-term collaboration with Thales, open source systems)**

FAMiliAR

FAMiliAR


?

UNIVERSITÉ DE RENNES 1
informatics mathematics
INSA RENNES
UMR IRISA
cnrs