

Modeling Variability (part 2)

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

Material

<http://mathieuacher.com/teaching/MDE/MRI1516/>

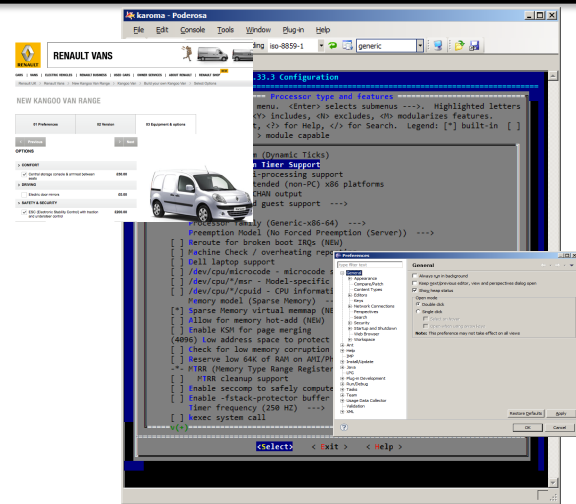
Plan

- Challenges and Overview
 - Developing billions of software product is hard but now a common practice
- Implementing Variability
 - Revisit of existing techniques and curriculum
- Specificity of Product Line Engineering
 - Process, methods
- **Feature Models**
 - **Defacto standard for modeling product lines and variability**
 - **Syntax, semantics, automated reasoning, synthesis**

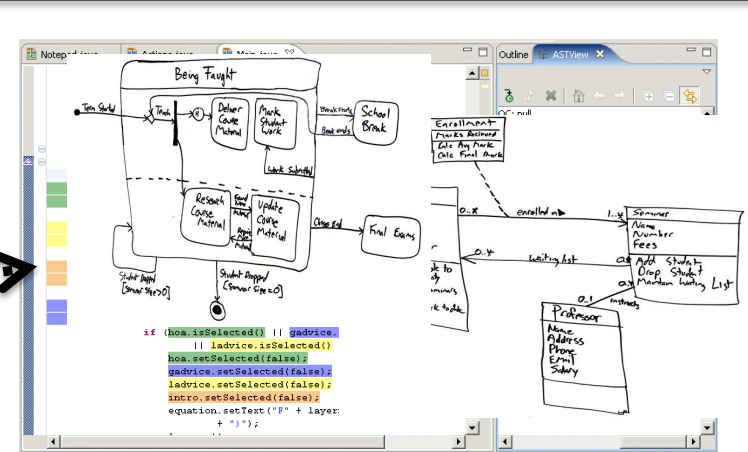
Contract

- The idea of software product lines and variability
 - You will be able to recognize this class of systems
 - Aware of the complexity, the specific development process, and existing techniques
- **Feature modeling**
 - **A widely used formalism for modeling product lines and configurable systems in a broad sense**
- **Composing/Decomposing feature models with a domain-specific language**
- **Reverse engineering variability models**

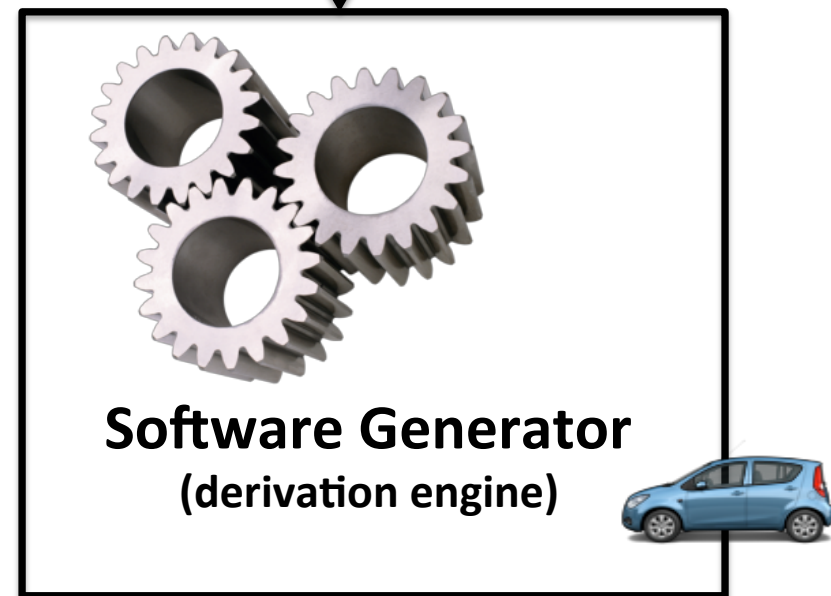
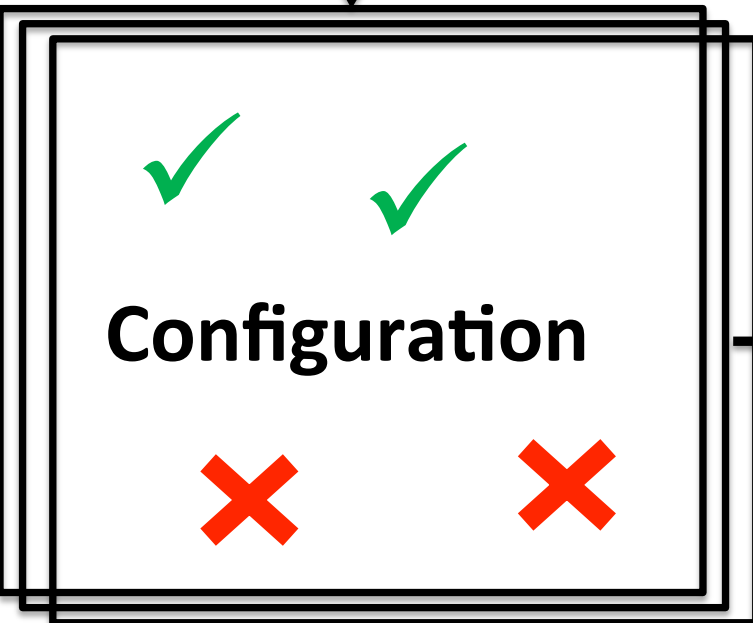
Previously



Variability Model



Base Artefacts (e.g., models)



```
macher-wifi:getting-started macher1$ yo jhipster  
  
I'm all done. Running npm install & bower install for you to install the required dependencies.
```

CHAPTER STACK FOR JAWA IDEWS

Welcome to the JHipster Generator v2.17.0

```
? (1/15) What is the base name of your application? jhipster  
? (2/15) What is your default Java package name? com.mycompany.myapp  
? (3/15) Do you want to use Java 8? Yes (use Java 8)  
? (4/15) Which *type* of authentication would you like to use? (Use arrow keys)  
? HTTP Session Authentication (stateful, default Spring Security mechanism)  
? OAuth2 Authentication (stateless, with an OAuth2 server implementation)  
? Token-based authentication (stateless, with a token)
```

Variability Model



mapping

```
Branch: master  
generator-jhipster / app / templates / src / main / java / package / config / _DatabaseConfiguration.java  
jdubois 2 days ago Use Spring Boot's configuration meta-data  
9 contributors  
184 lines (165 sloc) | 9.69 KB  
Raw Blame History  
1 package <packageName>.config;  
2 < if (databaseType == 'sql') { %>  
3 import <packageName>.config.liquibase.AsyncSpringLiquibase;  
4 import com.codahale.metrics.MetricRegistry;  
5 import com.fasterxml.jackson.datatype.hibernate4.HibernateModule;  
6 import com.zaxxer.hikari.HikariConfig;  
7 import com.zaxxer.hikari.HikariDataSource;  
8 import liquibase.integration.spring.SpringLiquibase;< %>< if (databaseType == 'mongodb' && authenticationType == 'oauth2') { %>  
9 import <packageName>.config.oauth2.OAuth2AuthenticationMeuConverter;< %>< if (databaseType == 'mongodb') { %>  
10 import org.mongeez.Mongeez;< %>  
11 import org.mongeez.< %>  
12 import org.slf4j.Logger;  
13 import org.slf4j.LoggerFactory;< %> if (databaseType == 'sql') { %>< if (hibernateCache == 'hazelcast') { %>  
14 import org.springframework.cache.CacheManager;< %>  
15 import org.springframework.beans.factory.annotation.Autowired;  
16 import org.springframework.boot.autoconfigure.condition.ConditionalOnExpression;< %>< if (databaseType == 'mongodb') { %>  
17 import org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguration;  
18 import org.springframework.boot.autoconfigure.mongo.MongoProperties;< %>< if (databaseType == 'sql') { %>  
19 import org.springframework.boot.autoconfigure.jdbc.DataSourceProperties;  
20 import org.springframework.boot.autoconfigure.liquibase.LiquibaseProperties;  
21 import org.springframework.context.ApplicationContextException;< %>  
22 import org.springframework.context.annotation.Bean;  
23 import org.springframework.context.annotation.Configuration;  
24 import org.springframework.context.annotation.Profile;< %>< if (databaseType == 'mongodb') { %>  
25 import org.springframework.context.annotation.Import;< %>< if (databaseType == 'sql') { %>  
26 import org.springframework.core.env.Environment;< %>< if (databaseType == 'mongodb' && authenticationType == 'oauth2') { %>  
27 import org.springframework.core.convert.converter.Converter;< %>< if (databaseType == 'mongodb') { %>  
28 import org.springframework.core.io.ClassPathResource;< %>< if (searchEngine == 'elasticsearch') { %>  
29 import org.springframework.data.elasticsearch.repository.config.EnableElasticsearchRepositories;< %>< if (databaseType == 'mon  
30 import org.springframework.data.mongodb.config.AbstractMongoConfiguration;  
31 import org.springframework.data.mongodb.config.EnableMongoAuditing;< %>< if (databaseType == 'mongodb' && authenticationType =  
32 import org.springframework.data.mongodb.core.convert.CustomConversions;< %>< if (databaseType == 'mongodb') { %>  
33 import org.springframework.data.mongodb.core.mapping.event.ValidatingMongoEventListener;  
34 import org.springframework.data.mongodb.repository.config.EnableMongoRepositories;  
35 import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;< %>< if (databaseType == 'sql') { %>
```

Base Artefacts

Software Generator
(derivation engine)



Illegal variant

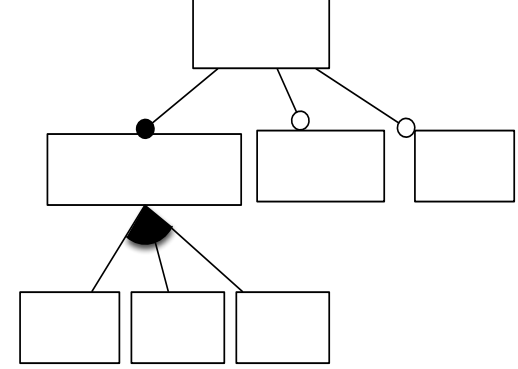
Unused flexibility



Feature Models

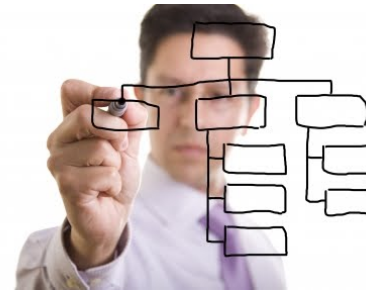
Syntacs, Semantics
Feature diagram, Formula

Feature Model

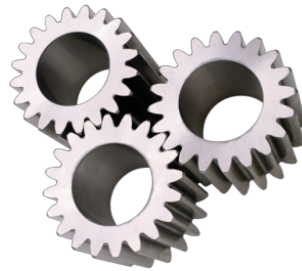


not, and, or, implies

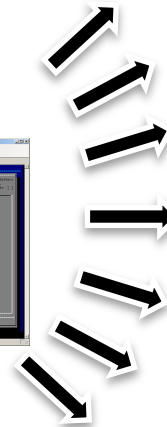
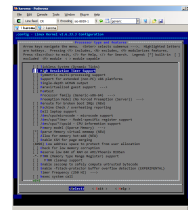
Communicative

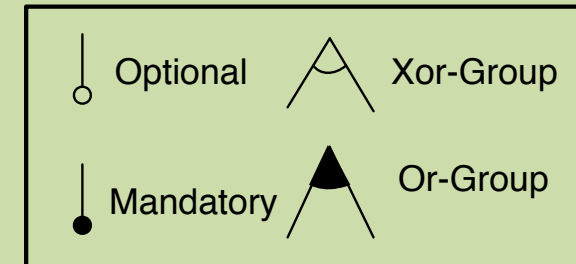
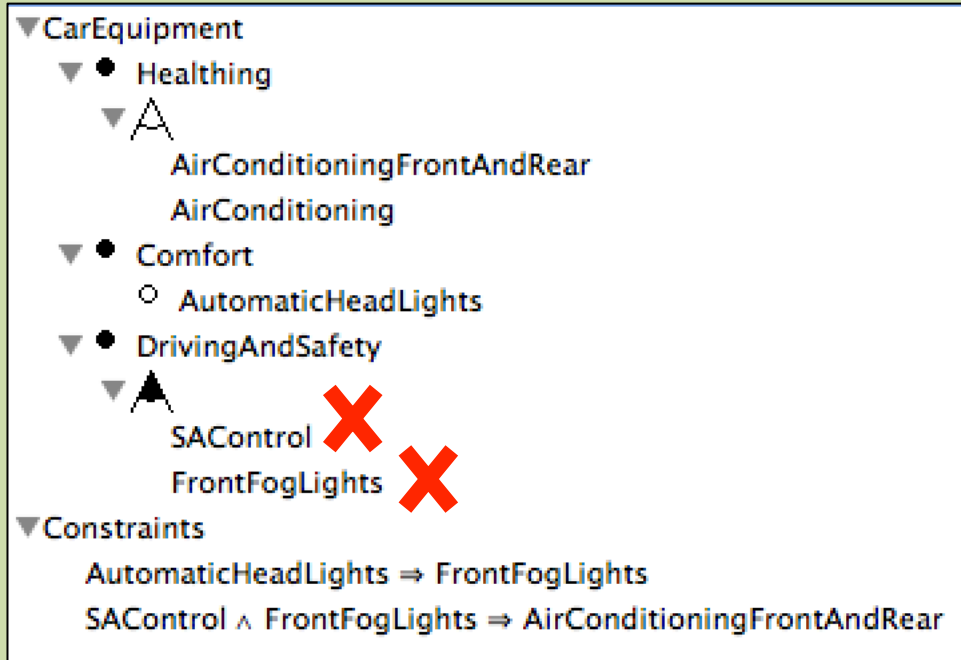


Analytic



Generative



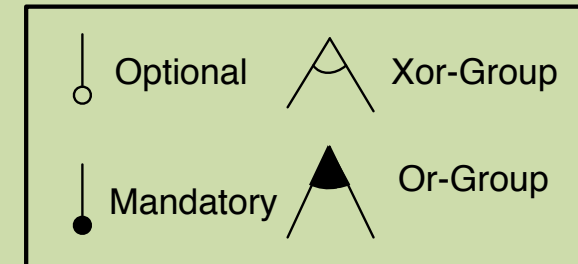
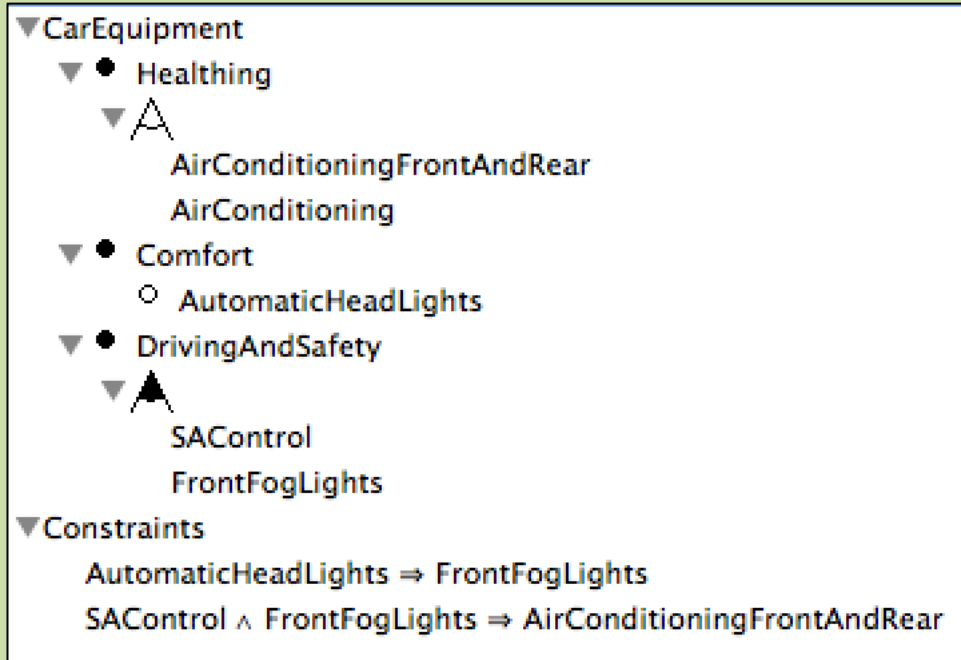


Hierarchy + Variability
 =
set of valid configurations



Or-group: at least one!





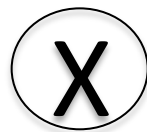
Hierarchy + Variability

=

set of valid configurations



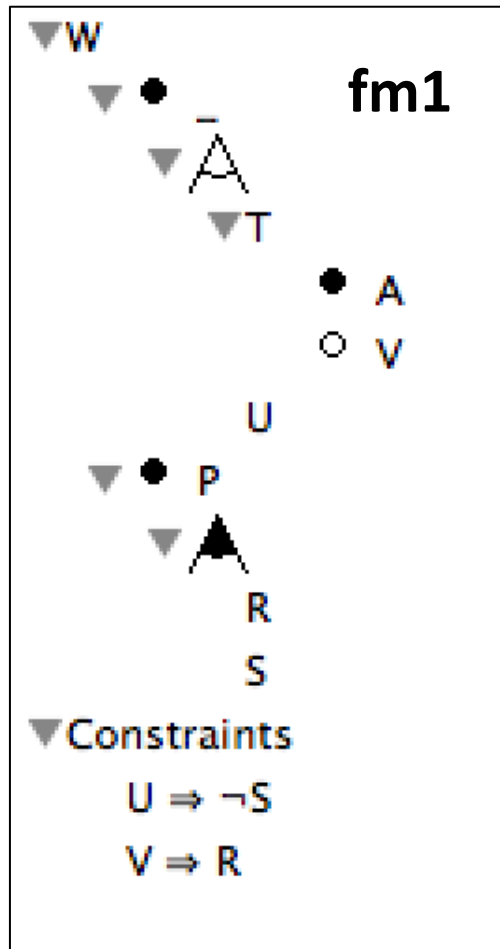
{CarEquipment, Comfort,
DrivingAndSafety,
Healthing}



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

(Boolean) Feature Models

Hierarchy + Variability = set of valid configurations



$$\llbracket fm1 \rrbracket = \{$$

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

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

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

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

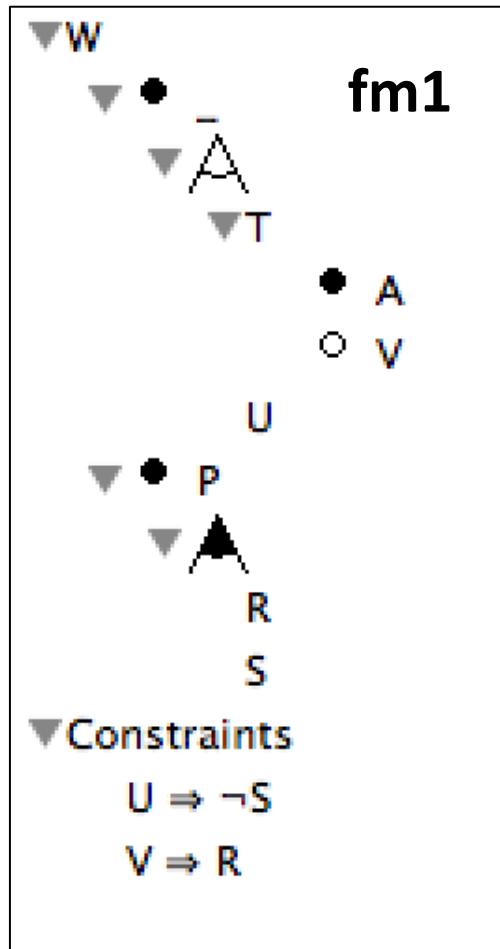
$$\{W, P, R, T, V, A\},$$

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

$$\}$$

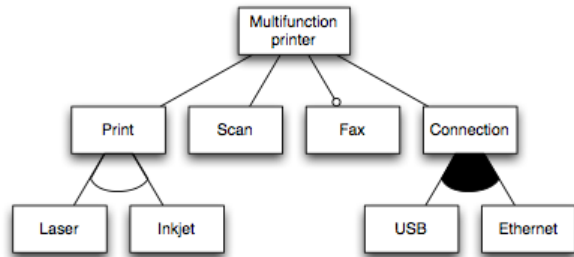
(Boolean) Feature Models

~ Boolean formula



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

Typical implementations



result



logics



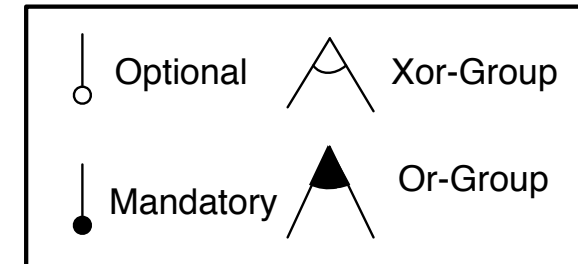
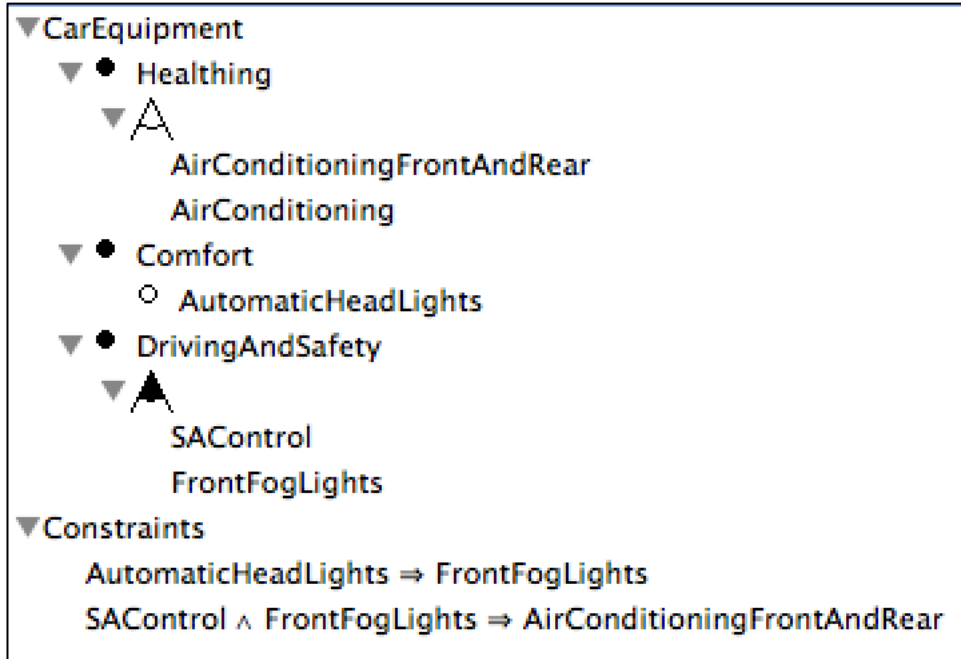
solvers



Z3

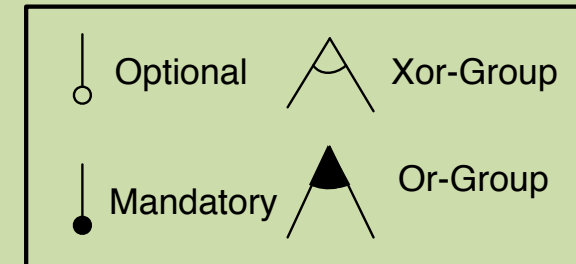
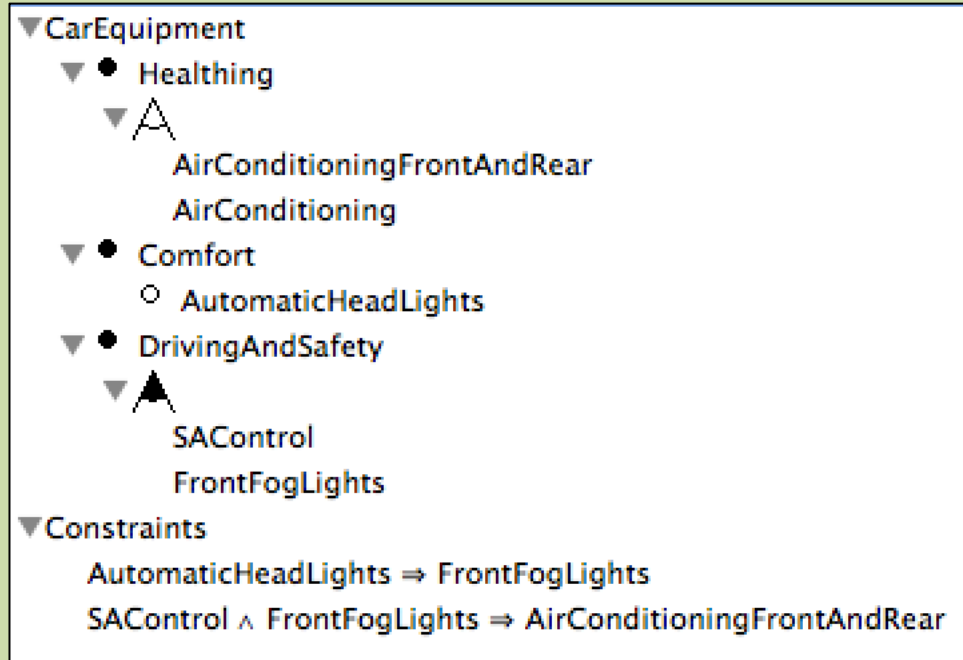
Formal semantics of a language

- **formal syntax** (L) – clearcut syntactic rules defining all legal diagrams, a.k.a. syntactic domain
- **semantic domain** (S) – a mathematical abstraction of the real-world concepts to be modelled
- **semantic function** ($M: L \rightarrow S$) – clearcut semantic rules defining the meaning of all legal diagrams



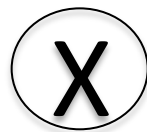
Definition 2 (Feature Diagram) A feature diagram $FD = \langle G, E_{MAND}, G_{XOR}, G_{OR}, I, EX \rangle$ is defined as follows: $G = (\mathcal{F}, E, r)$ is a rooted, labeled tree where $\mathcal{F}$ is a finite set of features, $E \subseteq \mathcal{F} \times \mathcal{F}$ is a finite set of edges and $r \in \mathcal{F}$ is the root feature ; $E_{MAND} \subseteq E$ is a set of edges that define mandatory features with their parents ; $G_{XOR} \subseteq \mathcal{P}(\mathcal{F}) \times \mathcal{F}$ and $G_{OR} \subseteq \mathcal{P}(\mathcal{F}) \times \mathcal{F}$ define feature groups and are sets of pairs of child features together with their common parent feature ; I a set of implies constraints whose form is $A \Rightarrow B$, EX is a set of excludes constraints whose form is $A \Rightarrow \neg B$ ($A \in \mathcal{F}$ and $B \in \mathcal{F}$).

Definition 3 (Feature Model) An FM is a tuple $\langle FD, \psi \rangle$ where FD is a feature diagram and ψ is a propositional formula over the set of features $\mathcal{F}$.



Definition 1 (Configuration Semantics) *A configuration of an FM fm is defined as a set of selected features. $\llbracket fm_1 \rrbracket$ denotes the set of valid configurations of fm_1 and is a set of sets of features.*

{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}

Quizz

1) Give two feature models with the same configuration semantics but with different syntax

2) Does it matter ?

$A \wedge$
 $A \Leftrightarrow B \wedge$
 $C \Rightarrow A \wedge$
 $D \Rightarrow A$

Feature Model Synthesis Problem

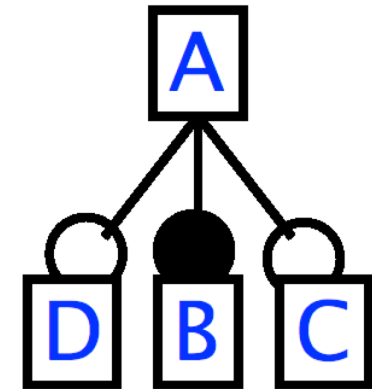
[Czarnecki et al., SPLC'07]

[She et al., ICSE'11]

[Andersen et al., SPLC'12]

Φ

FAMiLiAR



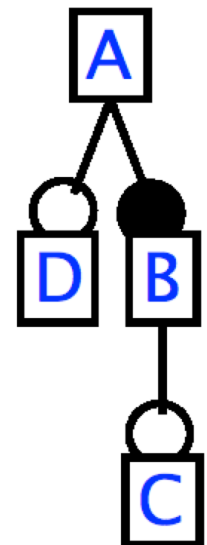
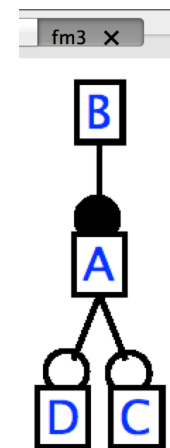
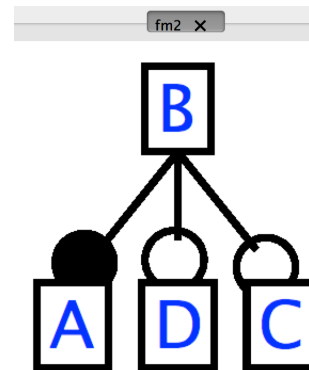
FM

```

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)
  
```



#1 Reverse Engineering Scenarios

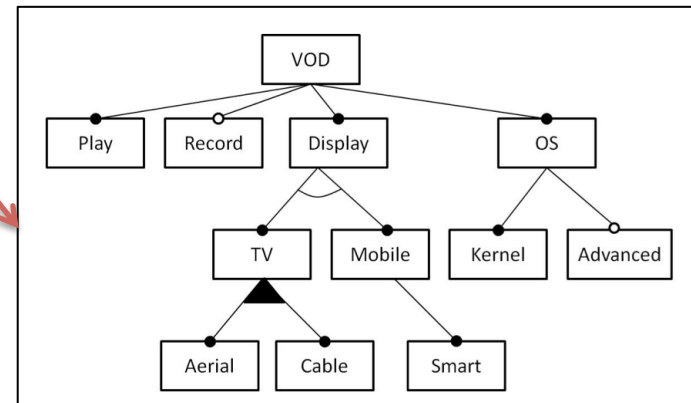
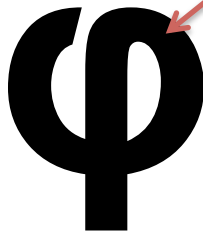
- [Haslinger et al., WCRE'11], [Acher et al., VaMoS'12]

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 union fm_*

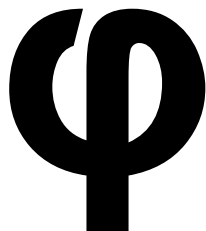
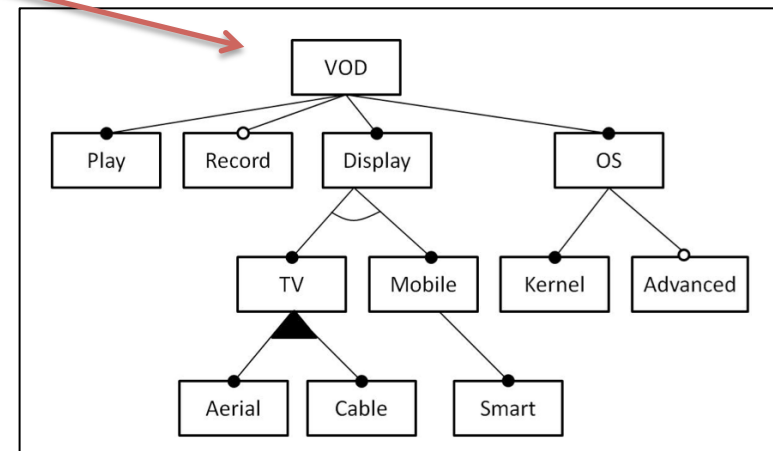
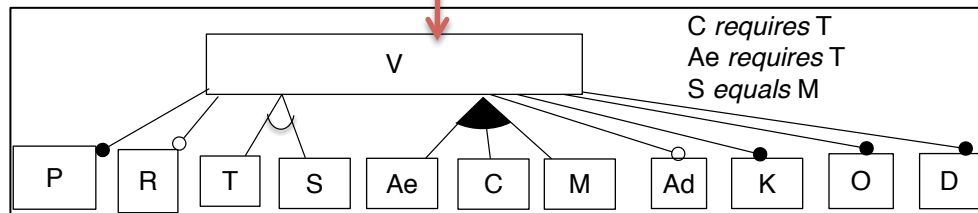
fmR2 = ksynthesis fmR with hierarchy= VOD : V P R D O ; O : K Ad ; D : T M ; T : Ae C ; M : S ;
```



#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 ;
```



```
fm1> compare fmR fmR2  
res0: (STRING) REFACTORING
```

Feature Model SemanticS

- As configuration semantics is not sufficient...
- **Ontological** semantics
 - Hierarchy
 - And feature groups

Quiz (Class Diagram)

- 1) Give two class diagrams with the same semantics but with different syntax
 - (preliminary) what is the semantic domain of CDs?
- 2) Does it matter ?

3.1 Class diagrams language

As a concrete CD language we use the class diagrams of UML/P [29], a conceptually refined and simplified variant of UML designed for low-level design and implementation. Our semantics of CDs is based on [11] and is given in terms of sets of objects and relationships between these objects. More formally, the semantics is defined using three parts: a precise definition of the syntactic domain, i.e., the syntax of the modeling language CD and its context conditions (we use MontiCore [16,24] for this); a semantic domain - for us, a subset of the System Model (see [7,8]) OM, consisting of all finite object models; and a mapping $sem : CD \rightarrow \mathcal{P}(OM)$, which relates each syntactically well-formed CD to a set of constructs in the semantic domain OM. For a thorough and formal account of the semantics see [8].

Note that we use a *complete* interpretation for CDs (see [29] ch. 3.4). This roughly means that ‘whatever is not in the CD, should indeed not be present in the object model’. In particular, we assume that the list of attributes of each class is complete, e.g., an **employee** object with an **id** and a **salary** is not considered as part of the semantics of an **Employee** class with an **id** only.

The CD language constructs we support include generalization (inheritance), interface implementation, abstract and singleton classes, class attributes, uni- and bi-directional associations with multiplicities, enumerations, aggregation, and composition.

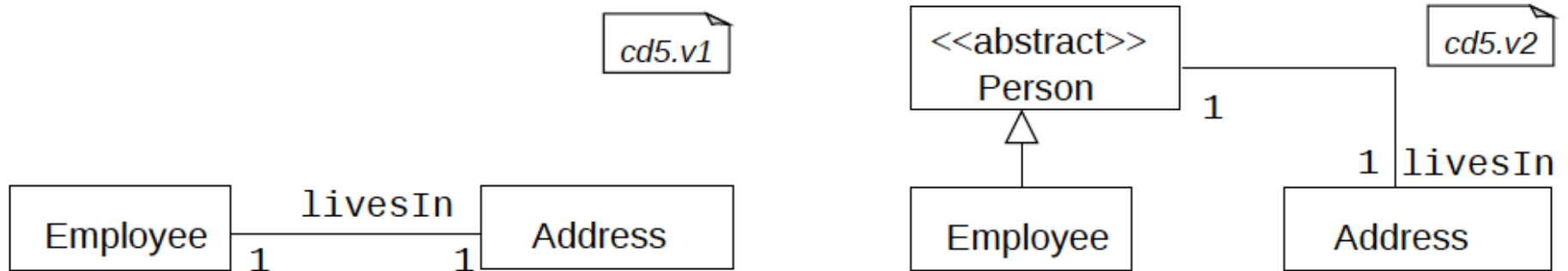


Fig. 4. *cd5.v1* and its revised version *cd5.v2*. The two versions have equal semantics.

Quizz (back to Feature Model)

Definition 2 (Feature Diagram) A feature diagram $FD = \langle G, E_{MAND}, G_{XOR}, G_{OR}, I, EX \rangle$ is defined as follows: $G = (\mathcal{F}, E, r)$ is a rooted, labeled tree where $\mathcal{F}$ is a finite set of features, $E \subseteq \mathcal{F} \times \mathcal{F}$ is a finite set of edges and $r \in \mathcal{F}$ is the root feature ; $E_{MAND} \subseteq E$ is a set of edges that define mandatory features with their parents ; $G_{XOR} \subseteq \mathcal{P}(\mathcal{F}) \times \mathcal{F}$ and $G_{OR} \subseteq \mathcal{P}(\mathcal{F}) \times \mathcal{F}$ define feature groups and are sets of pairs of child features together with their common parent feature ; I a set of implies constraints whose form is $A \Rightarrow B$, EX is a set of excludes constraints whose form is $A \Rightarrow \neg B$ ($A \in \mathcal{F}$ and $B \in \mathcal{F}$).

Definition 3 (Feature Model) An FM is a tuple $\langle FD, \psi \rangle$ where FD is a feature diagram and ψ is a propositional formula over the set of features $\mathcal{F}$.

Given a set of configurations s , can we always characterize s with a feature diagram fd ?

ie $[[fd]] = s$

In other words: is the formalism of feature diagram expressive enough wrt Boolean logic?

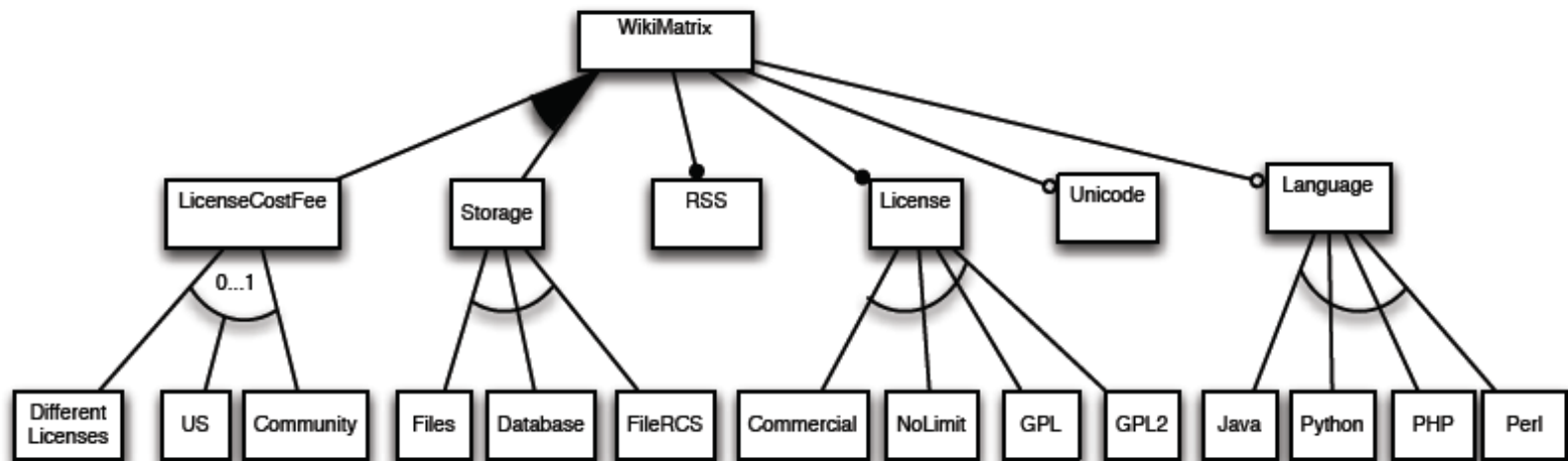
Feature Diagram ?

```
s = {{A},  
{A,C,B},  
{B,A},  
{C,D,A},  
{D,A},  
{A,D,B},  
{A,C}  
}
```

```
fm1 = FM (A : [B] [C] [D] ^  
// B, C and D are optional features of A  
  
((B & C) -> !D)  
)
```

Feature Diagram ?

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



BI =
 Storage <-> Unicode
 Community <-> FileRCS
 Commercial <-> US10
 FileRCS <-> Perl
 Unicode <-> Language
 US10 <-> Java

I = GPL2 -> PHP
 GPL -> Storage

Ψ_{cst}

E =
 DifferentLicenses -> ~GPL
 Database -> ~Python
 NoLimit -> ~DifferentLicenses
 Unicode -> ~NoLimit
 LicenseCostFee -> ~Files



Quizz

Feature Model (bis)

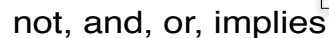
$s = \{\{A\}, \{B\}\}$

$fd = ?$

Feature Model: Key Insights

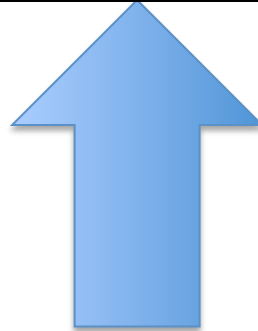
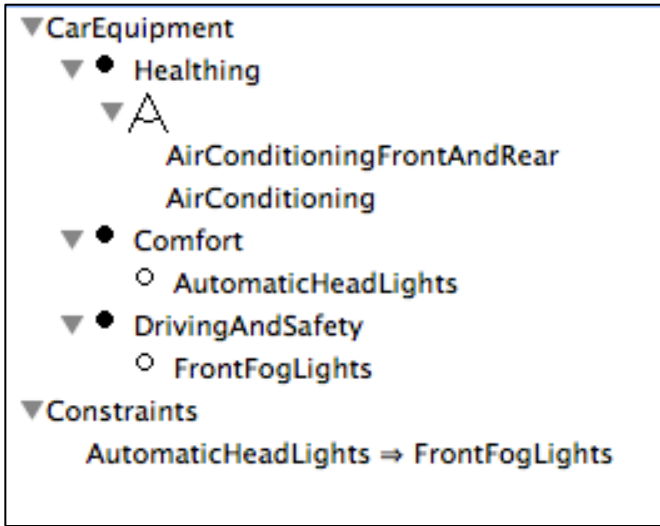
- Semantics
 - Configuration and ontological
- Syntax
 - Feature diagram vs Feature Model
 - Feature diagram not expressively complete
- Feature models are a (syntactical) view of a propositional formula

Synthesising Feature Models (from semantics to syntacs)

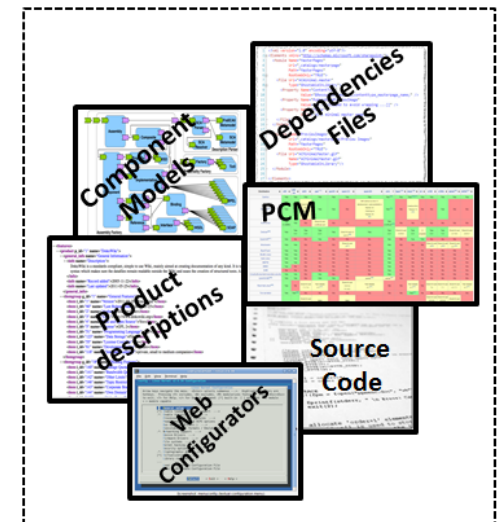


Variability Models (feature models)

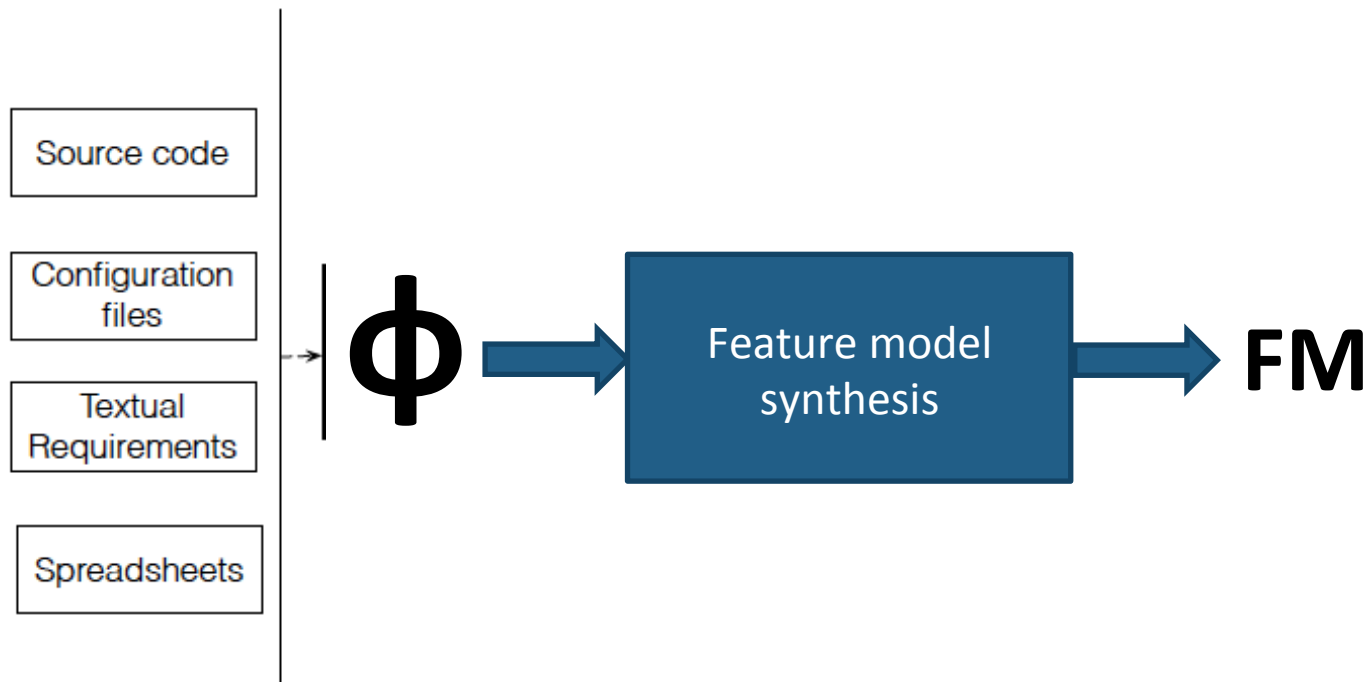




Φ



Product ▲ ▼	CarEquipment ▼	Comfort ▼	DrivingAndSafety ▼	Healing ▼	AirConditioning ▼
Find	Yes ☐ No ☐	Yes ☐ No ☐	Yes ☐ No ☐	Yes ☐ No ☐	Yes ☐ No ☐
Car1	yes	yes	yes	yes	yes
Car2	yes	yes	yes	yes	yes
Car3	yes	yes	yes	yes	no
Car4	yes	yes	yes	yes	no
Car5	yes	yes	yes	yes	yes
Car6	yes	yes	yes	yes	no



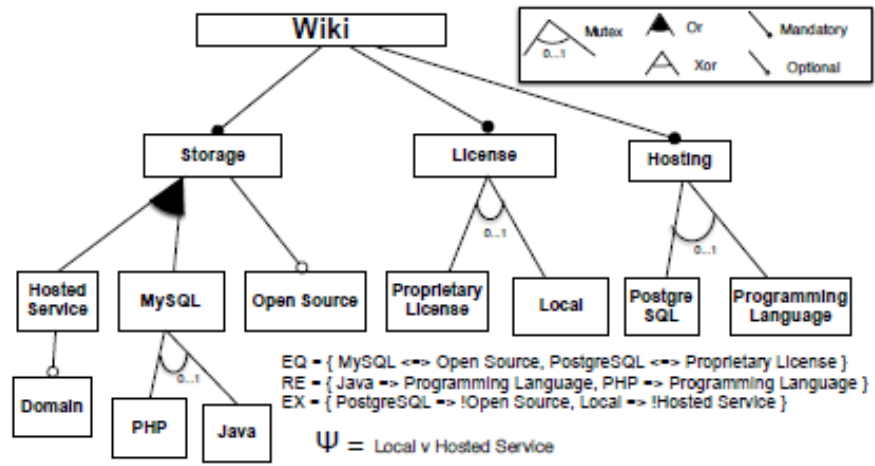
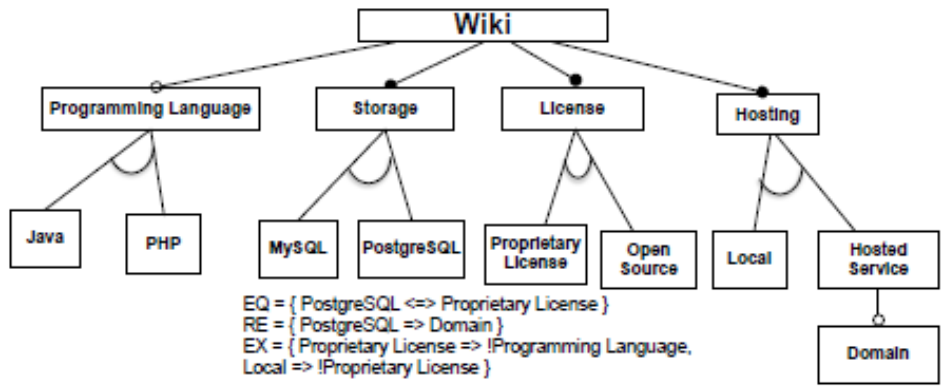
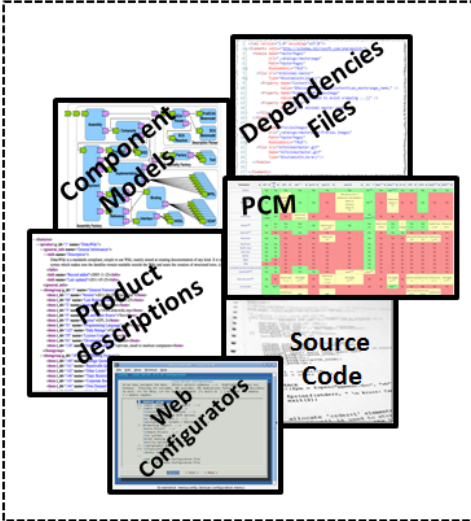
Feature model synthesis problem

Input: Φ , a propositional formula representing the **dependencies** over a set of features F .

Output: a maximal feature model with a **sound configuration semantics**

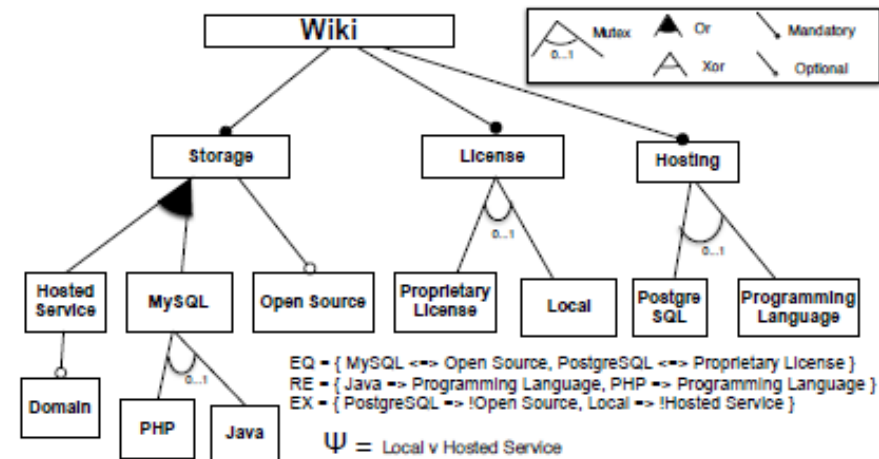
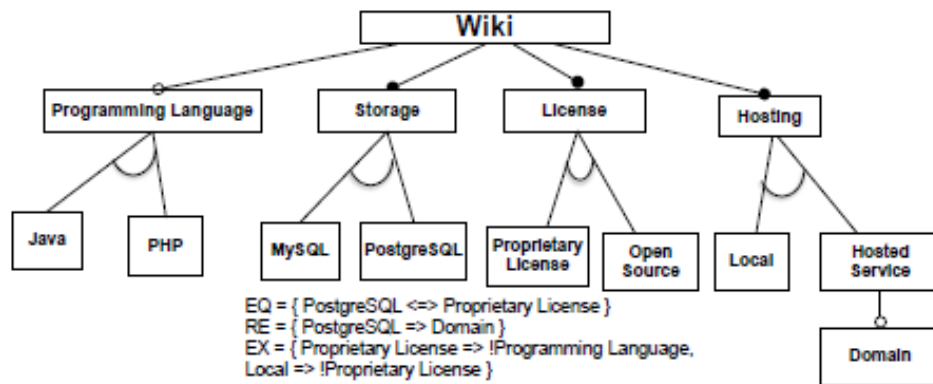
For a given configuration set,

many (maximal) feature diagrams



Maximal feature diagrams

~ “as much logical information as possible is represented in the diagram itself, without resorting to the constraints”

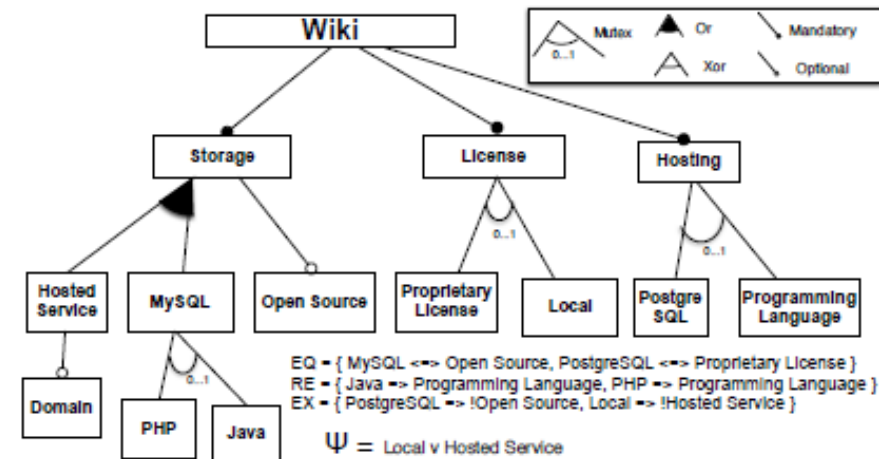
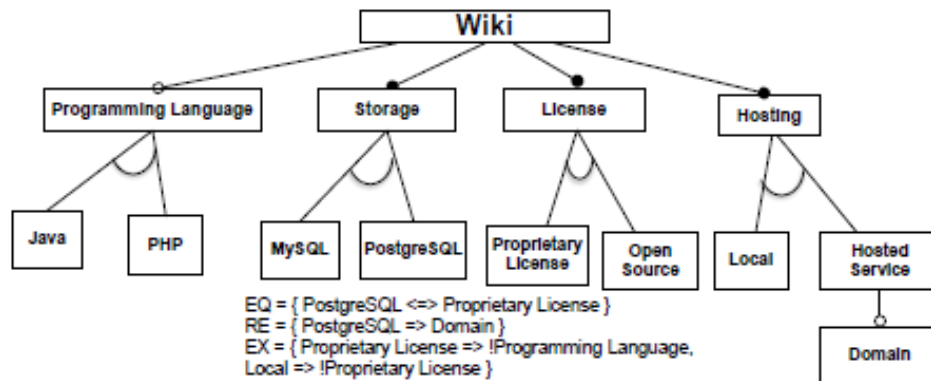


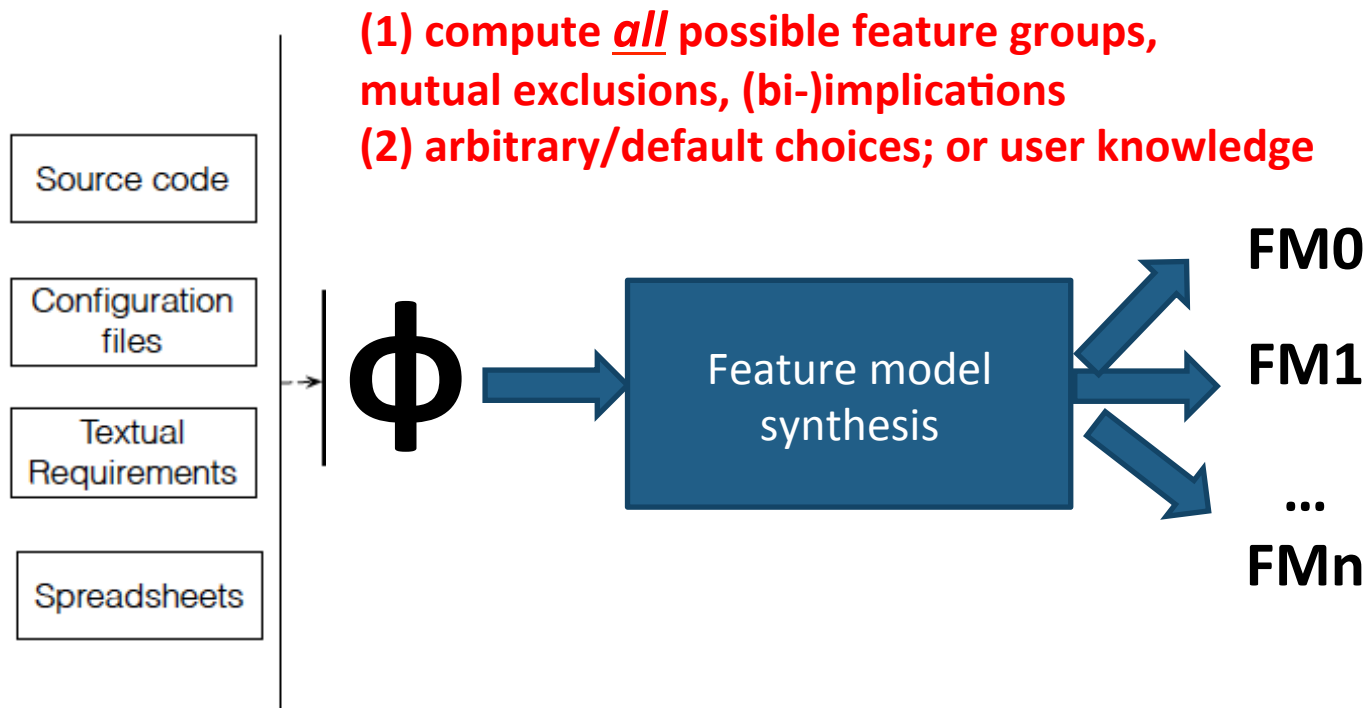
Maximal feature diagrams

~ “as much logical information as possible is represented in the diagram itself, without resorting to the constraints”

Quizz:

give a non maximal feature diagram





Feature model synthesis problem

Input: Φ , a propositional formula representing the **dependencies** over a set of features F .

Output: a maximal feature model with a **sound configuration semantics**

(φ : formula over F rooted in r , $r \in F$)

▷ Find and remove all dead features

$$1 \quad D = \{f \in F \mid \varphi \wedge r \rightarrow \neg f\}$$

$$2 \quad \varphi = \varphi[d \mapsto 0]_{d \in D}$$

▷ Compute the implication graph $G(V, E)$

$$3 \quad V = F \setminus D$$

$$4 \quad E = \{(u, v) \in V \times V \mid \varphi \wedge u \rightarrow v\}$$

▷ Compute strongly connected components

$$5 \quad V' = \{S \subseteq V \mid S \text{ is a SCC of } G\}$$

▷ Make edges between SCCs creating a DAG

$$6 \quad E' = \{(u, v) \in V' \times V' \mid u \neq v \text{ and}$$

$$7 \quad \exists u' \in u, v' \in v. (u', v') \in E\}$$

▷ Compute the mutex graph $M(V, E_x)$

$$8 \quad E_x = \{\{u, v\} \subseteq V' \mid \exists u' \in u, v' \in v. \varphi \wedge u' \rightarrow \neg v'\}$$

▷ Compute mutex-groups

$$9 \quad G_m = \{\{(f_1, p), \dots, (f_k, p)\} \mid \{f_1, \dots, f_k\} \text{ is}$$

$$10 \quad \text{a maximal clique in } M \text{ and } \forall f_i. (f_i, p) \in E'\}$$

▷ Compute or-groups

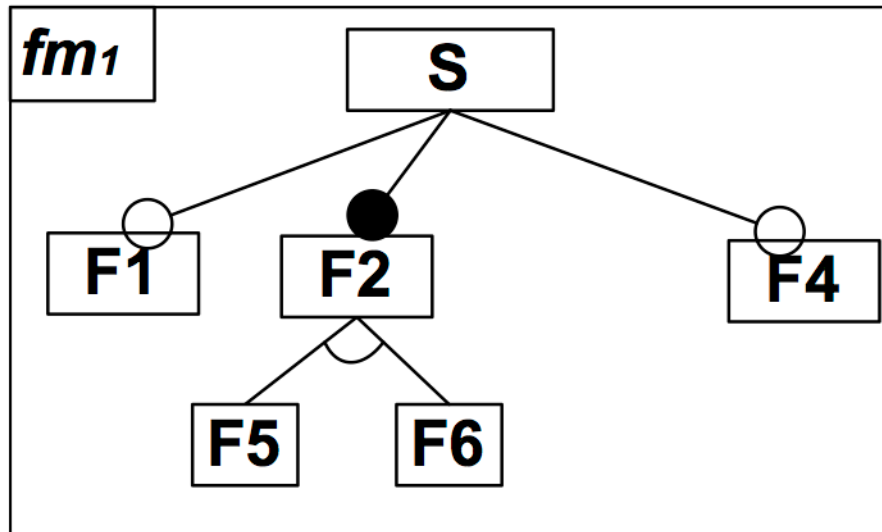
$$11 \quad G_o = \{\{(f_1, p), \dots, (f_k, p)\} \mid f'_1 \vee \dots \vee f'_k \text{ is}$$

$$12 \quad \text{a prime implicate of } \varphi \wedge p' \text{ and}$$

$$13 \quad p' \in p \text{ and } \forall f_i. f'_i \in f_i \wedge (f_i, p) \in E'\}$$

▷ Compute xor-groups

$$14 \quad G_x = \{\{(f_1, p), \dots, (f_k, p)\} \in G_o \mid \forall i \neq j. (f_i, f_j) \in E_x\}$$

$$\begin{aligned}
 & (F5 \rightarrow F2) \wedge (F2 \rightarrow S) \wedge \text{SYNTETIC_ROOT_FEATURE} \wedge \\
 & (F6 \rightarrow \neg F5) \wedge (F1 \rightarrow S) \wedge (\text{SYNTETIC_ROOT_FEATURE} \\
 & \rightarrow S) \wedge (F4 \rightarrow S) \wedge (S \rightarrow F2) \wedge (F6 \rightarrow F2) \wedge ((\neg F2 \mid F6) \mid \\
 & (\neg F2 \mid F5)) \wedge (S \rightarrow \text{SYNTETIC_ROOT_FEATURE})
 \end{aligned}$$


```
fm1> computeImplies fm1
```

```
computeImplies fm1 ==> {(F1 -> F2);(F5 -> S);(S -> F2);(F6 -> F2);(F4 -> S);(F1 -> S);(F5 -> F2);(F4 -> F2);(F2 -> S);(F6 -> S)}
```

```
fm1> cliques fm1
```

```
cliques fm1 ==> {{S;F2}}
```

```
fm1> computeXORGroups fm1
```

```
computeXORGroups fm1 ==> {[F6, F5] -> S (XOR);[F6, F5] -> F2 (XOR)}
```

```
fm1> computeMUTEXGroups fm1
```

```
computeMUTEXGroups fm1 ==> {}
```

```
fm1> computeExcludes fm1
```

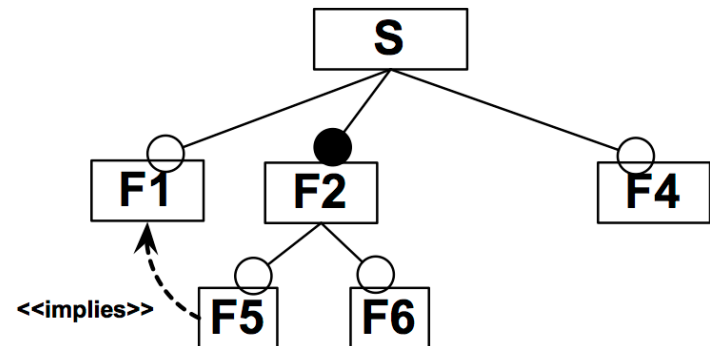
```
computeExcludes fm1 ==> {(F5 -> !F6)}
```

Quizz

There and Back Again

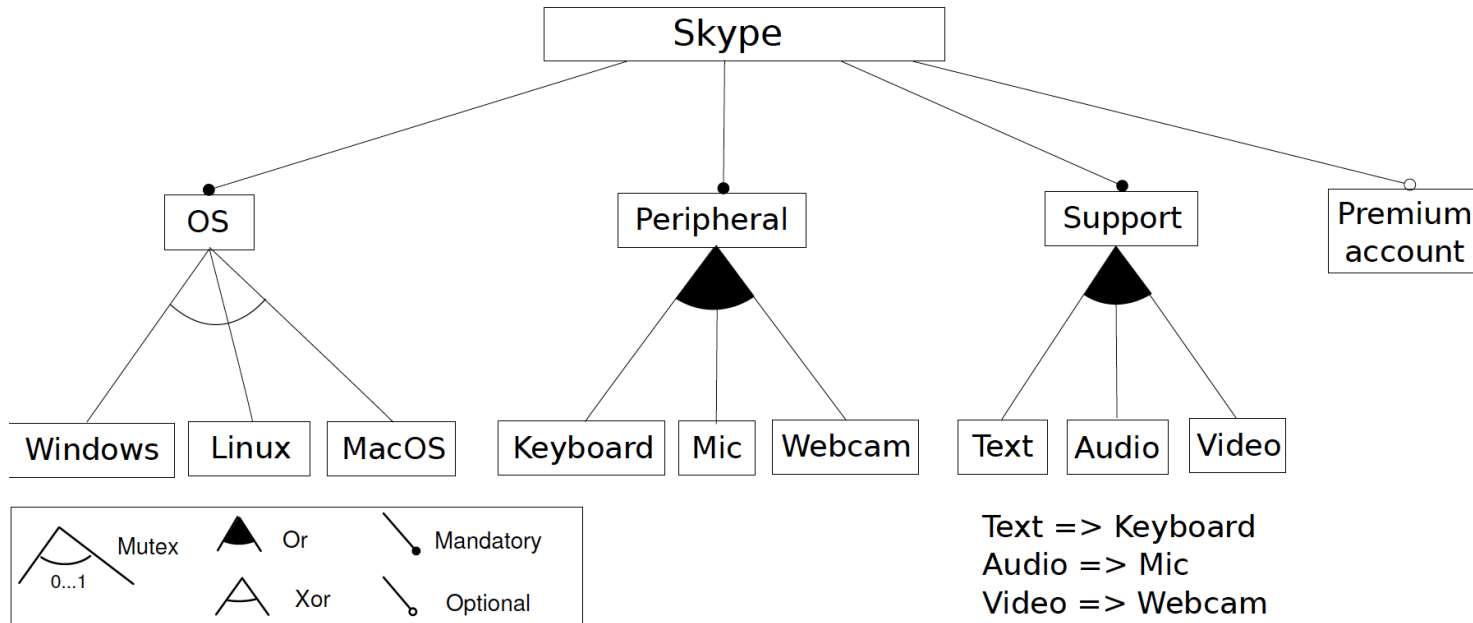
1) Translate this feature model to a propositional formula

2) Compute the synthesis « structures »

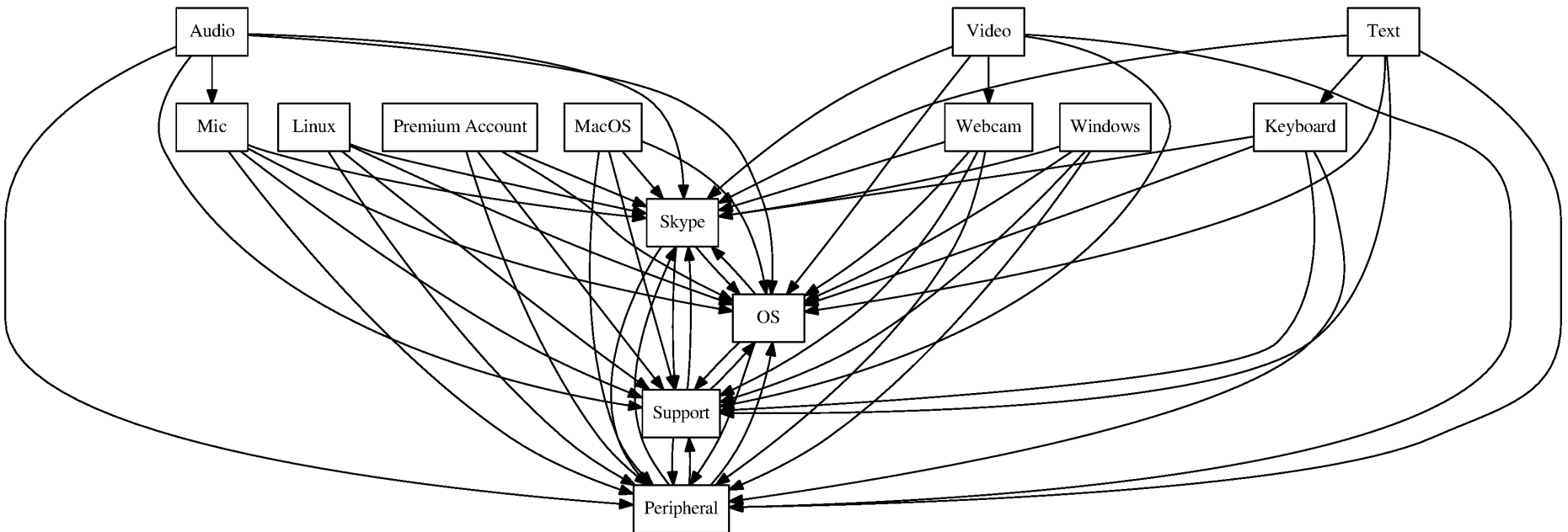


Key: Binary Implication Graph

Sound and complete representation of possible hierarchies



Sound and complete representation of possible hierarchies

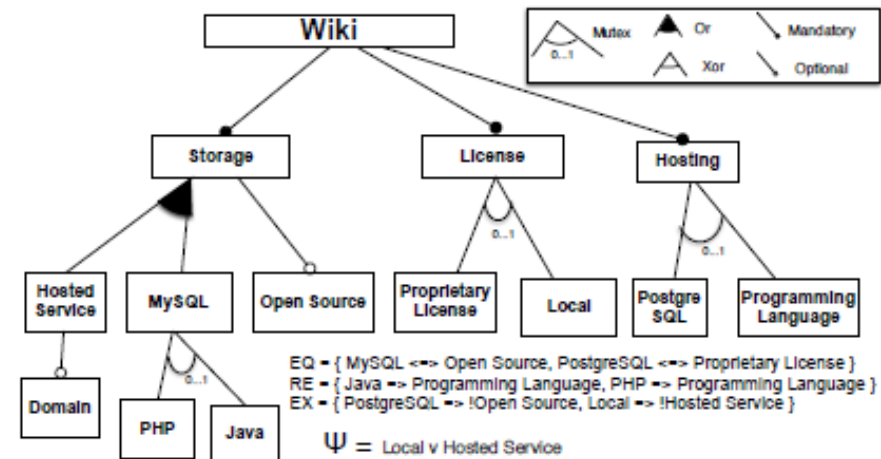
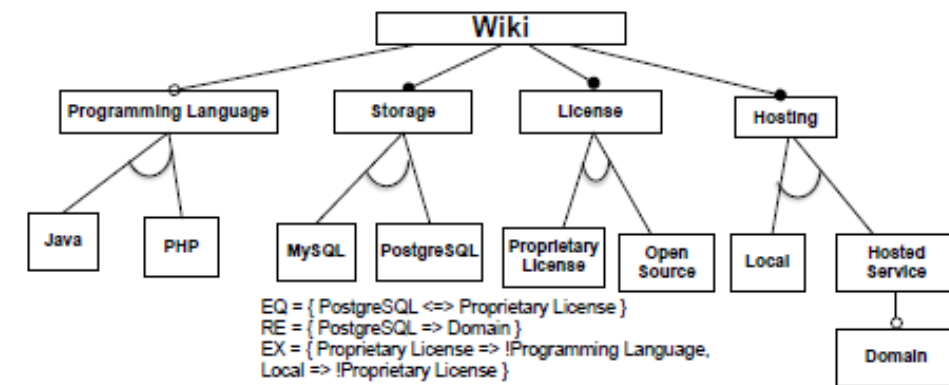
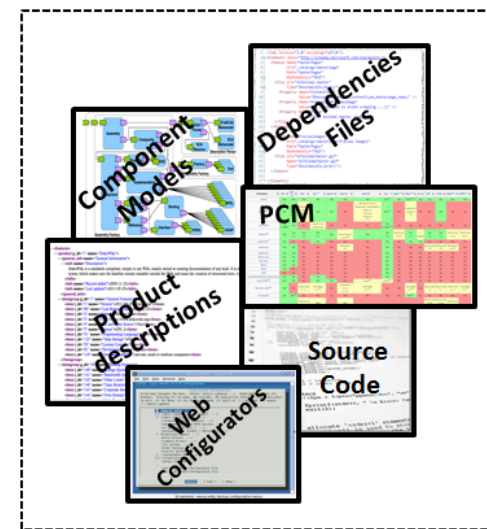


For a given configuration set,

many (maximal) feature diagrams

with different ontological semantics

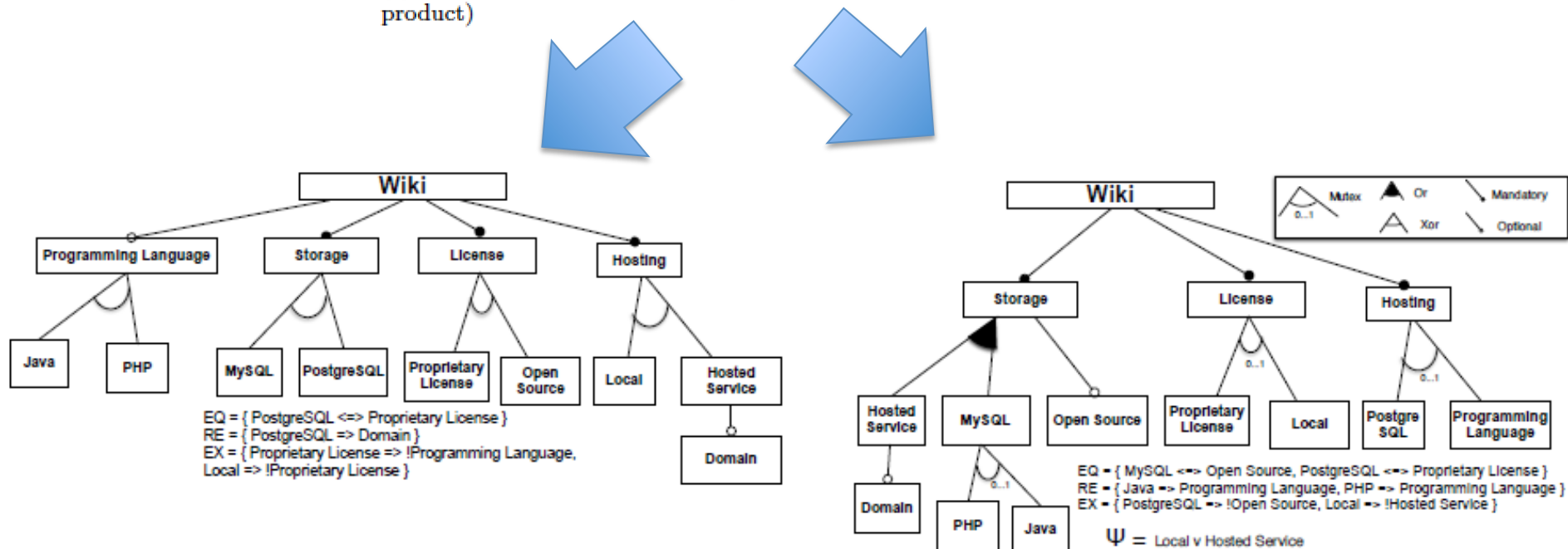
[She et al. ICSE'11, Andersen et al. SPLC'12, Acher et al. VaMoS'13]



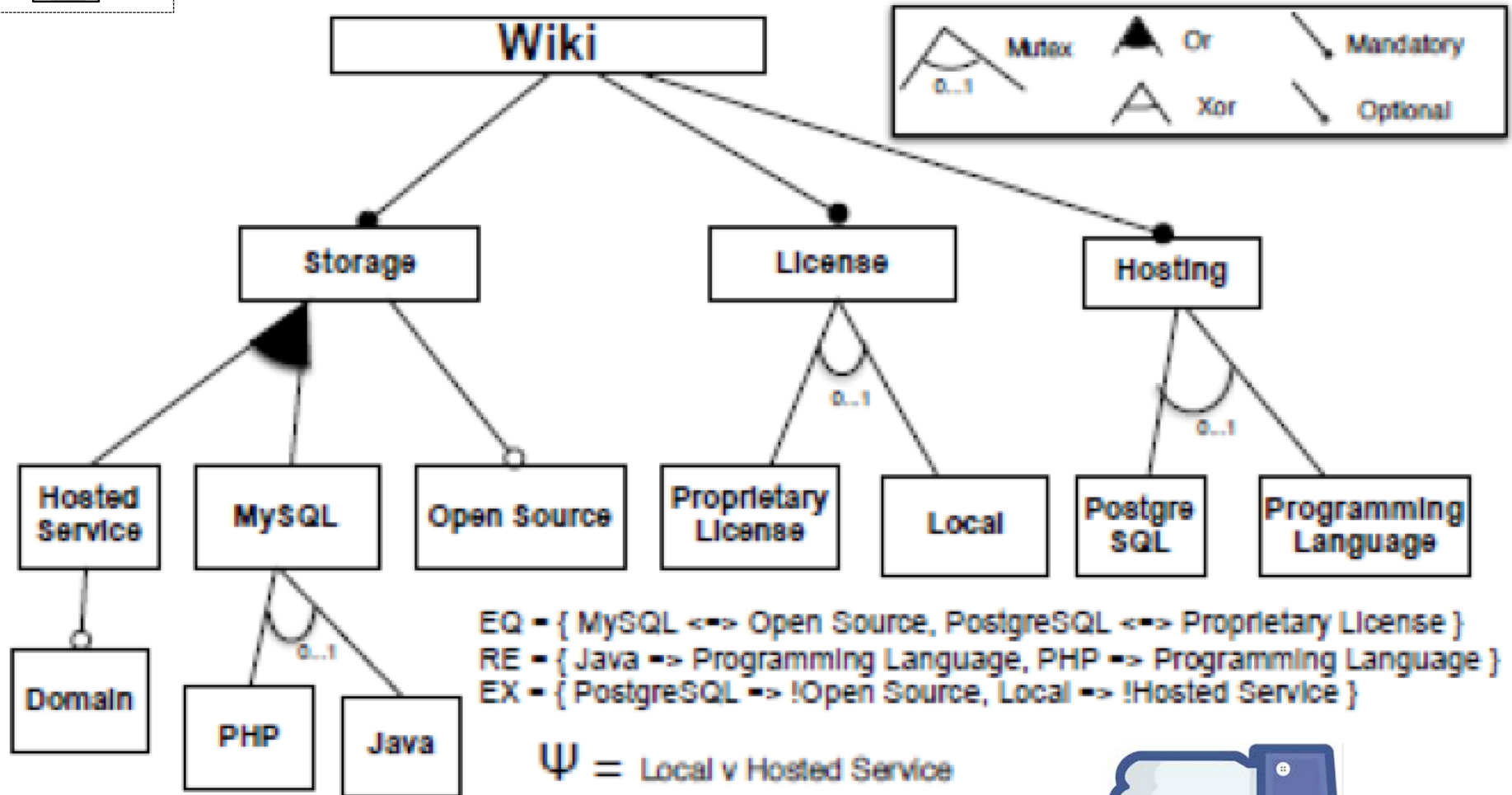
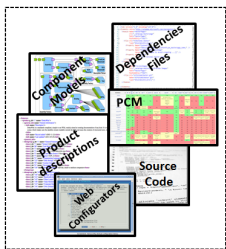
For a given configuration set, many (maximal) feature diagrams with different ontological semantics [She et al. ICSE'11, Andersen et al. SPLC'12, Acher et al. VaMoS'13]

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
PostgreSQL	✓	×	×	×	×	×	×	×	×	×
MySQL	×	✓	✓	✓	✓	✓	✓	✓	✓	✓
License	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Domain	✓	×	×	×	×	✓	×	✓	✓	×
Proprietary License	✓	×	×	×	×	×	×	×	×	×
Local	×	×	✓	×	✓	×	×	×	×	✓
Programming Language	×	✓	×	×	✓	✓	✓	✓	×	✓
Java	×	✓	×	×	×	✓	×	×	×	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PHP	×	×	×	×	✓	×	✓	✓	×	×
Open Source	×	✓	✓	✓	✓	✓	✓	✓	✓	✓
Wiki	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Hosting	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Hosted Service	✓	✓	×	✓	×	✓	✓	✓	✓	×

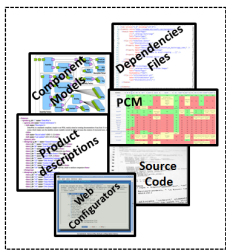
(a) Product comparison matrix (✓ feature is in the product ; × feature is not in the product)



Importance of ontological semantics (1)



Importance of ontological semantics (2)



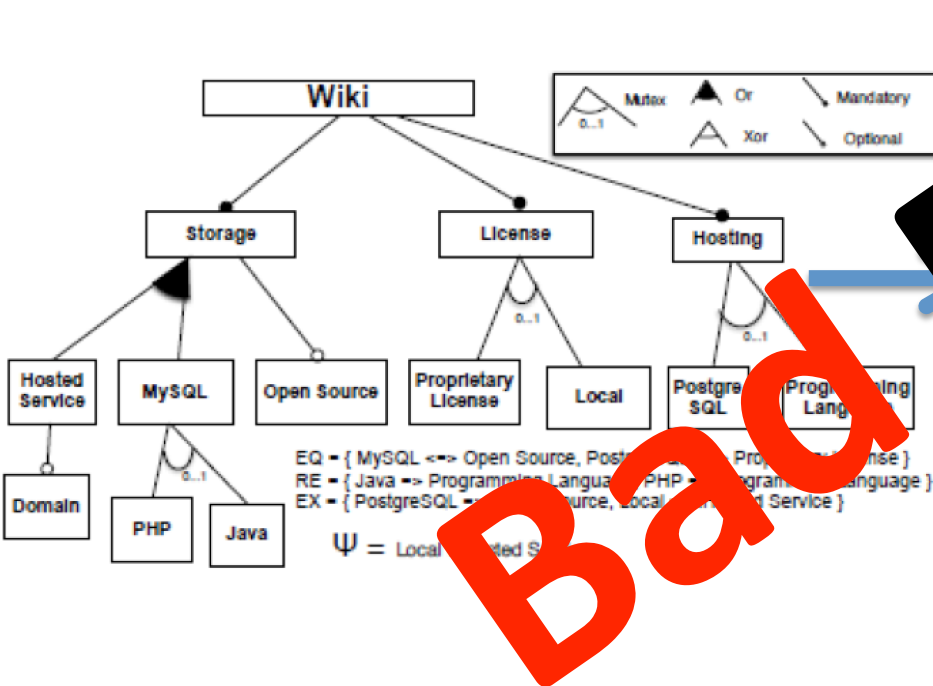
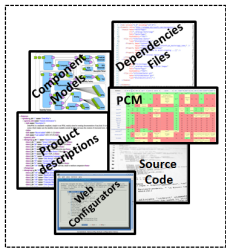
Wiki Wizard

Choose your Wiki

Storage	License	Hosting
Do you want to choose Open Source as your storage system? ☒ Yes, I want. ☐ No. Should your hosted service provide a Domain ? ☒ Yes. ☐ No. Choose your MySQL database: ☒ JAVA ☐ PHP ☐ None.	Select your License: ☐ Proprietary License ☒ Local ☐ None	What would you like to choose? ☐ PostgreSQL ☒ Programming Language ☐ None.



Importance of ontological semantics (3)



Wiki Wizard

Choose your Wiki

License

Select your License:

☐ Proprietary License

☒ Local

☐ None

Hosting

What would you like to choose?

☐ PostgreSQL

☒ Programming Language

☐ None

Should your hosted service provide a Domain ?

☒ Yes

☐ No

Choose your MySQL database:

☒ JAVA

☐ PHP

☐ None

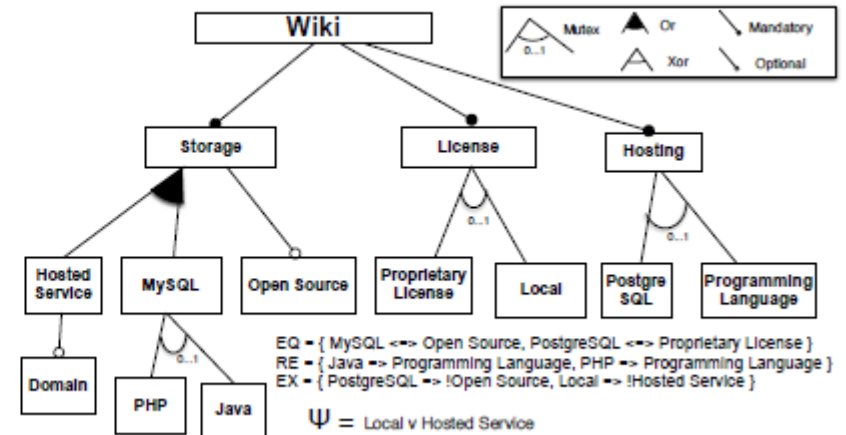
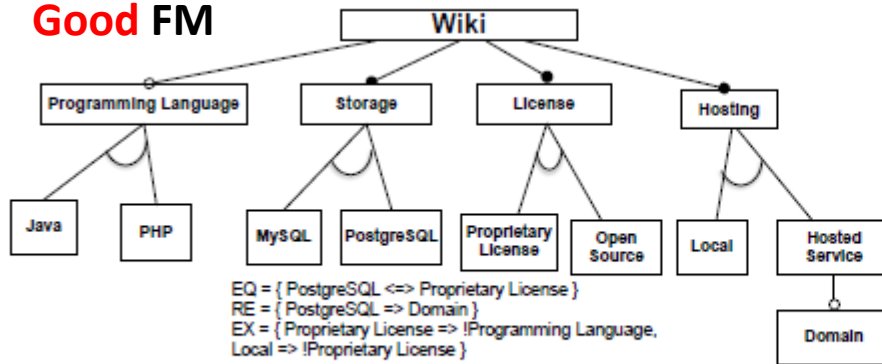


Communication
Comprehension
Forward engineering (e.g., generation)

Importance of ontological semantics (4)

Bad FM

Good FM



Good configuration interface

The Good configuration interface for Wiki Wizard shows a 'Choose your Wiki' window with three sections: 'Storage', 'License', and 'Hosting'. The 'Storage' section has two options: 'MySQL' (selected) and 'PostgreSQL'. The 'License' section has two options: 'Proprietary License' and 'Open Source' (selected). The 'Hosting' section has two options: 'Local' (selected) and 'Hosted Service'. Below the 'Hosting' section, there is a question: 'Should your hosted service provide a Domain ?' with two options: 'Yes.' and 'No.' (selected). The 'Programming Language' section has three options: 'JAVA' (selected), 'PHP', and 'None.'

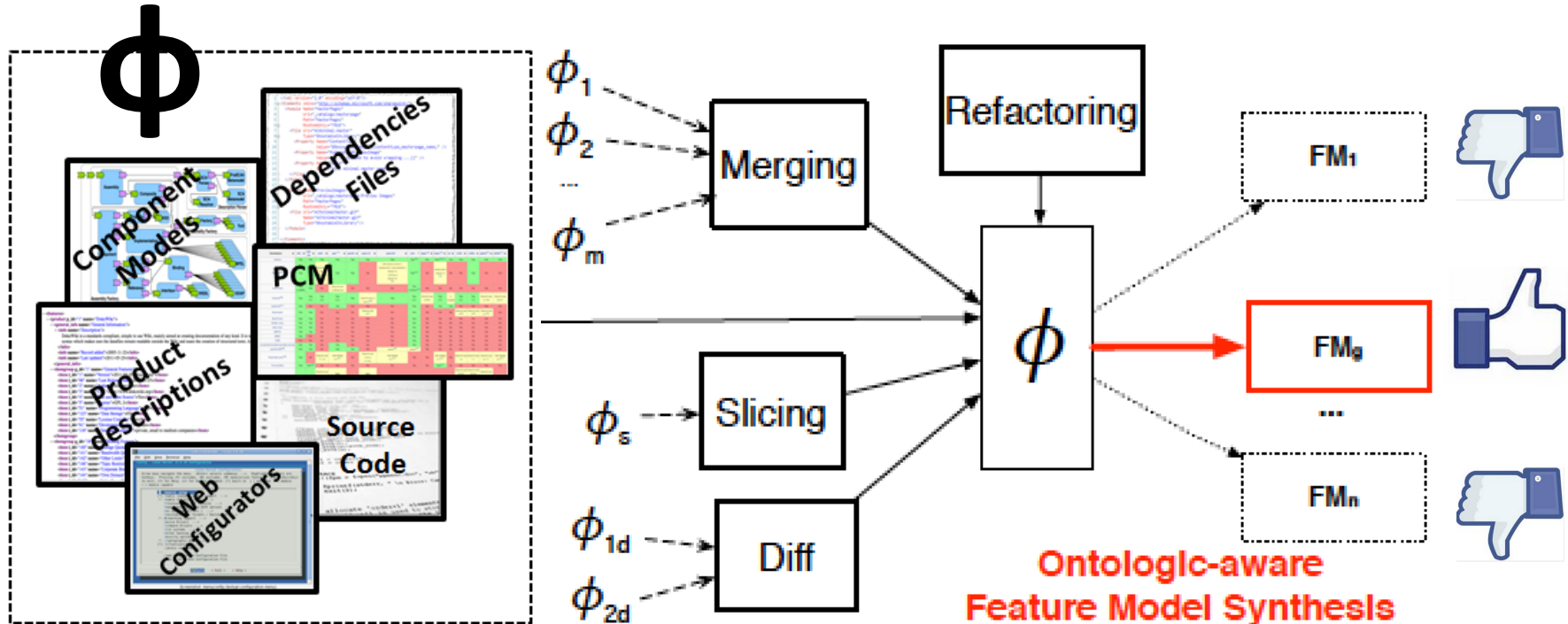
Bad configuration interface

The Bad configuration interface for Wiki Wizard shows a 'Choose your Wiki' window with three sections: 'Storage', 'License', and 'Hosting'. The 'Storage' section has two options: 'Yes, I want.' (selected) and 'No.'. Below this, there is a question: 'Should your hosted service provide a Domain ?' with two options: 'Yes.' (selected) and 'No.'. The 'License' section has three options: 'Proprietary License', 'Local' (selected), and 'None'. The 'Hosting' section has three options: 'PostgreSQL', 'Programming Language' (selected), and 'None.'. Below the 'Storage' section, there is a question: 'Choose your MySQL database:' with three options: 'JAVA' (selected), 'PHP', and 'None.'

Two product configurators generated from two FMs with the **same configuration semantics** but **different ontological semantics**.

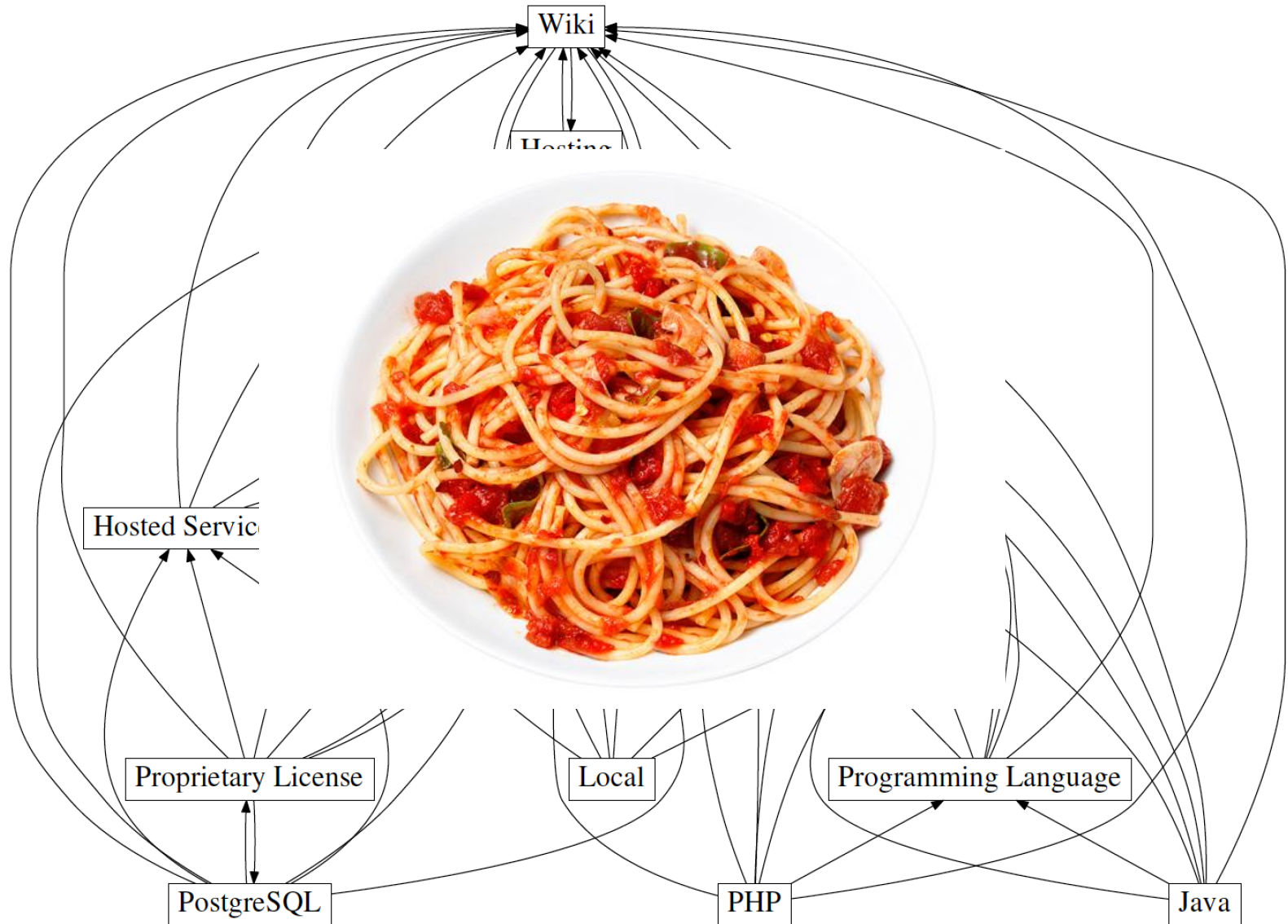
Most of the existing approaches neglect either configuration or ontological semantics.

We want both!



Fundamental Problem

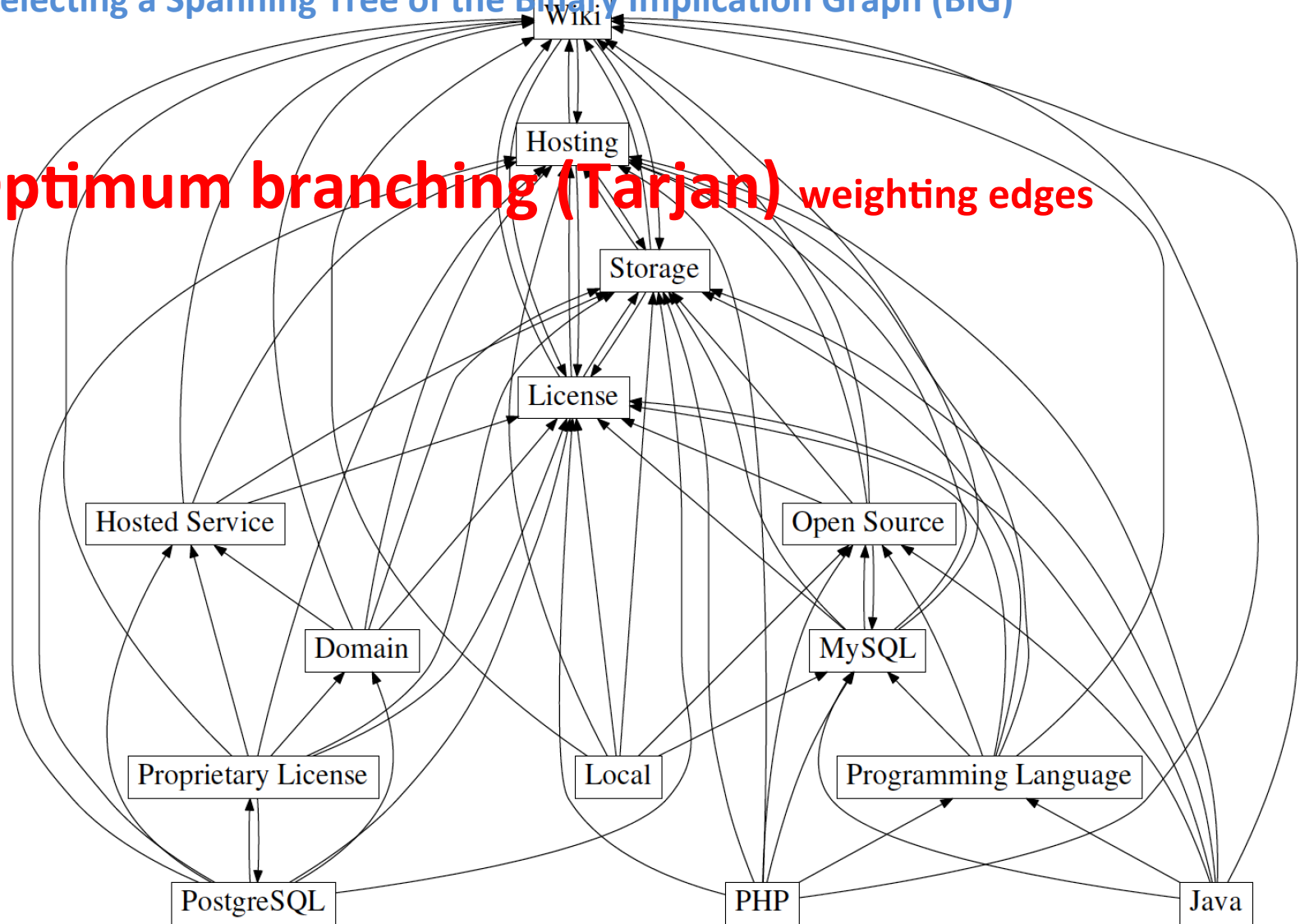
Selecting a Spanning Tree of the Binary Implication Graph (BIG)



Fundamental Problem

Selecting a Spanning Tree of the Binary Implication Graph (BIG)

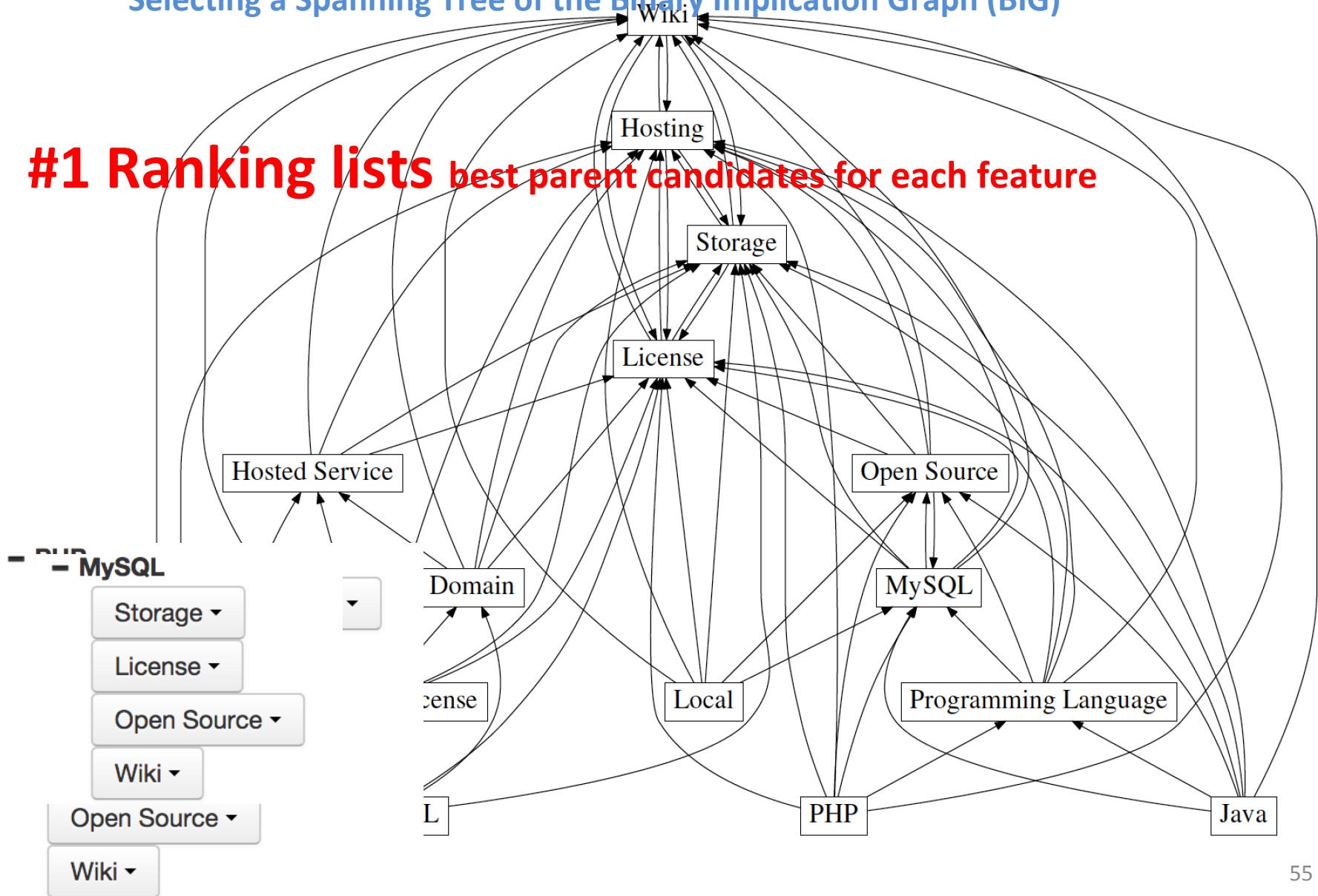
#0 Optimum branching (Tarjan) weighting edges



Fundamental Problem

Selecting a Spanning Tree of the Binary Implication Graph (BIG)

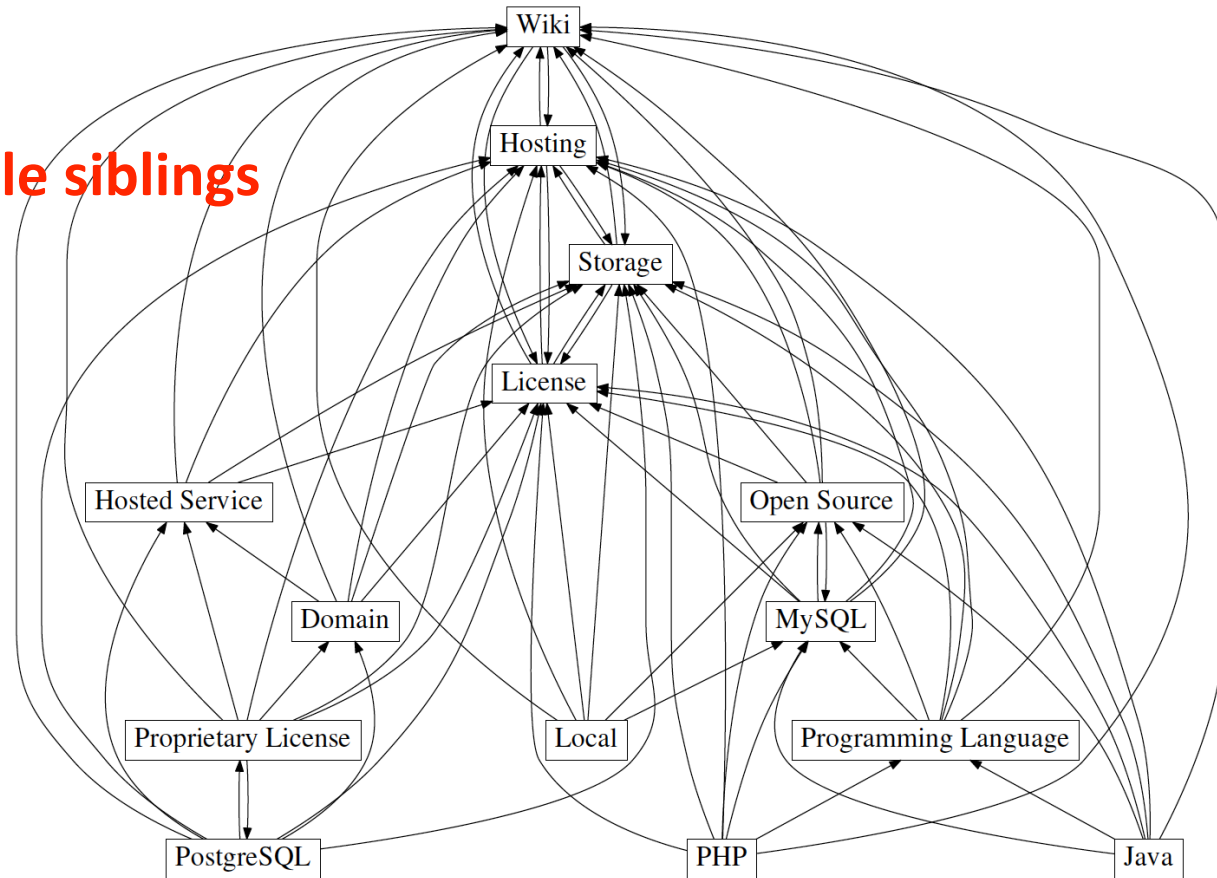
#1 Ranking lists best parent candidates for each feature



Fundamental Problem

Selecting a Spanning Tree of the Binary Implication Graph (BIG)

#2 Clusters ~possible siblings



Clusters



- License
- Proprietary License



- ☒ PostgreSQL
- ☒ MySQL

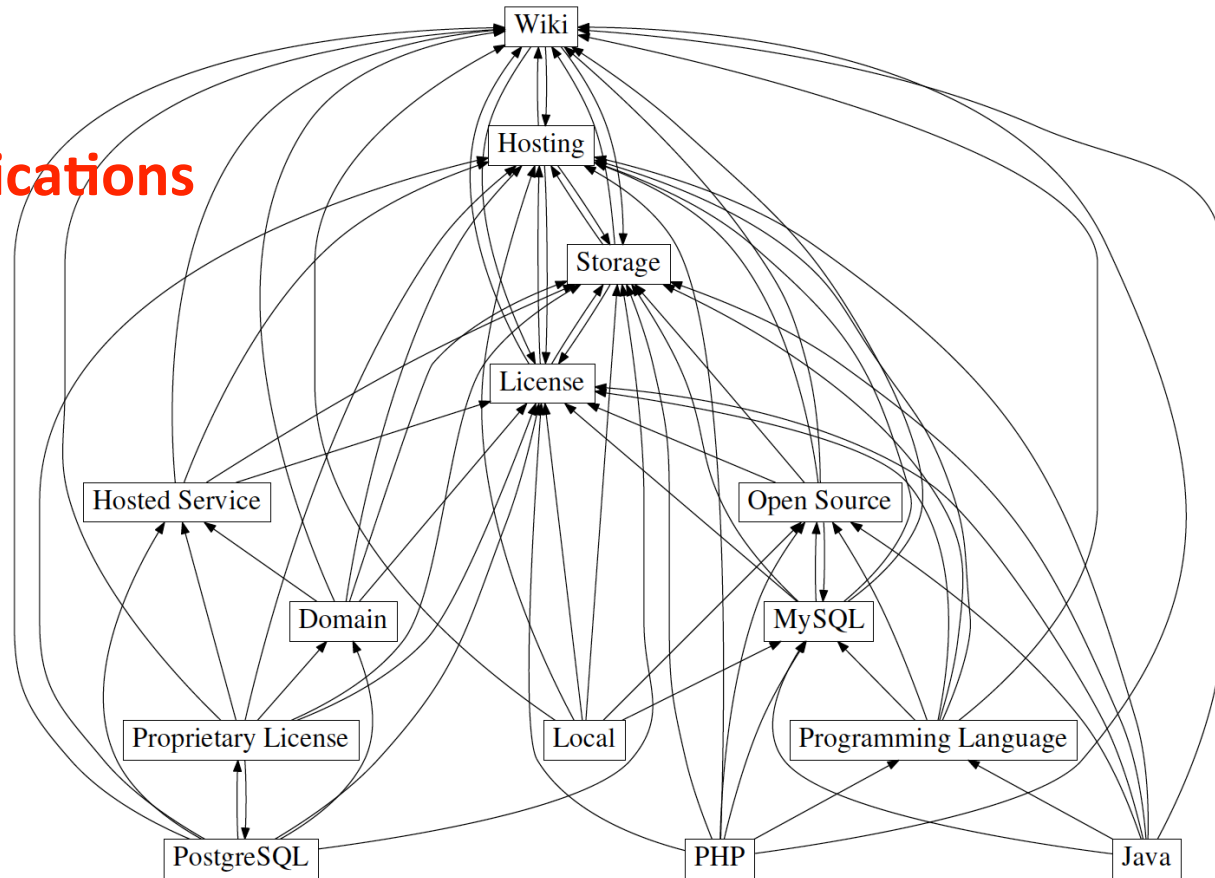
Fundamental Problem

Selecting a Spanning Tree of the Binary Implication Graph (BIG)

#3 Cliques ~bi-implications

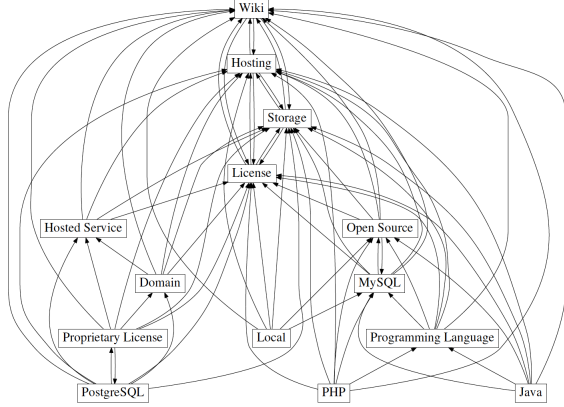
Cliques

- - Storage
 - License
 - Wiki
- - PostgreSQL
 - Proprietary License
- - MySQL
 - Open Source

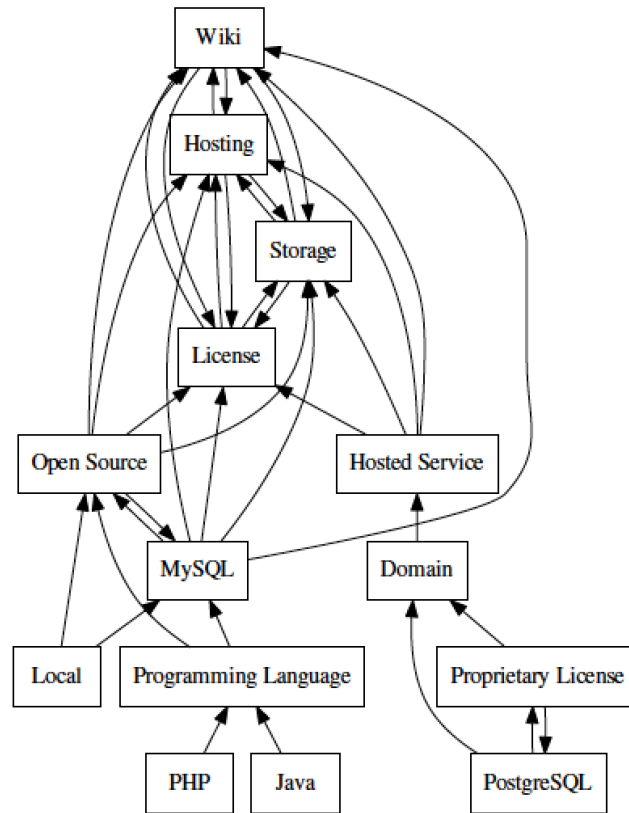


Fundamental Problem

Selecting a Spanning Tree of the Binary Implication Graph (BIG)

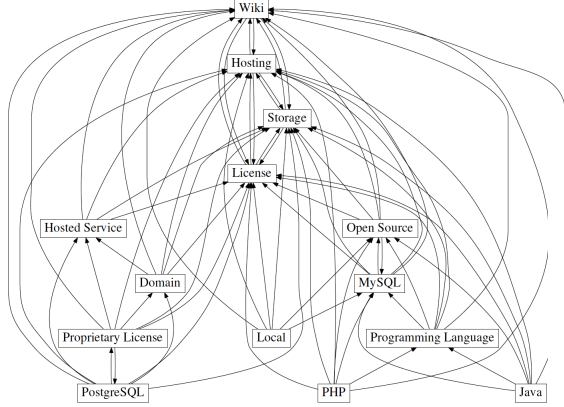


#4 small BIG



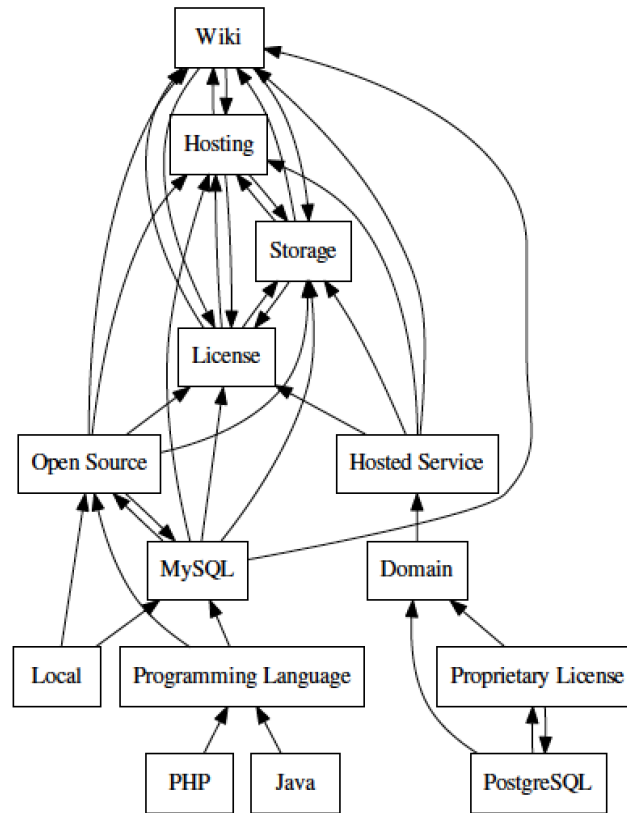
Fundamental Problem

Selecting a Spanning Tree of the Binary Implication Graph (BIG)



#4 reduced BIG

incomplete but dramatically reduce the problem



Ontological Heuristics

- For optimum branching, computing ranking lists and clusters
 - ~ « closedness » of features based on their names
- Syntactical heuristics
 - Smith-Waterman (SW) and Levenshtein (L)
- Wordnet
 - PathLength (PL) and Wu&Palmer (WP)
- Wikipedia Miner offers an API to browse Wikipedia's articles and compute their relatedness
 - Wiktionary (Wikt)

40 GB!

Familiar
Toggle Console
Enter filename...
Execute FAMILIAR code
Reset variables' environment
Save as...

repository
fmwiki.dimacs
user_study.fml
fm2.dimacs
fmwiki.fml
fm1.dimacs

FML Editor
KSynthesis
Ranking lists
Smith Waterm
Clustering
Smith Waterm
0,5
Undo
Redo
Complete
Save

fmwiki
badfmwiki

Wiki
Programming Language
License
Storage
Java
PHP
Open Source

Ranking Lists
Clusters
Cliques

```

FAMILIAR interpreter
Synthesising in progress... over fmwiki
fm1> compare fmwiki badfmwiki
res1 = REFACTORING
fm1>

```

Proprietary License
License
Wiki
Storage
PostgreSQL
MySQL

License
Proprietary License
Storage
PostgreSQL
MySQL

Storage
License
Wiki
PostgreSQL
Proprietary License
MySQL
Open Source

Wiki: License Storage ["Programming Language"]; License: ("Proprietary License"|"Open Source"); Storage: (PostgreSQL|MySQL); "Programming Language": (Java|PHP); ("Proprietary License" -> !"Programming Language"); (PostgreSQL <-> "Proprietary License");

61

Support and Empirical Study (1)

Goal: evidence and empirical insights of what heuristics are effective and what support is needed in WebFML

- Dataset
 - 120+ feature models of SPLOT
 - 30+ product comparison matrices from Wikipedia (see Becan et al. ASE'14 and ASE'13)
 - Ground truths are known
- Effectiveness of techniques (reduced BIG + ontological heuristics)
 - One shot synthesis
 - Quality of the ranking lists (top 2), clusters
 - Comparison with randomized and existing techniques

Support and Empirical Study (2)

Default heuristics/support has been determined through an empirical study

- Dataset
 - 120+ feature models of SPLOT
 - 30+ product comparison matrices from Wikipedia (see Becan et al. ASE'14 and ASE'13)
 - Ground truths are known
- Effectiveness of techniques (reduced BIG + ontological heuristics)
 - One shot synthesis
 - Quality of the ranking lists (top 2), clusters
 - Comparison with randomized and existing techniques

Support and Empirical Study (3)

Default heuristics/support has been determined through an empirical study

- One-step synthesis is far from the ground truth
 - despite state-of-the-art techniques we have developed
 - interactive support is thus crucial
- State-of-the-art heuristics for ranking lists and clusters
- Empirical insights on « cliques » and BIG reduction
 - e.g., support for unfolding of cliques

Support and Empirical Study (3)

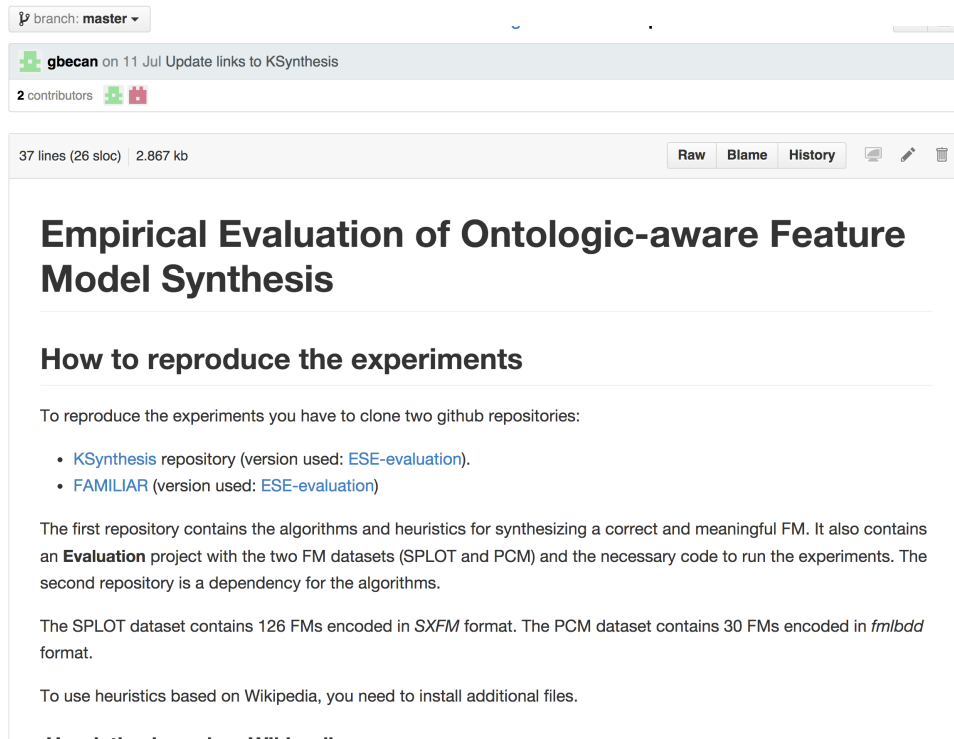
Breathing Ontological Knowledge Into Feature Model Synthesis: An Empirical Study

Guillaume Bécane, Mathieu Acher, Benoit Baudry, Sana Ben Nasr

► To cite this version:

Guillaume Bécane, Mathieu Acher, Benoit Baudry, Sana Ben Nasr. Breathing Ontological Knowledge Into Feature Model Synthesis: An Empirical Study. Empirical Software Engineering, Springer Verlag (Germany), 2015, pp.51. <10.1007/s10664-014-9357-1>. <hal-01096969>

 FAMILIAR-pi



The screenshot shows a GitHub repository page for the project "Empirical Evaluation of Ontologic-aware Feature Model Synthesis". The repository is owned by "gbecan" and has 2 contributors. It contains 37 lines of code (26 SLOC) and is 2.867 kb in size. The page title is "Empirical Evaluation of Ontologic-aware Feature Model Synthesis". Below the title, there is a section titled "How to reproduce the experiments". The text in this section states: "To reproduce the experiments you have to clone two github repositories: KSynthesis repository (version used: ESE-evaluation). FAMILIAR (version used: ESE-evaluation)". It then explains that the first repository contains algorithms and heuristics for synthesizing a correct and meaningful FM, and also contains an "Evaluation" project with two FM datasets (SPLOT and PCM) and the necessary code to run the experiments. The second repository is a dependency for the algorithms. It further details that the SPLOT dataset contains 126 FMs encoded in SXFM format, and the PCM dataset contains 30 FMs encoded in fmlbdd format. Finally, it mentions that to use heuristics based on Wikipedia, additional files need to be installed.

<https://hal.inria.fr/hal-01096969>

<http://tinyurl.com/OntoFMExperiments>

Reverse Engineering Feature Models ICSE'11 (Linux/eCos/FreeBSD case study)

Configuring FreeBSD:

```
# IPI_PREEMPTION relies on the PREEMPTION option

# Mandatory:
Device apic                # I/O apic

# Optional:
options MPTABLE_FORCE_HTTP #enable HTTP CPUs ...
options IPI_PREEMPTION
```

Code:

```
MODULE_DEPEND(at91_twi, iicbus, ...);
#ifdef A ... #endif
```

Features and dependencies are scattered through code and documentation.

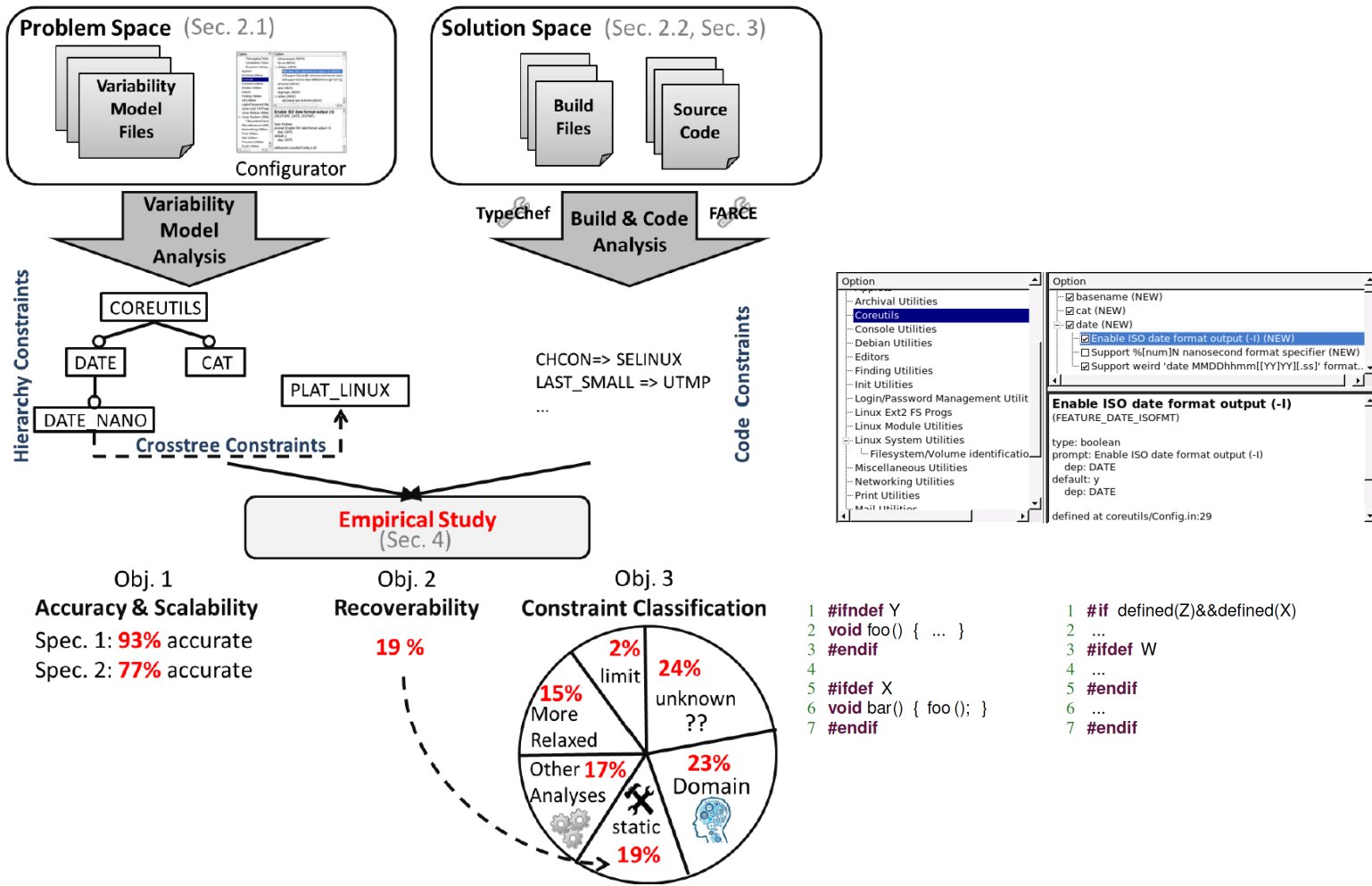
Mining Configuration Constraints: Static Analyses and Empirical Results

Sarah Nadi
University of Waterloo,
Canada

Thorsten Berger
IT University of
Copenhagen, Denmark

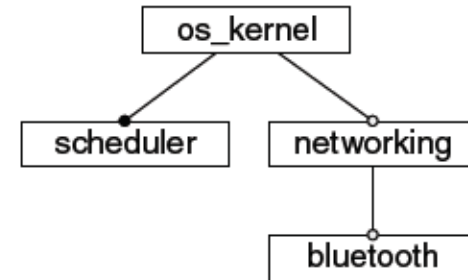
Christian Kästner
Carnegie Mellon
University, USA

Krzysztof Czarnecki
University of Waterloo,
Canada

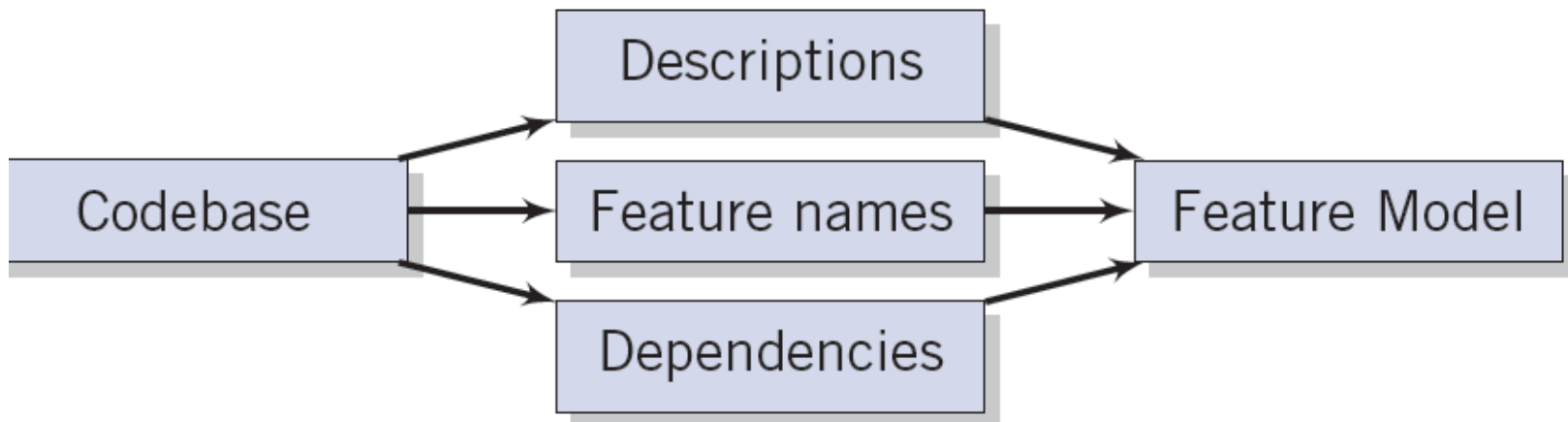


```
#ifdef A
  #ifndef B
    #error ...
  #endif
#endif
```

scheduler $\leftrightarrow$ os_kernel
networking $\rightarrow$ os_kernel
bluetooth $\rightarrow$ networking



bluetooth is a network driver.

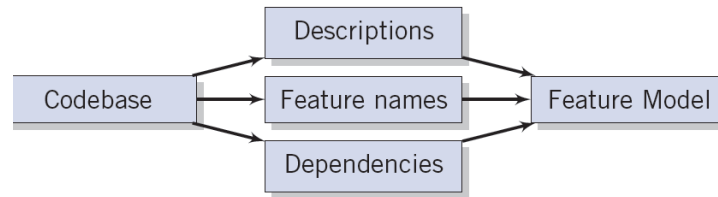
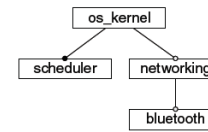


Leverage both names and descriptions for additional domain knowledge.

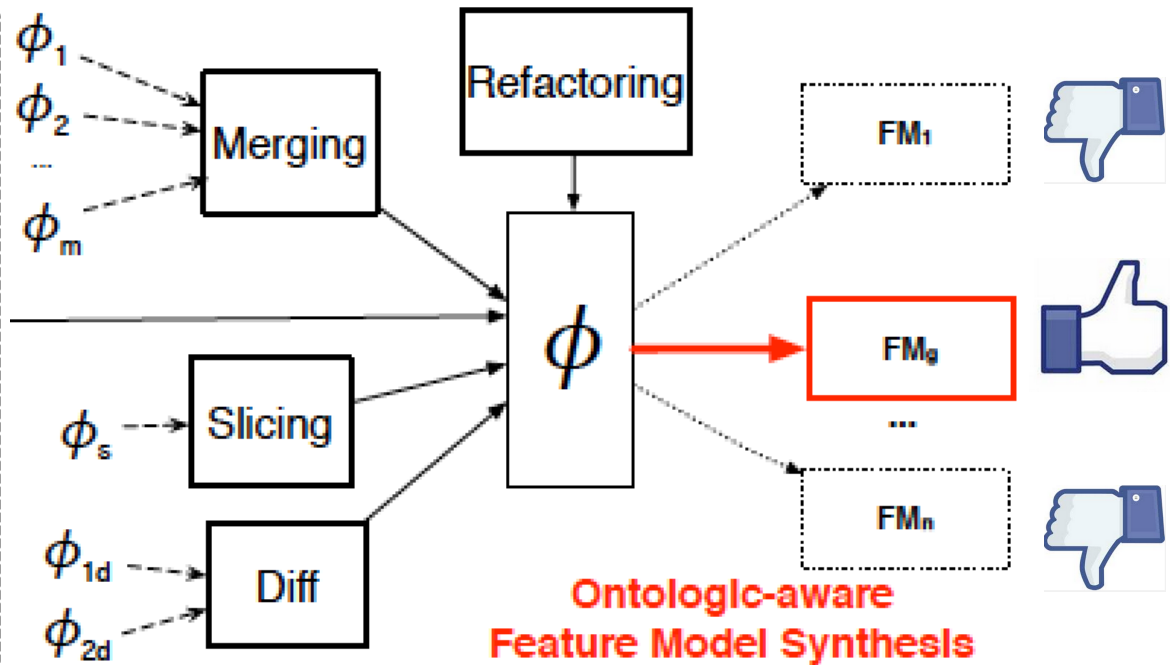
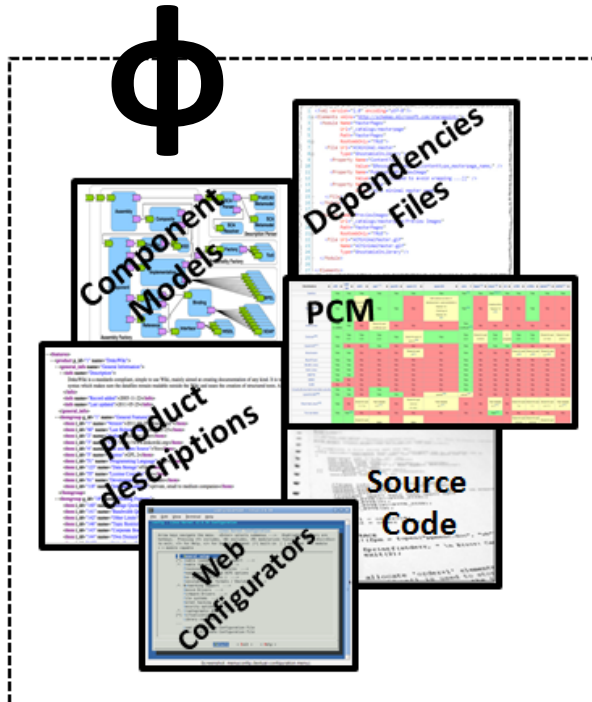
```
#ifdef A
#ifndef B
#error ...
#endif
#endif
```

```
scheduler ↔ os_kernel
networking → os_kernel
bluetooth → networking
```

bluetooth is a network driver.



Leverage both names and descriptions for additional domain knowledge.



Ontological heuristics (based on feature descriptions)

Feature similarity.

Feature names and descriptions

os_kernel Operating system.

scheduler I/O scheduling.

networking Networking drivers.

ethernet Type of local area networking.

Selecting a parent for:

bluetooth, a network driver.

Ontological heuristics (based on feature descriptions)

Feature similarity.

Feature names and descriptions

os_kernel Operating system.

scheduler I/O scheduling.

networking Networking **drivers**.

ethernet Type of local area networking.

Selecting a parent for:

bluetooth, a network **driver**.

Ontological heuristics (based on feature descriptions)

Feature similarity.

Feature names and descriptions

os_kernel Operating system.

scheduler I/O scheduling.

networking **Networking** drivers.

ethernet Type of local area **networking**.

Selecting a parent for:

bluetooth, a **network** driver.

Ontological heuristics (based on feature descriptions)

Feature similarity.

Ranked list

Feature names and descriptions

1. `networking` Networking drivers.
2. `ethernet` Type of local area networking.
3. `os_kernel` Operating system.
4. `scheduler` I/O scheduling.

Selecting a parent for:

`bluetooth`, a network driver.

Empirical results

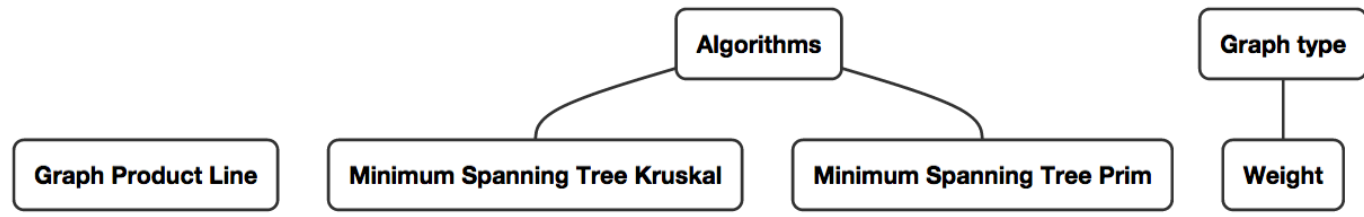
How many features have their reference parents ranked in the top 5 of our RIFs?

- Linux: 76% of features, eCos: 79% of features.
- Ignoring root features, 90% for Linux and 81% for eCos.
- For incomplete descriptions, At least 50% of words needed for good results (roughly 10 words in Linux).

Clusters

Preview

- ◢
 - Undirected
 - Connected Components
 - Directed
 - Strongly Connected
- ◢
 - Graph Product Line
 - Graph type
- ◢
 - Weight
 - Weighted
 - Unweighted
- ◢
 - Search
 - Depth-First Search
 - Breadth-First Search



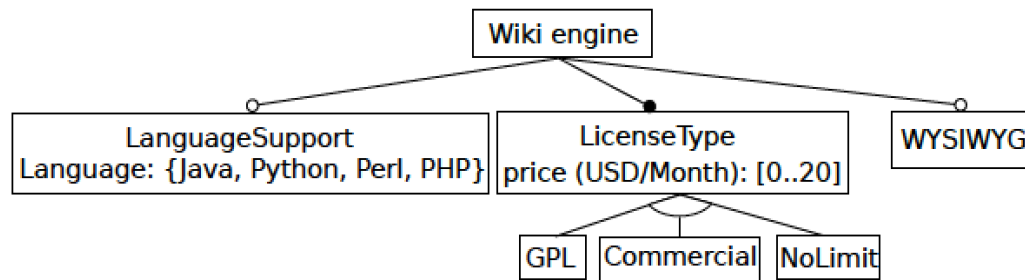
Algorithms ☐ ☐

- ☒ Minimum Spanning Tree Prim
- ☒ Minimum Spanning Tree Kruskal

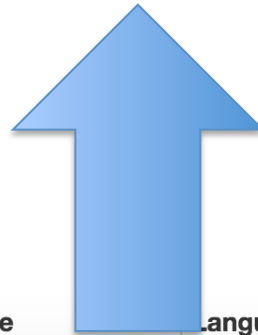
☐ ☐

- ☒ Minimum Spanning Tree Prim
- ☒ Minimum Spanning Tree Kruskal

Synthesising Attributed Feature Models



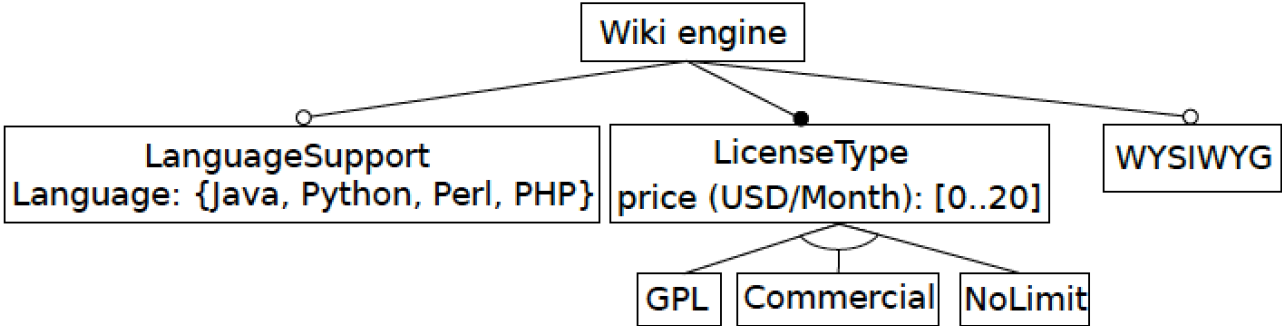
Commercial $\Leftrightarrow$ LicenseType.price = 10 GPL $\Rightarrow$ LanguageSupport
 Commercial $\Leftrightarrow$ Java NoLimit $\Rightarrow$ LicenseType.price $\geq$ 10
 GPL $\Rightarrow$ LicenseType.price $\leq$ 10 $\neg$ PHP $\Rightarrow$ WYSIWYG
 NoLimit $\Leftrightarrow$ $\neg$ LanguageSupport Python $\Rightarrow$ LicenseType.price = 0
 $\Phi = \neg$ WYSIWYG $\Leftrightarrow$ PHP $\wedge$ LicenseType.price = 0



Product ▲ ▼	License ▼	Price	Language Support ▼	Language ▼	WYSIWIG ▼
Find					
W1	Commercial	10	Yes	Java	Yes
W2	NoLimit	20	No		Yes
W3	NoLimit	10	No		Yes
W4	GPL	0	Yes	Python	Yes
W5	GPL	0	Yes	Perl	Yes
W6	GPL	10	Yes	Perl	Yes
W7	GPL	0	Yes	PHP	No
W8	GPL	10	Yes	PHP	Yes

Identifier	LicenseType	LicenseType.price	Language Support	Language	WYSIWYG
Confluence	Commercial	10	Yes	Java	Yes
PBwiki	NoLimit	20	No	–	Yes
MyWiki	NoLimit	10	No	–	Yes
MoinMoin	GPL	0	Yes	Python	Yes
TWiki	GPL	0	Yes	Perl	Yes
MyWiki2	GPL	10	Yes	Perl	Yes
MediaWiki	GPL	0	Yes	PHP	No
MyWiki3	GPL	10	Yes	PHP	Yes

(a) A configuration matrix for Wiki engines.



Commercial $\Leftrightarrow$ LicenseType.price = 10 GPL $\Rightarrow$ LanguageSupport
Commercial $\Leftrightarrow$ Java NoLimit $\Rightarrow$ LicenseType.price $\geq$ 10
GPL $\Rightarrow$ LicenseType.price $\leq$ 10 $\neg$ PHP $\Rightarrow$ WYSIWYG
NoLimit $\Leftrightarrow$ $\neg$ LanguageSupport Python $\Rightarrow$ LicenseType.price = 0
 $\Phi = \neg$ WYSIWYG $\Leftrightarrow$ PHP $\wedge$ LicenseType.price = 0

(b) An attributed feature model representing the configuration matrix in Figure 2(a)

Upload a Matrix

**Features
Attributes**

Hierarchy

Feature Groups

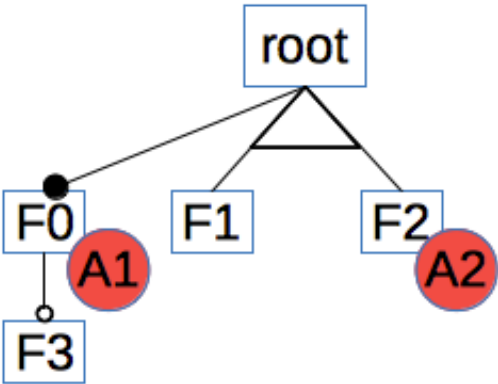
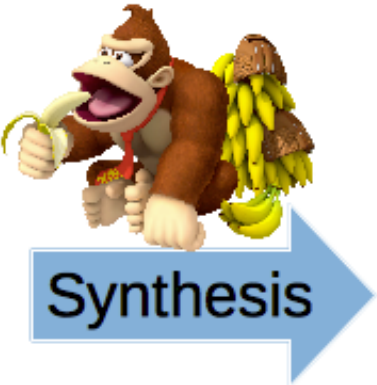
Constraints

AFM

Next

#	root	F0	F1	F2	F3	A1	A2
	Feature ▾	Feature ▾	Feature ▾	Feature ▾	Feature ▾	Attribute ▾	Attribute ▾
	Null value	Null value	Null value	Null value	Null value	0	0
0	Yes	Yes	Yes	No	Yes	3	0
1	Yes	Yes	No	Yes	Yes	2	2
2	Yes	Yes	Yes	No	No	2	0
3	Yes	Yes	No	Yes	No	0	8

root	F0	F1	F2	F3	A1	A2
Yes	Yes	Yes	No	Yes	3	0
Yes	Yes	No	Yes	Yes	2	2
Yes	Yes	Yes	No	No	2	0
Yes	Yes	No	Yes	No	0	8

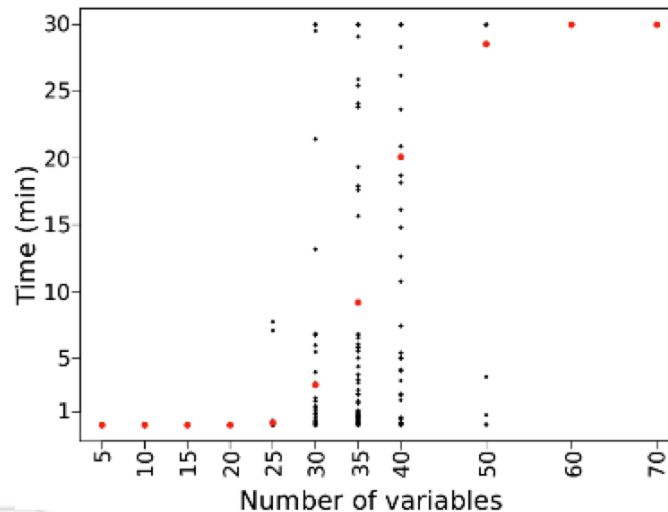


$A2 < 8 \Rightarrow A1 \geq 2$
 $A1 > 0 \Rightarrow A2 \leq 2$

Scalability

Random dataset

- Generator of configuration matrices
 - Number of variables (features + attributes)
 - Number of configurations
 - Maximum domain size (number of distinct values in a column)
- Execution time of or-group computation



= default heuristics only

- 1000 configurations
- max domain size of 10

Timeout always reached with more than 60 variables

Or groups do not scale !

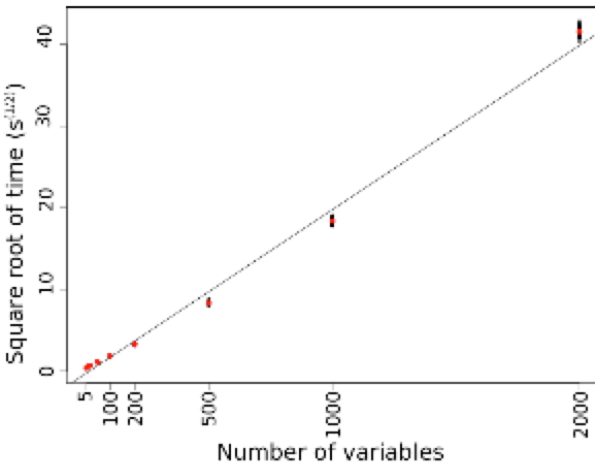
Scalability

Random dataset

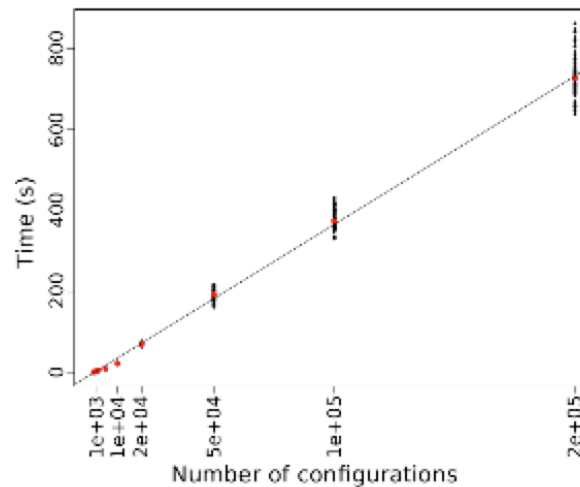
- Execution time (no or-groups)



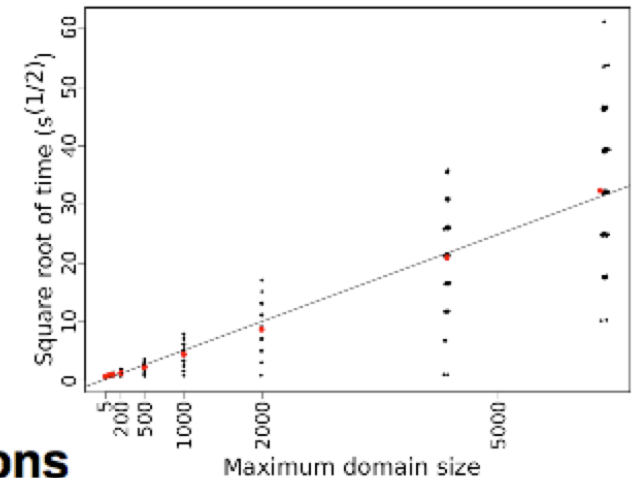
= default heuristics only



- **Up to 2,000 variables**
- 1,000 configurations
- Max domain size of 10



- 100 variables
- **Up to 200,000 configurations**
- Max domain size of 10



- 10 variables
- 10,000 configurations
- **Up to 6000 distinct values**

On all experiments:
Average: 2.6 min
Max: 62 min

Scalability

Best Buy dataset

- 242 matrices
- < 25% of empty cells
- Interpretation of empty cells



= default heuristics only

Media Card Reader	Yes	Yes	Yes
Number Of Ethernet Ports	1	1	
Number Of HDMI Outputs	1	1	1
Number Of USB Port(s)	3	3	3
Number Of VGA Ports	1		
Operating System	Windows 8.1	Windows 8.1	Windows 8.1

Execution time of 2.1s for the most challenging matrix:

- 77 variables
- 185 configurations
- Maximum domain size of 185

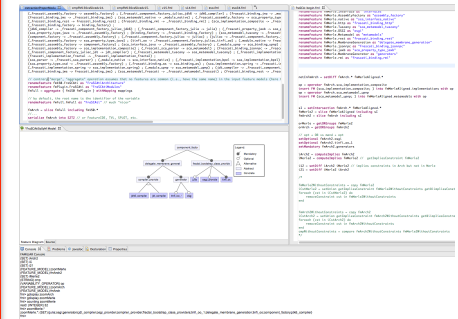
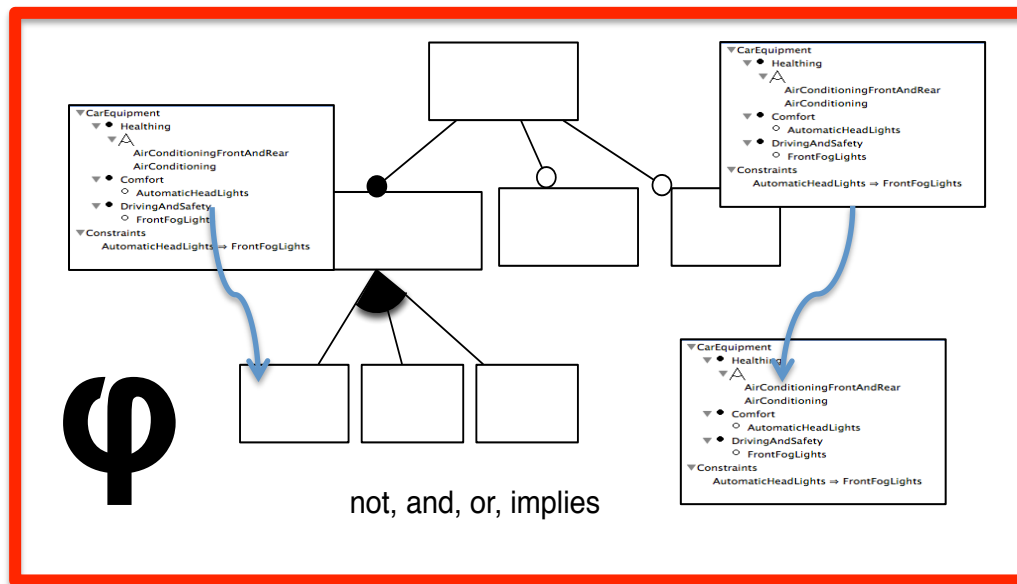
Execution time is similar to the random dataset

Managing Feature Models

FAMiLiAR

(FeAture Model script Language for manipulation and Automatic Reasoning)

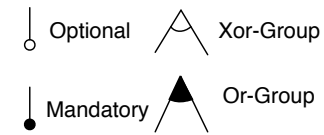
<http://familiar-project.github.com/>



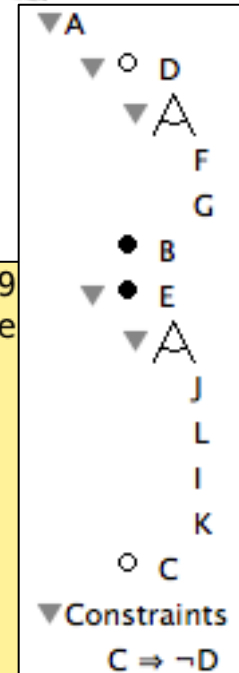
importing, exporting, composing, decomposing, editing, configuring, reverse engineering, computing "difs", refactoring, testing, and reasoning about (multiple) variability models

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

Configuration



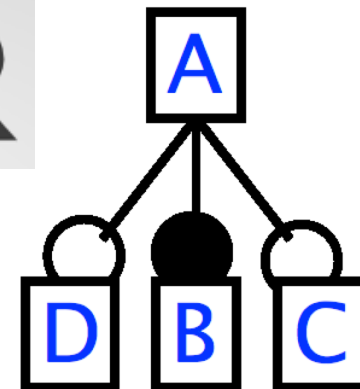
```
fml = FM (A: B [C] [D] E; D : (F|G) ; E : (I|J|K|L) ; C -> !D ; )
c1 = configuration fml
select C in c1
sc1 = selectedF c1 // accessors
cFM1 = asFM c1 // configuration and FM: back!
```



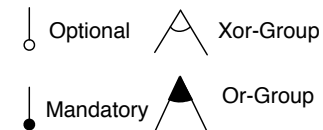
```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Re
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cFM1
cFM1: (FEATURE_MODEL) A: 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);
```

$A \wedge$
 $A \Leftrightarrow B \wedge$
 $C \Rightarrow A \wedge$
 $D \Rightarrow A$

FAMiliAR



FM



Φ

```

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

```

```

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

```

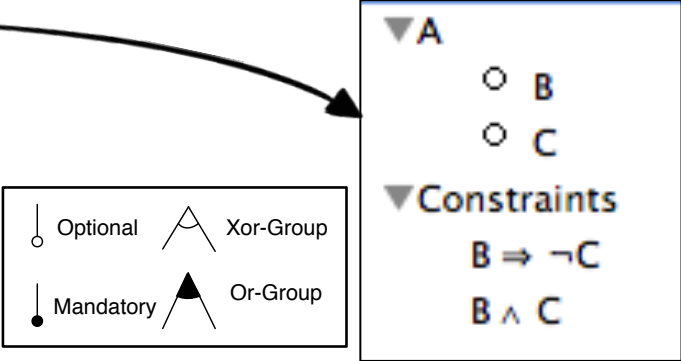
```

fm1> c1 = cores fm1
fm1> s1 c1: (SET) {B;A}
s1: (SET) {B;A}
fm1> c1bis = cores fm1bis
c1bis: (SET) {B;A}
fm1> compare fm1 fm1bis
res7: (STRING) REFACTORING {B;A;B;D}}
fm1> compare fm1bis fm1bisbis
res8: (STRING) REFACTORING {B;A};{B;A}}
fm1> s1 res7: (BOOLEAN) true
res3: (fm1> c1 eq c1bisbis)
fm1> s1 res6: (BOOLEAN) true
res4: (BOOLEAN) true

```

Operations for Feature Models (1)

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

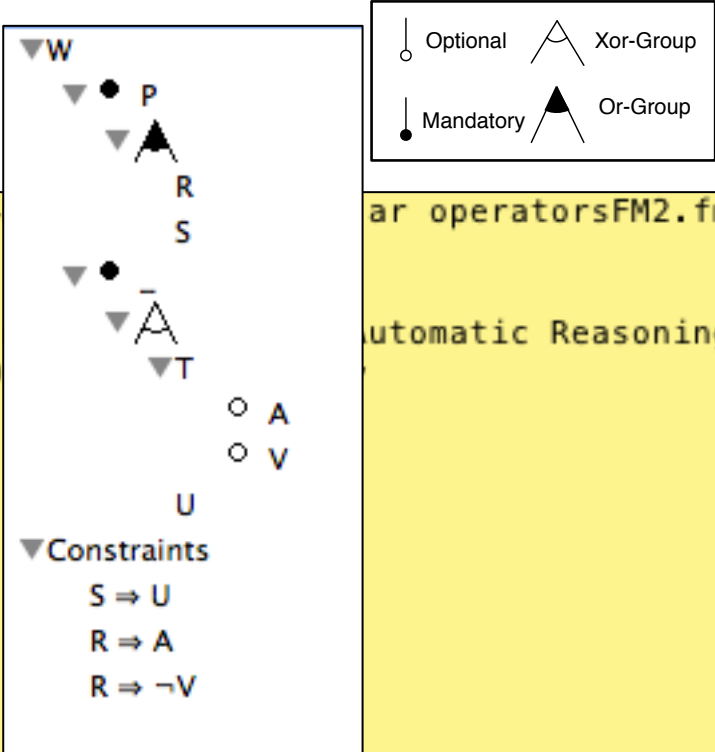


```
acher-scr:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar operatorsFM.fml
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> ls
(FEATURE_MODEL)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



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 -Xmx1024M operatorsFM2.fml
cores: {P;W}

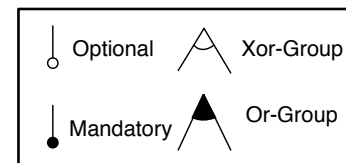
FAMILIAR (for FeAture Model scrIPt Language for
University of Nice Sophia Antipolis, UMR CNRS 600
https://nyx.unice.fr/projects/familiar/
fml> ls
(SET) fo1
(SET) dfm1
(SET) s1
(DOUBLE) c1
(BOOLEAN) b1
(FEATURE_MODEL) fm1
fml> c1
c1: (DOUBLE) 2.0
fml> fo1
fo1: (SET) {A}
fml> dfm1
dfm1: (SET) {V}
```

```
ar operatorsFM2.fml

Automatic Reasoning)
```



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
```

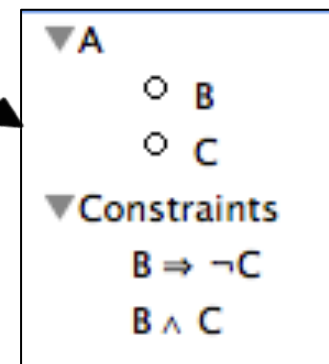
```
    println "removing constraint... ", cst
```

```
    removeConstraint cst in fm1
```

```
    c = counting fm1
```

```
    println "now the number of valid configurations is... ", c
```

```
end
```

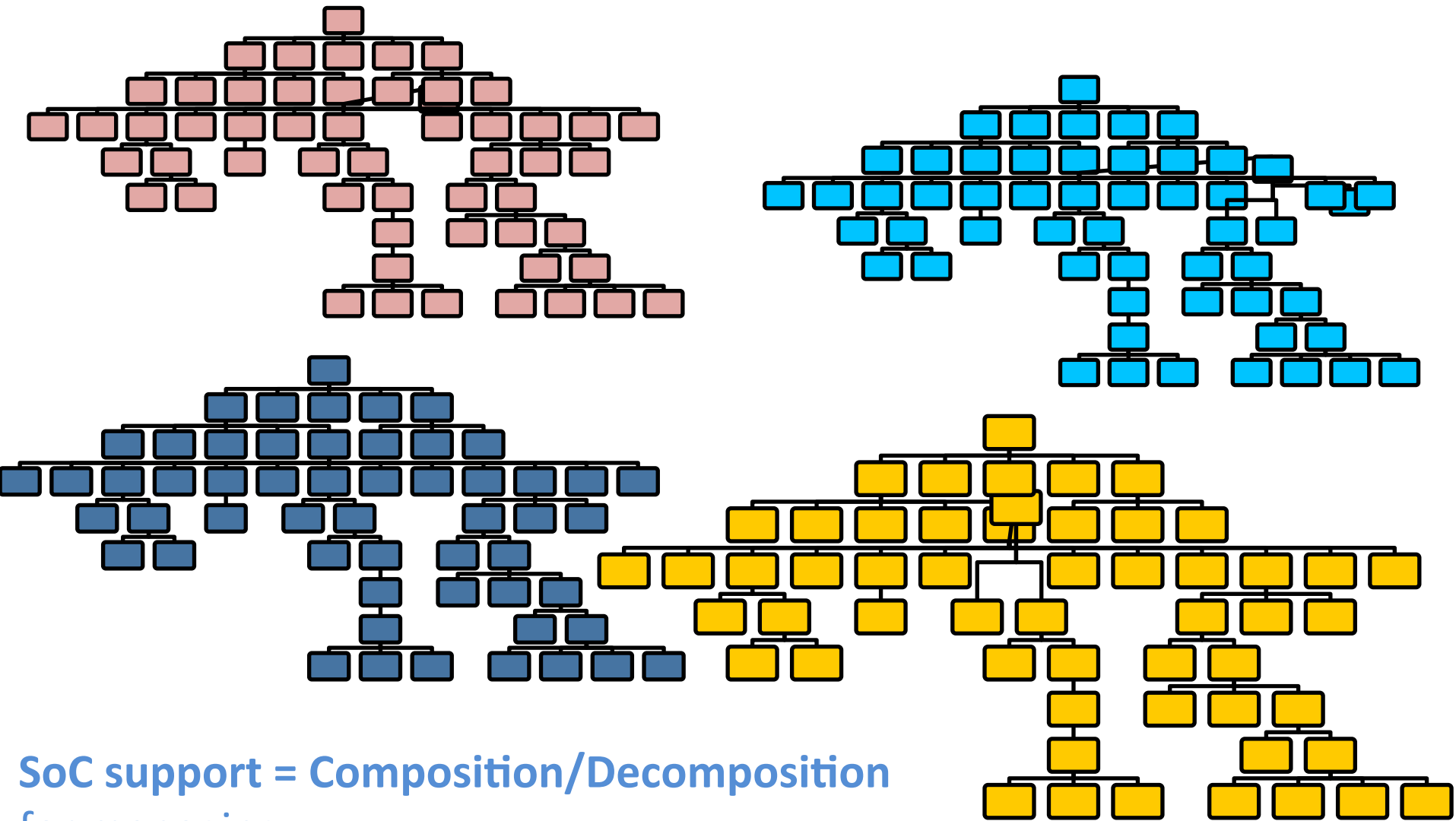


```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar operatorsFM3.fml
removing constraint... (B & C)
```

```
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.

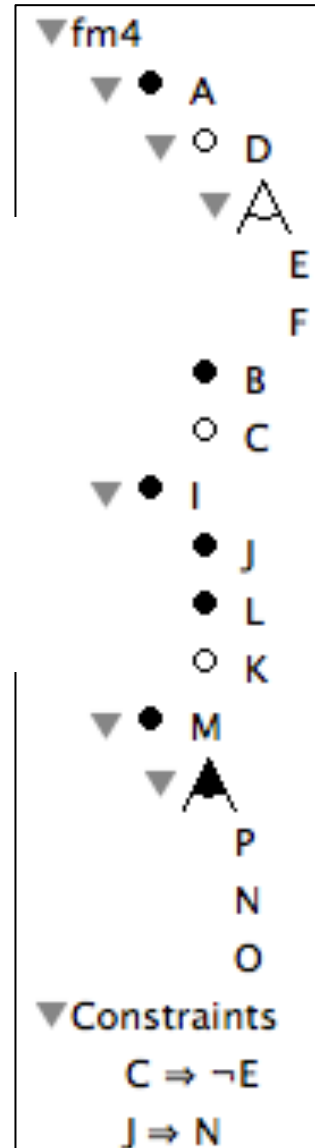
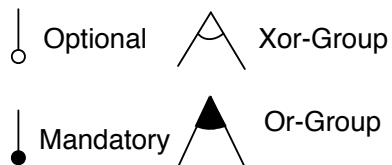
Composing Feature Models (1)

```

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

```



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

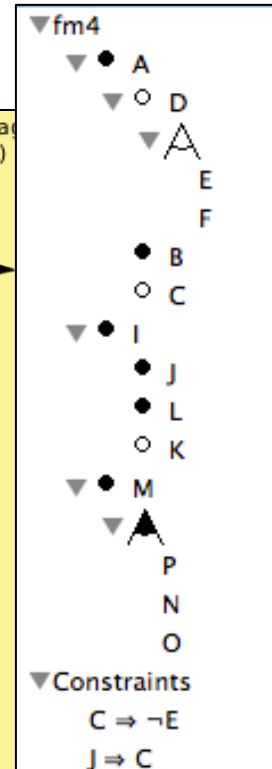
// composition sometimes leads to "anomalies"
dfm4 = deads fm4
```

```
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 ; )

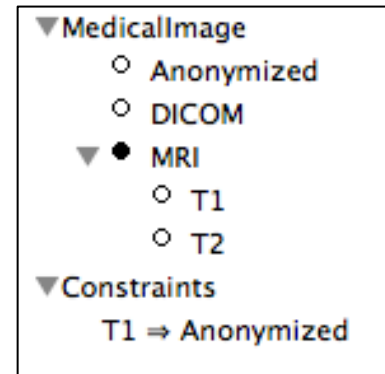
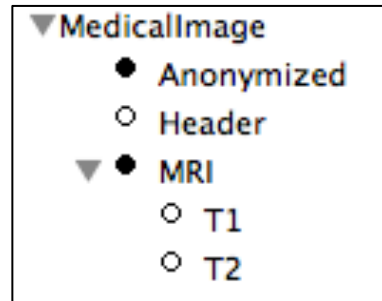
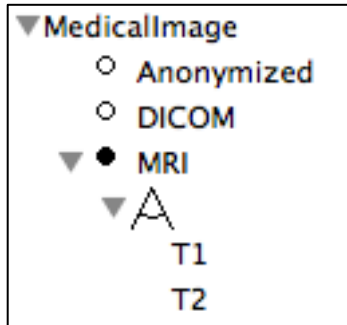
// equivalent to aggregate { fm1 fm2 fm3 }
fm4 = aggregate fm* withMapping cst
```

Previous version

```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar a
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);
```



Merging Feature Models (1)



```
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; )
```

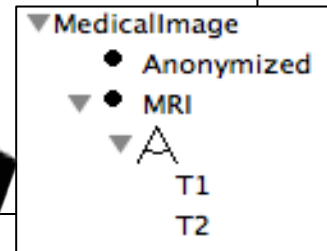
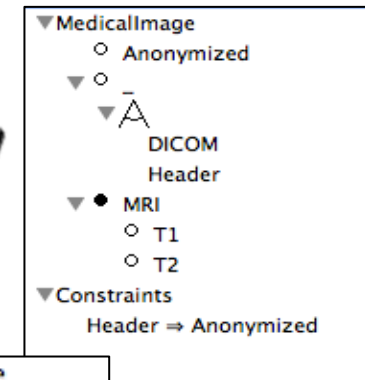
```
s1 = configs fmsupp1
s2 = configs fmsupp2
s3 = configs fmsupp3
```

```
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
```



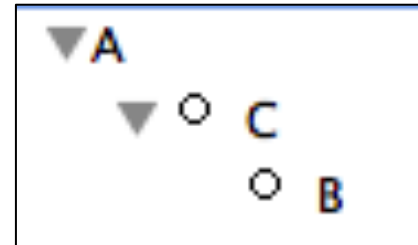
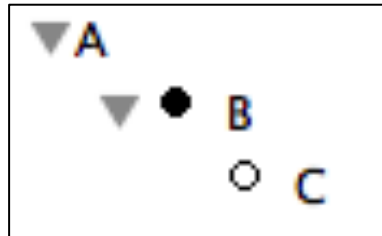
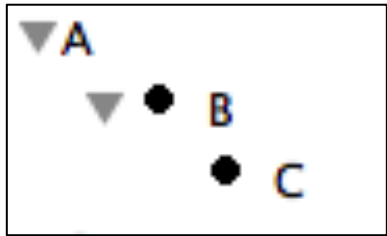
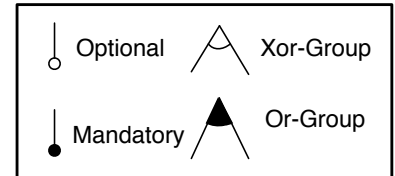
Merging operation (2)

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}}

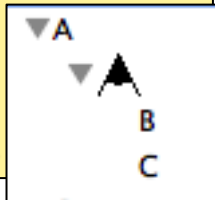
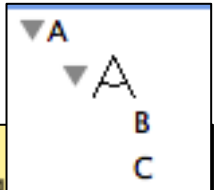
?

Comparing Feature Models

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

c
a
c

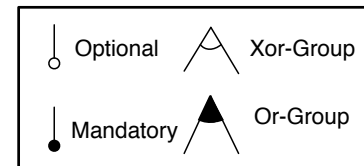
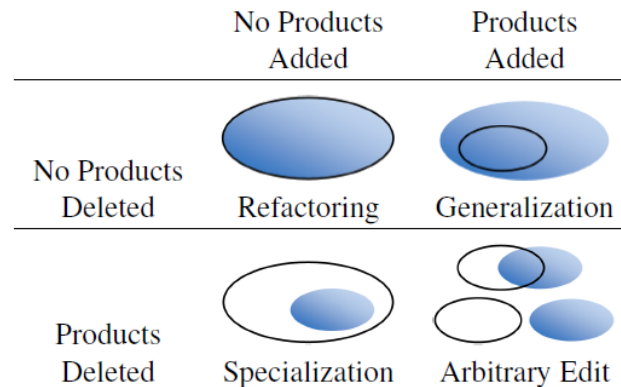
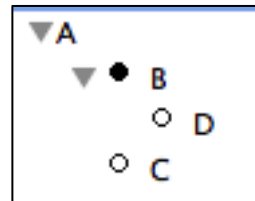
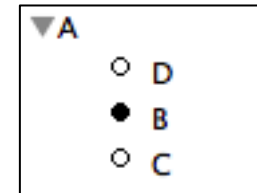
```
MacBook-Pro-de-Mathieu-2:FML-scripts macher$ java -jar -Xmx1024M ../FML-0.9.9.5.jar
FAMILIAR (for FeAture Model sCrIpt Language for manIpulation and Automatic Reasoning)
University of Nice Sophia Antipolis, UMR CNRS 6070, I3S Laboratory
https://nyx.unice.fr/projects/familiar/
fml> cmp23
cmp23: (STRING)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



```
assert (cmp10 eq GENERALIZATION)
```

// example taken from Automated Analysis of Feature Models

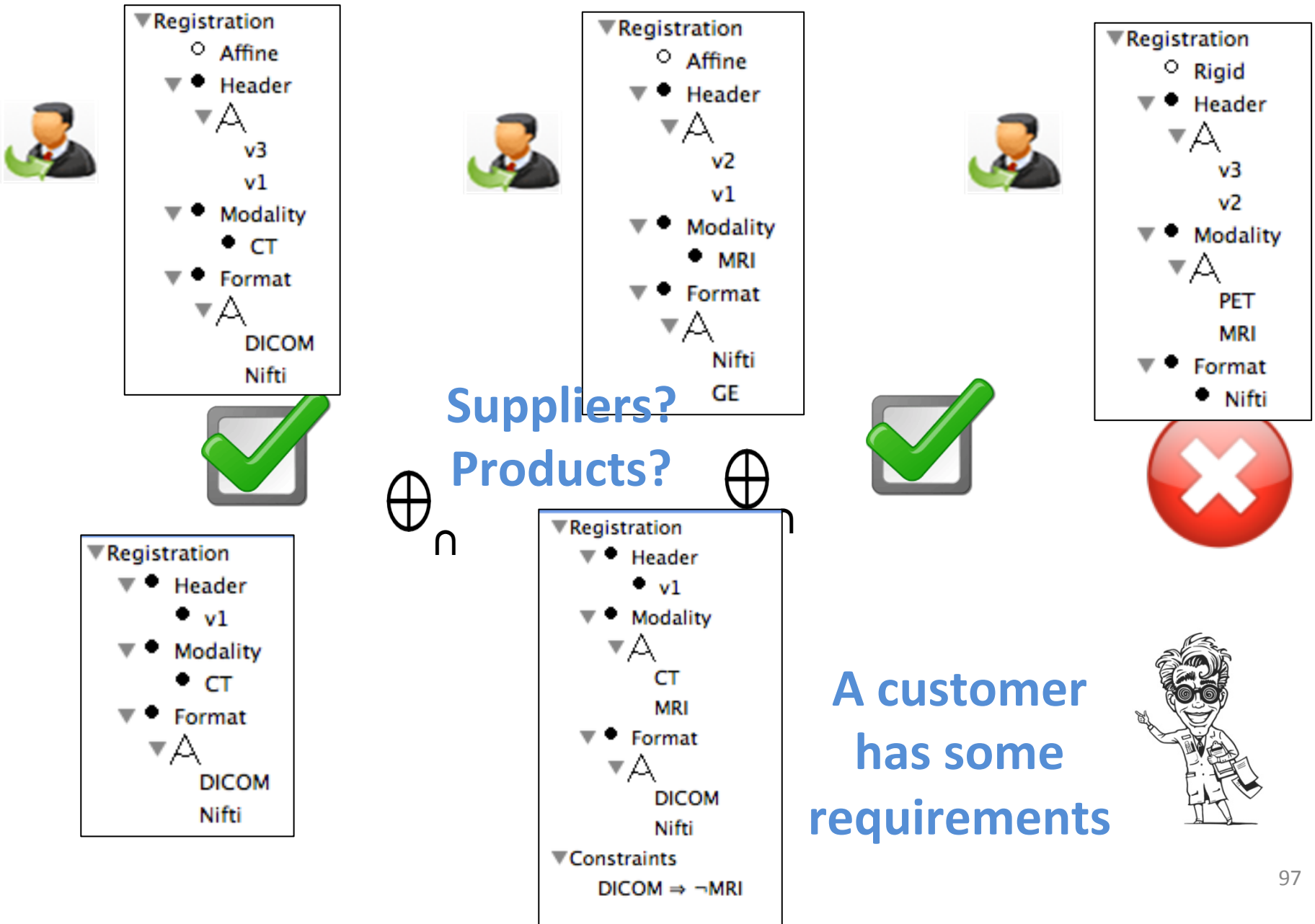
```
fm2 = FM (A: B [C] [D]; )
fm3 = FM (A: B [C]; B : [D]; )
cmp23 = compare fm2 fm3
```



FAMiliAR

Combining operators:
an example

Merge Intersection: Available Suppliers



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
```

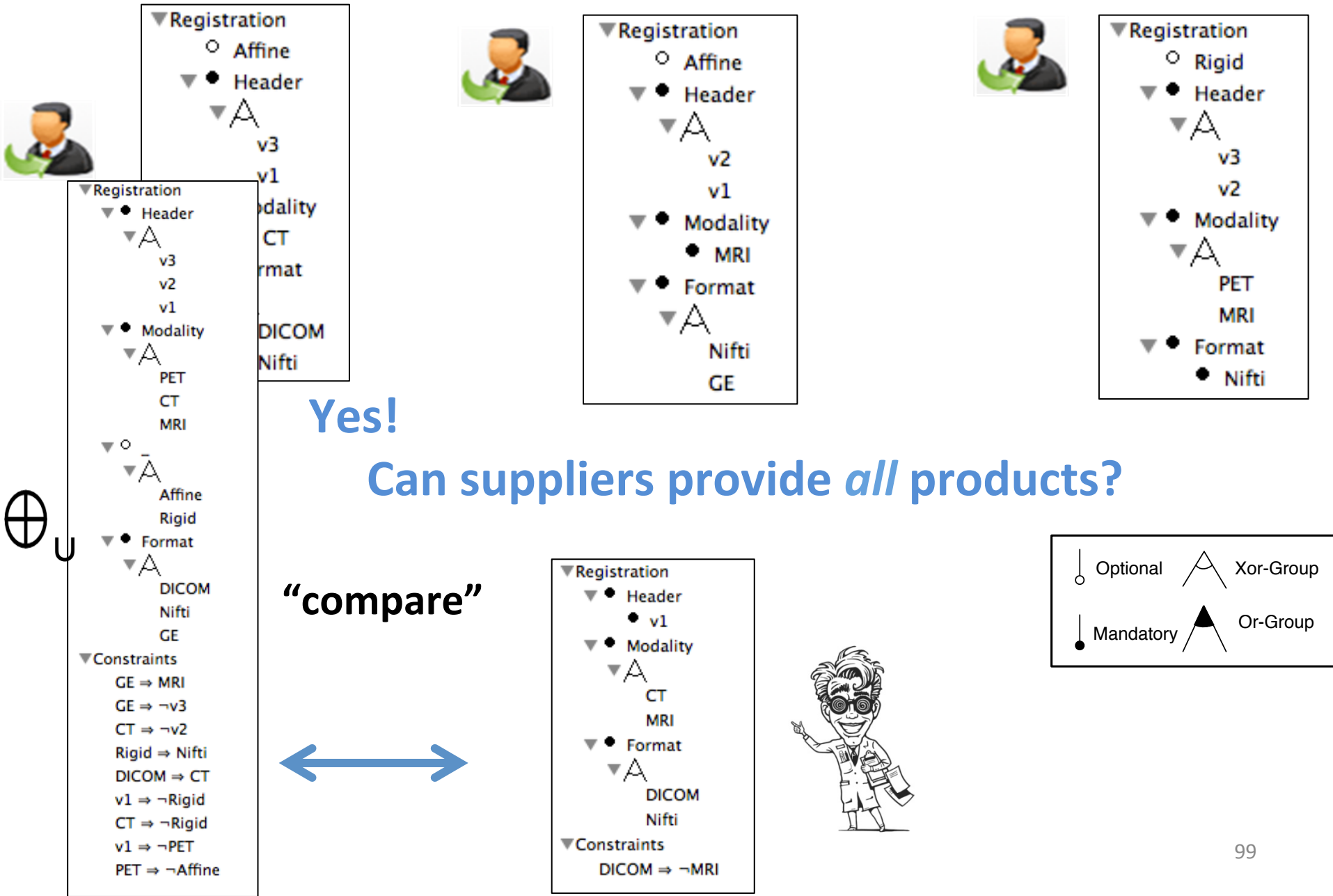
```
REGsupp1p: (FEATURE_MODEL) Registration: Header Modality Format ;  
Header: v1 ;  
Modality: CT ;  
Format: (DICOM|Nifti) ;
```

```
REGsupp1p = merge intersection { REGrequired REGsupp1 }
```

```
REGsupp2p = merge intersection { REGrequired REGsupp2 }
```

```
REGsupp3p = merge intersection { REGrequired REGsupp3 }
```

Merge Union: Availability Checking



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); )

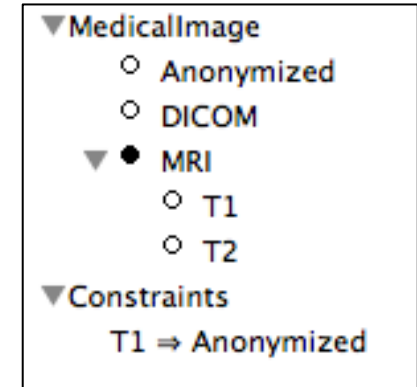
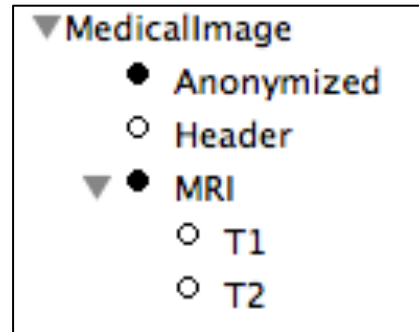
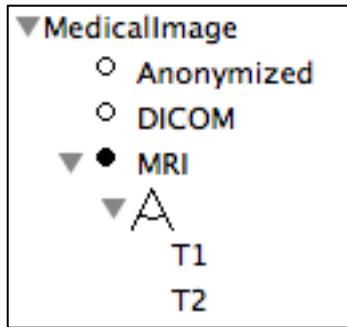
REGrequired = FM (Registration : Header Format Modality ;
                  Header : v1 ; //v3;
                  Format : (DICOM|Nifti) ;
                  Modality: (MRI|CT);
                  !DICOM or !MRI;
                  )

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
```

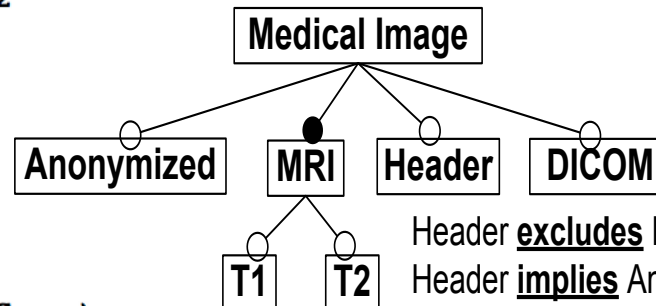
```
s123 = setUnion s3 setUnion s1 s2
```

// you WONT scale

//...

```
fmSupp = merge sunion fmsupp*
```

```
assert (size s123 eq counting fmSupp)
```

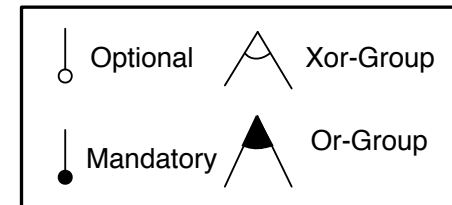


Header excludes DICOM

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

$s_0 = \{$

$\{MI, MA, F, CT, Nifti\},$

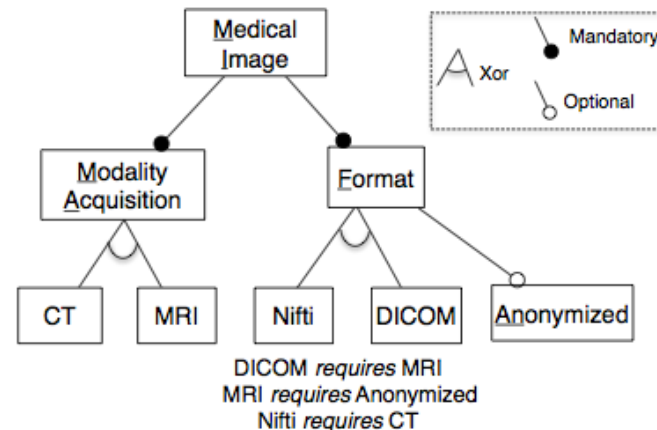
$\{MI, MA, F, CT, Nifti, AN\},$

$\{MI, MA, F, DICOM, MRI, AN\}$

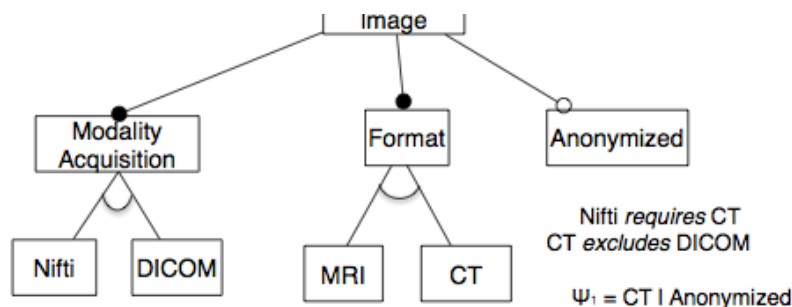
$\}$

Union
Intersection

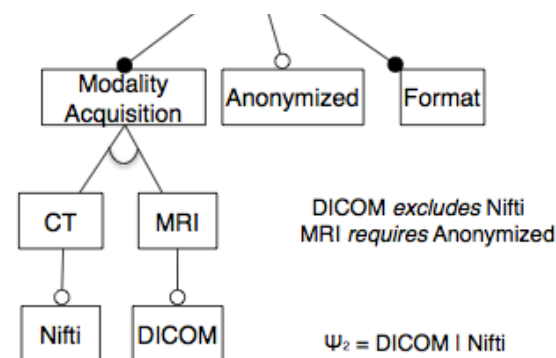
Diff



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



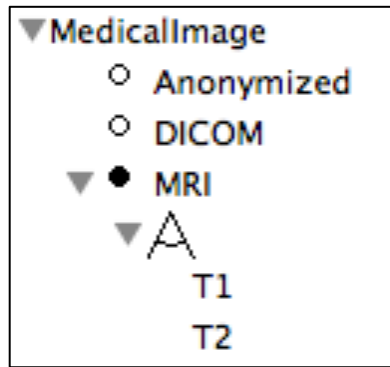
(c) fm_1



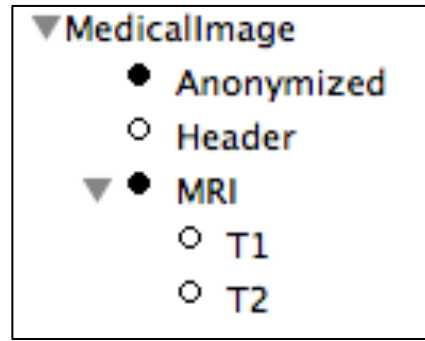
(d) fm_2

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

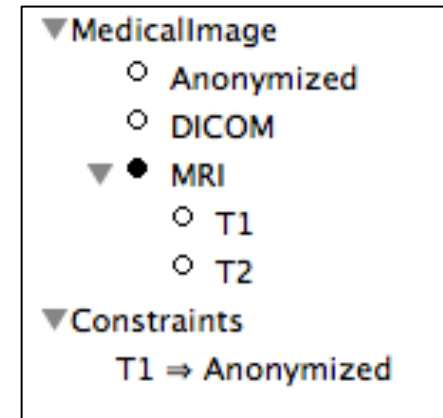
Merging operation: algorithm



Φ_1



Φ_2



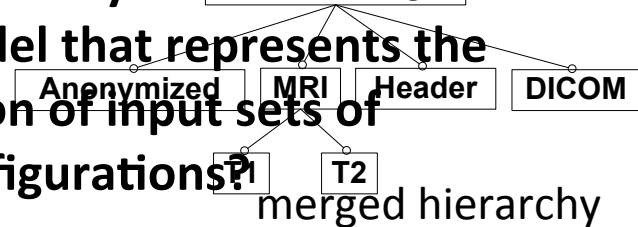
Φ_3

Φ_{123}

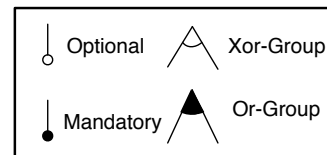
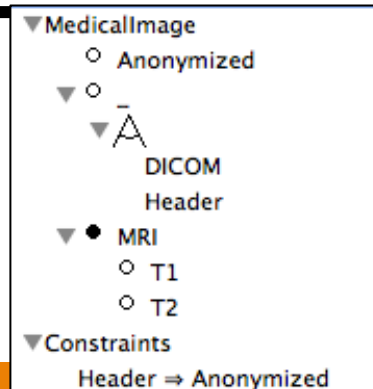
merged propositional formula

+

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]

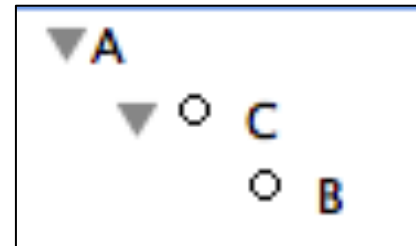
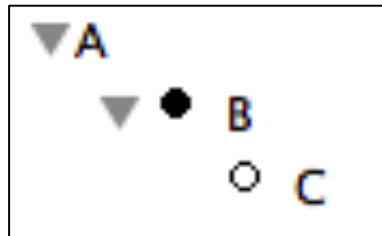
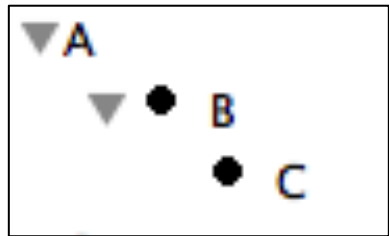
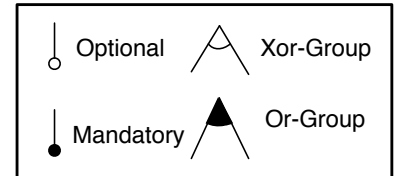
Merging operation: back to hierarchy

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]

The logo for FAMILiAR is displayed in a white rectangular box. The word "FAMILiAR" is written in a bold, sans-serif font. The letters "F", "M", "i", and "A" are dark grey, while "L", "i", and "R" are red. The letter "A" has a red triangle inside it, and the letter "i" has a red dot above it. The letter "R" has a red arc below it.

FAMILiAR

Another application of
composing feature models

(purpose: automated synthesis of feature models)

Problem: multiple „car models“

Vorsprung durch Technik Audi



A1 A3 A4

A5 A8 Q3 Q5

Q7 TT R8

► Enter Audi Code **B**

Model line

- ☐ Audi A1
- ☐ Audi A3
- ☐ Audi A4
- ☐ Audi A5
- ☐ Audi A8

Model line

- ☐ Audi A1
- ☐ Audi A3
- ☐ Audi A4
- ☐ Audi A5
- ☐ Audi A8

Body style

Model

- ☐ Audi Q3
- ☐ Audi Q5
- ☐ Audi Q7
- ☐ Audi TT
- ☐ Audi R8

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

Next ►

Problem: multiple „car models“

Model line	Body style	Model
● Audi A1 ● Audi A3 ● Audi A4 ● Audi A5 ● Audi A8	● Audi Q3 ● Audi Q5 ● Audi Q7 ● Audi TT ● Audi R8	● Saloon ● Avant ● 3 door ● Sportback ● Coupé ● Cabriolet ● Roadster ● Spyder ● allroad quattro ● SUV
		● 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

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

Problem: multiple „car models“

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

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

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

Problem: multiple „car models“

Model line	Body style	Model
● Audi A1 ● Audi A3 ● Audi A4 ● Audi A5 ● Audi A8	● Saloon ● Avant ● 3 door ● Sportback ● Coupé	● Cabriolet ● Roadster ● Spyder ● allroad quattro ● SUV
● Audi Q3 ● Audi Q5 ● Audi Q7 ● Audi TT ● Audi R8		● 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

#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

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

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

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

#1 top-down

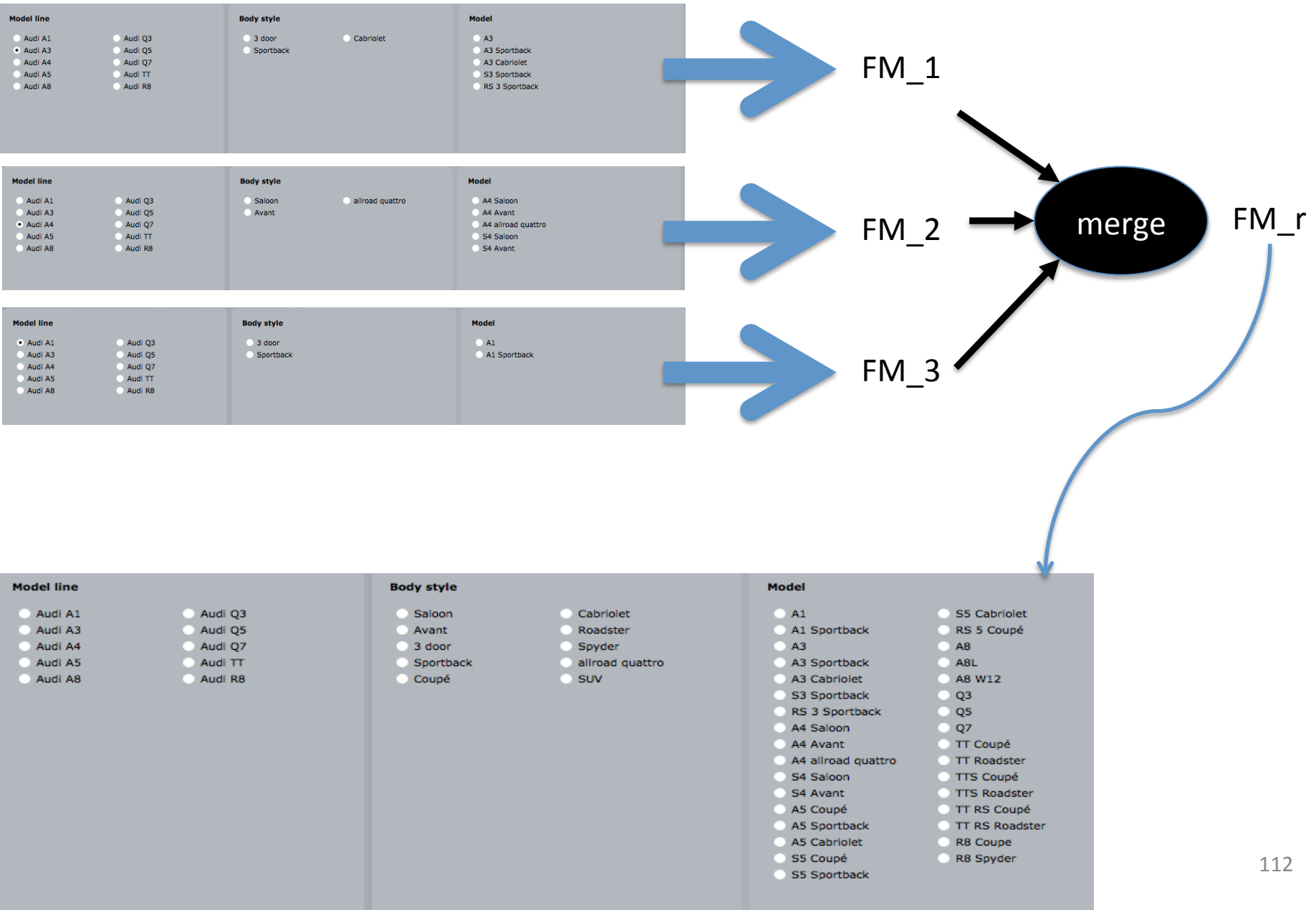
Model line	Body style	Model						
- Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT	- Saloon - Avant 3 door - Sportback - Cabriolet - Roadster - Spyder - allroad quattro - SUV	- A1 - A1 Sportback - A3 - A3 Sportback - A3 Cabriolet - S3 Sportback - RS 3 Sportback - A4 Saloon - 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						
<pre> 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); }, }, } </pre>								
<table> <tr> <th>Model line</th><th>Body style</th><th>Model</th></tr> <tr> <td> - Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8 </td><td> 3 door - Sportback - Cabriolet </td><td> - A3 - A3 Sportback - A3 Cabriolet - S3 Sportback - RS 3 Sportback </td></tr> </table>	Model line	Body style	Model	- Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8	3 door - Sportback - Cabriolet	- A3 - A3 Sportback - A3 Cabriolet - S3 Sportback - RS 3 Sportback		
Model line	Body style	Model						
- Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8	3 door - Sportback - Cabriolet	- A3 - A3 Sportback - A3 Cabriolet - S3 Sportback - RS 3 Sportback						
<table> <tr> <th>Model line</th><th>Body style</th><th>Model</th></tr> <tr> <td> - Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8 </td><td> - Saloon - Avant - allroad quattro </td><td> - A4 Saloon - A4 Avant - A4 allroad quattro - S4 Saloon - S4 Avant </td></tr> </table>	Model line	Body style	Model	- Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8	- Saloon - Avant - allroad quattro	- A4 Saloon - A4 Avant - A4 allroad quattro - S4 Saloon - S4 Avant		
Model line	Body style	Model						
- Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8	- Saloon - Avant - allroad quattro	- A4 Saloon - A4 Avant - A4 allroad quattro - S4 Saloon - S4 Avant						
<table> <tr> <th>Model line</th><th>Body style</th><th>Model</th></tr> <tr> <td> - Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8 </td><td> 3 door - Sportback </td><td> - A1 - A1 Sportback </td></tr> </table>	Model line	Body style	Model	- Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8	3 door - Sportback	- A1 - A1 Sportback		
Model line	Body style	Model						
- Audi A1 - Audi A3 - Audi A4 - Audi A5 - Audi A8 - Audi Q3 - Audi Q5 - Audi Q7 - Audi TT - Audi R8	3 door - Sportback	- A1 - A1 Sportback						

```

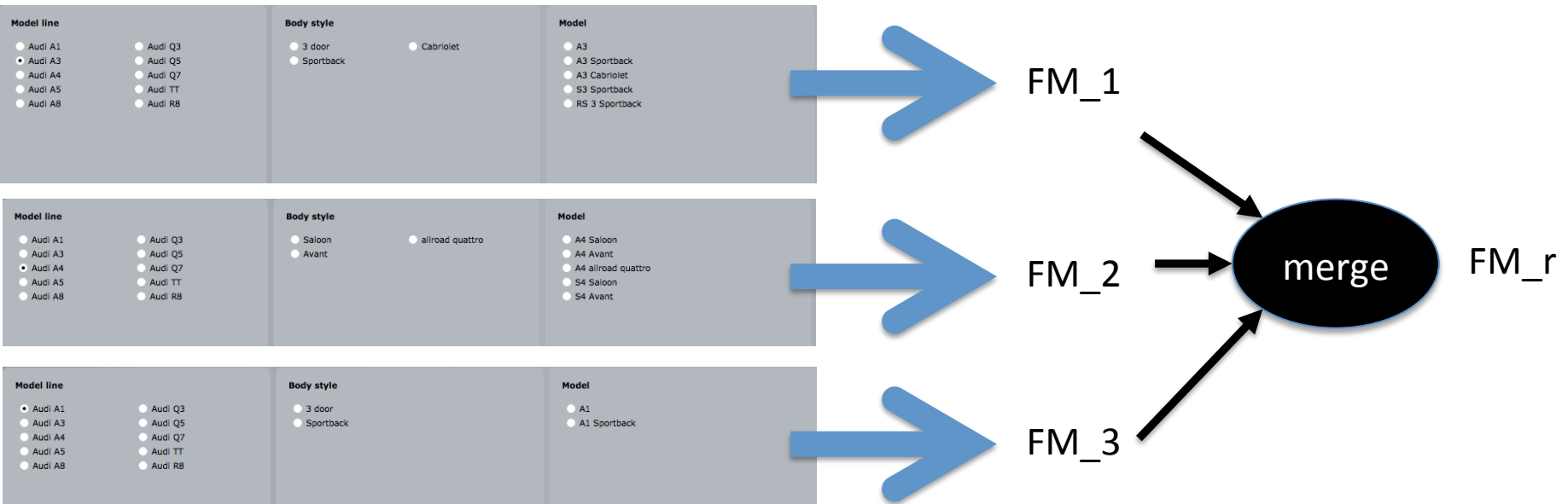
BodyStyle {
  group oneof {
    Saloon
    {
      Saloon -> (A4Saloon || S4Saloon || A6Saloon || A8 || A8L || A8W12);
    },
    Avant
    {
      Avant -> (A4Avant || S4Avant || A6Avant);
    },
    Door3
    {
      Door3 -> (A1 || A3 || S3);
    },
    Sportback
    {
      Sportback -> (A1Sportback || A3Sportback || S3Sportback || RS3Sportback || A5Sportback || S5Sportback || A7Sportback);
    },
    Coupe
    {
      Coupe -> (A5Coupe || S5Coupe || RS5Coupe || TTCoupe || TTSCoupe || TTRS Coupe || R8Coupe);
    },
  },
}

```

#1 bottom-up



#1 bottom-up (FAMILIAR)



```

a fml> fmAudiS = merge union { fm1 fm2 fm3 }
a fmAudiS: (FEATURE_MODEL) Audi: ModellLine BodyStyle ;
F ModellLine: (A1|A4|A3) ;
L BodyStyle: (Saloon|Door3)? (Cabriolet|AllroadQuattro)? (Sportback|Avant)? ;
h
+ (A4 -> !Sportback);
+ (A1 -> !Avant);
M (A1 -> !Saloon);
E (A3 -> !Saloon);
+ (A4 -> !Cabriolet);
+ (A1 -> !AllroadQuattro);
M (A1 -> !Cabriolet);
E (A1 -> !Door3);
+ (A3 -> !Avant);
M (A3 -> !AllroadQuattro);
E

```

9.9.4 (beta)

The logo for FAMiliAR is displayed in a white rectangular box. The word "FAMiliAR" is written in a bold, sans-serif font. The letters "F", "M", and "R" are dark grey. The letter "A" is dark grey with a red triangle inside. The letters "i" and "i" are dark grey with red dots above them. The letter "A" is dark grey with a red arc inside. The letter "R" is dark grey.

FAMiliAR

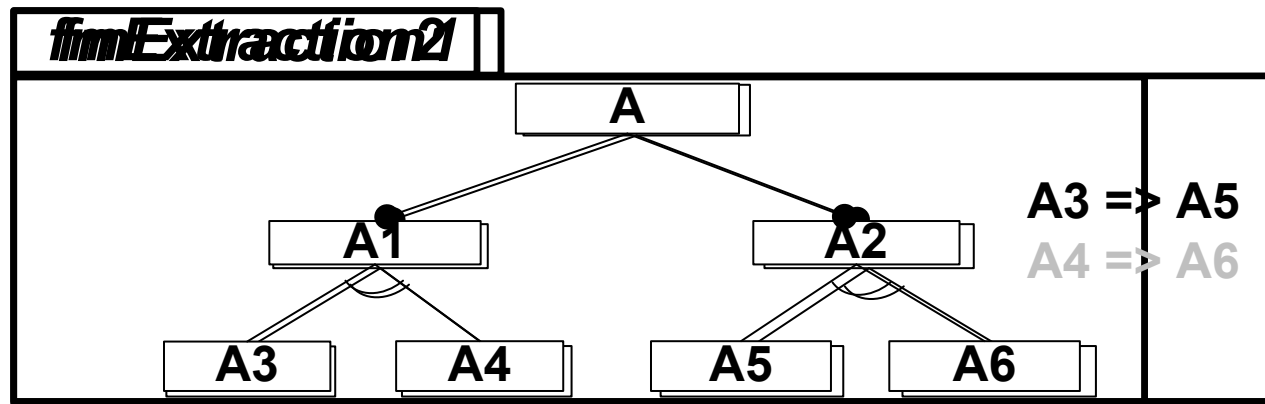
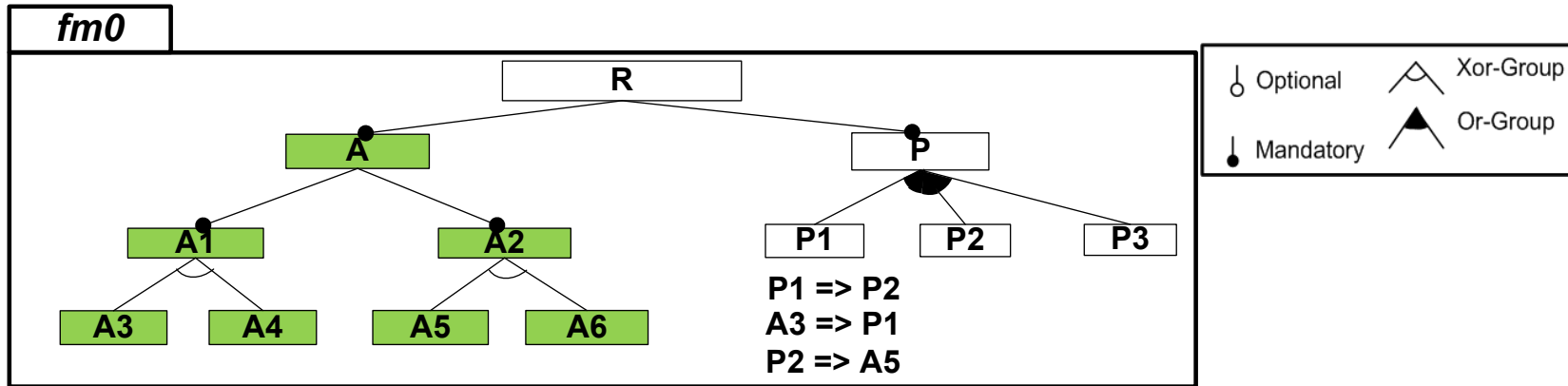
Decomposition support

(and its combination with other operators)

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

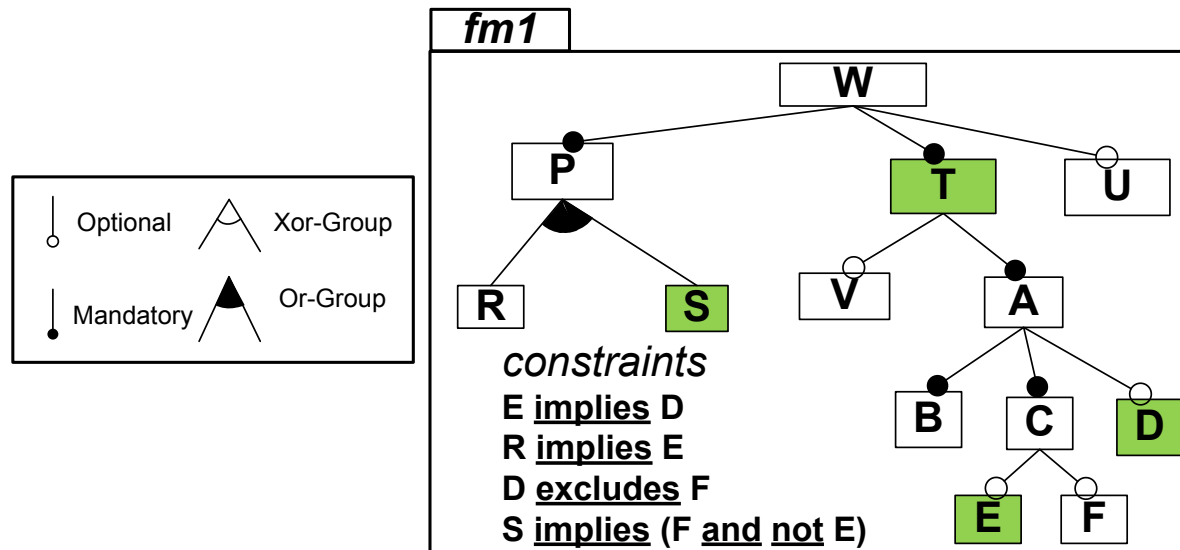


Problem: You can select **A3** without **A5**

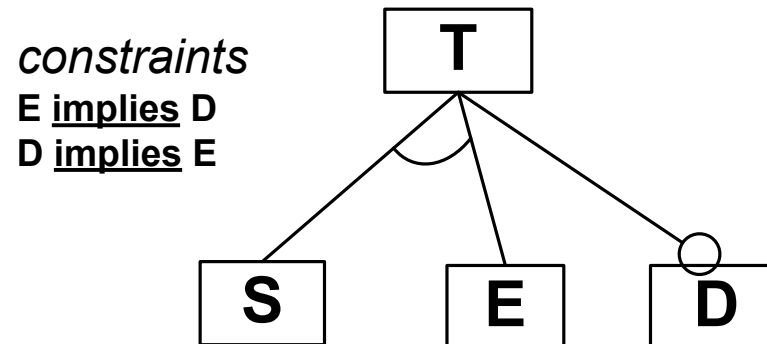
Hierarchy and Configuration matter!

Slicing Operator

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

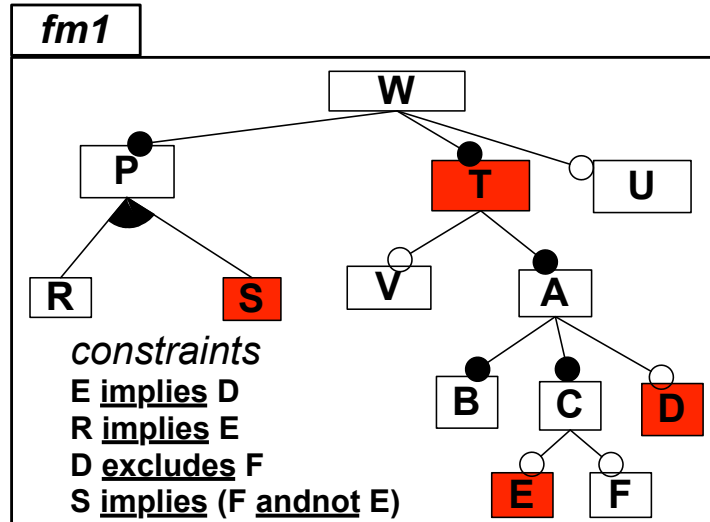
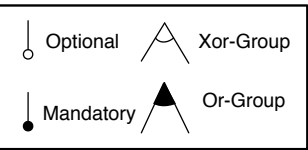


slice : a new feature model, representing a projected set of configurations



Slicing operator: going into details

projected set of configurations

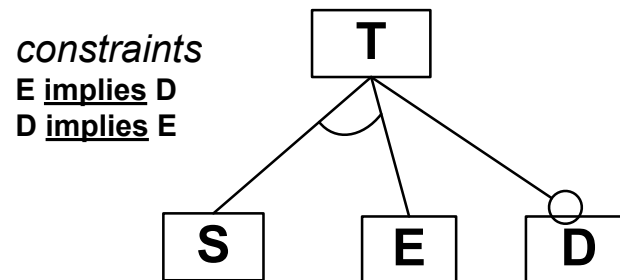
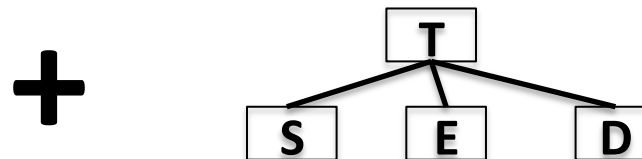
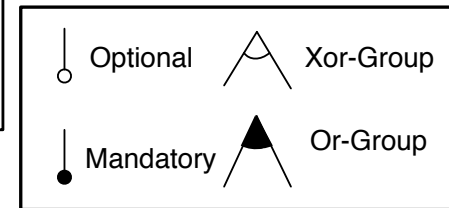
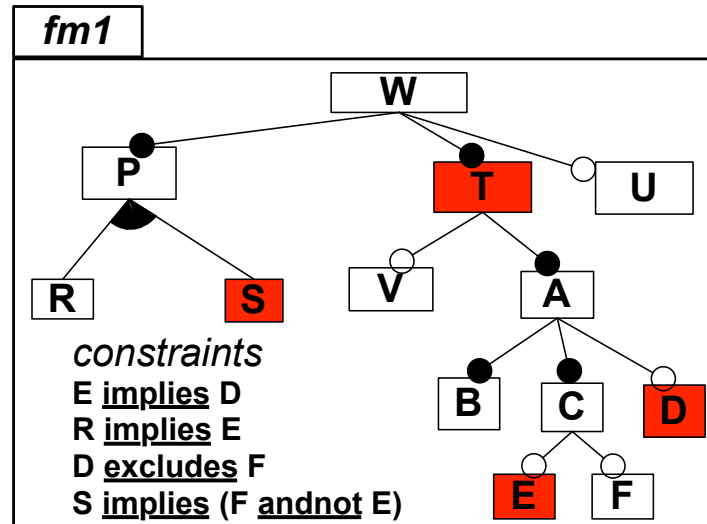
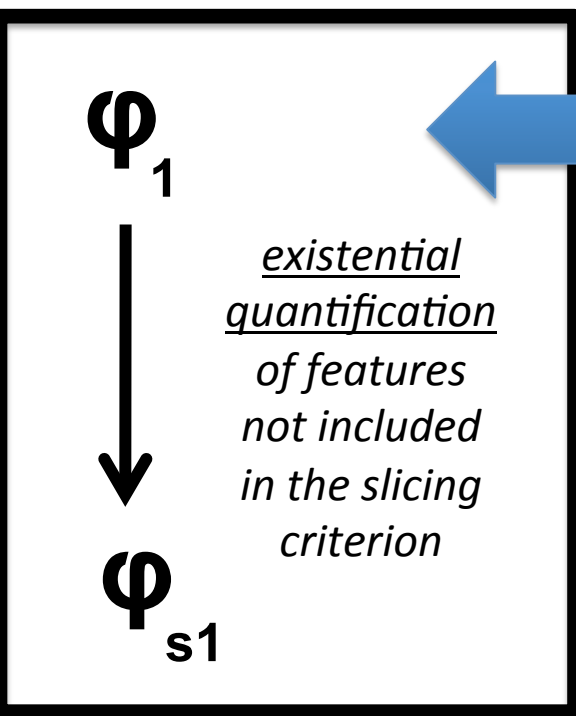


fm1p = {
 {A,B,C,D,E,P,R,T,U,V,W},
 {A,B,C,E,P,S,T,U,V,W},
 {A,B,C,D,E,P,R,T,W},
 {A,B,C,E,P,S,T,W},
 {A,B,C,E,P,S,T,U,V,W},
 {A,B,C,E,P,S,T,W},
 {A,B,C,D,E,P,R,T,W},
 } }

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

Slicing operator with FAMILIAR (1)

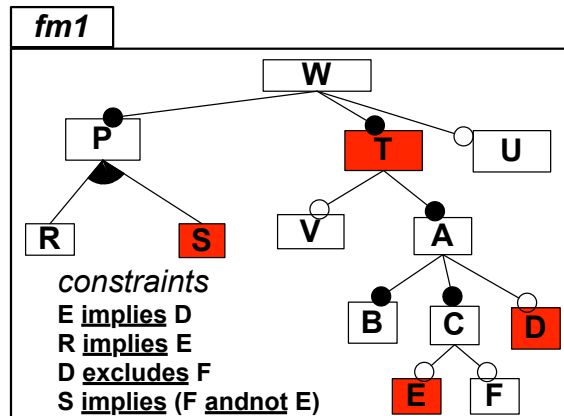
```

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 ; )

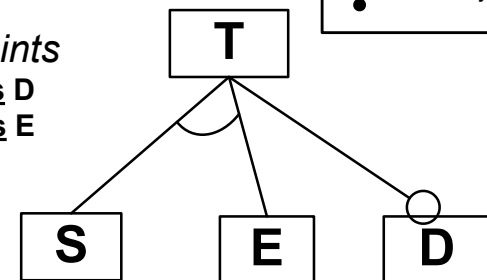
fm2 = slice fm1 including { S T E D }
fm2bis = slice fm1 excluding { W P R V A B C F U }

cmp = compare fm2 fm2bis
assert (cmp eq REFACTORING)

```



constraints
E **implies** D
D **implies** E



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

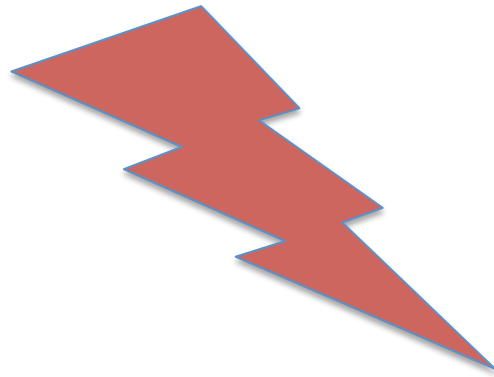
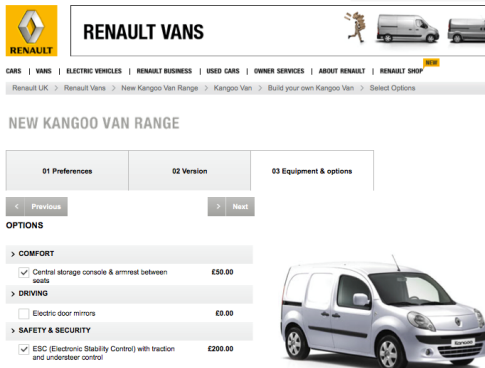
fm4 = slice fm1 including { fm1.E fm1.D fm1.F }

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

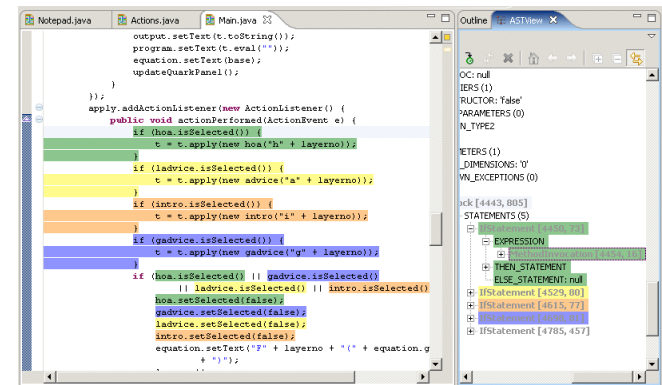
```



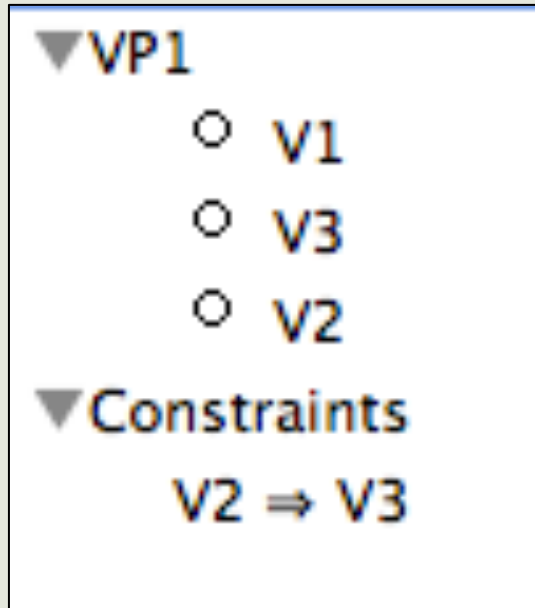
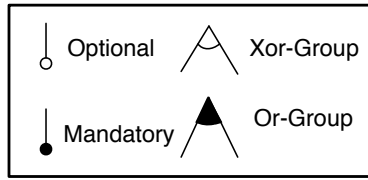
***From marketing,
customers, product
management***



***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*

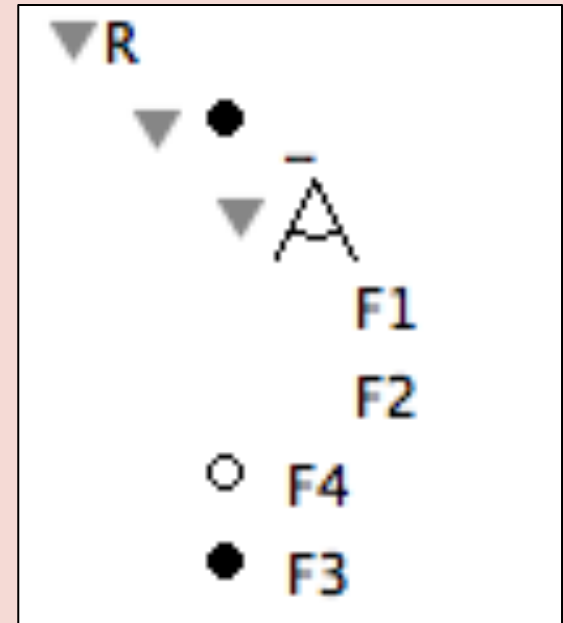
usefulness



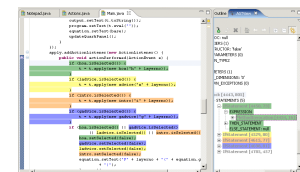
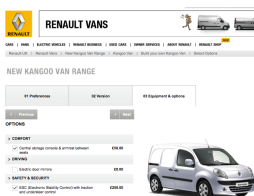
$V1 \Leftrightarrow f1$
 $V2 \Leftrightarrow f2$
 $V3 \Leftrightarrow f3$



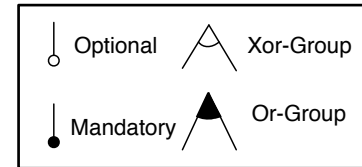
realizability



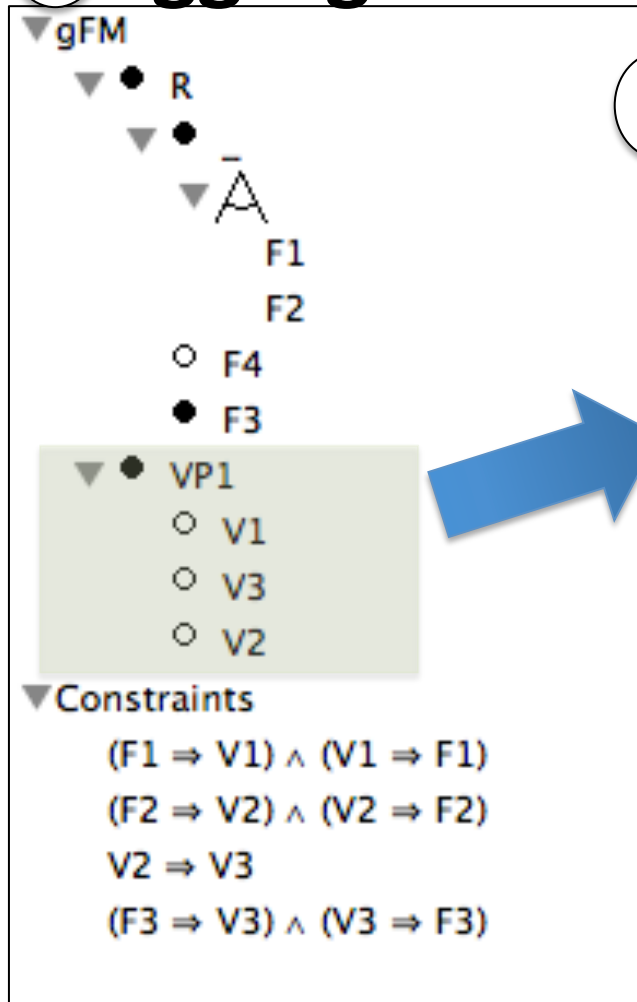
*From existing
software assets*



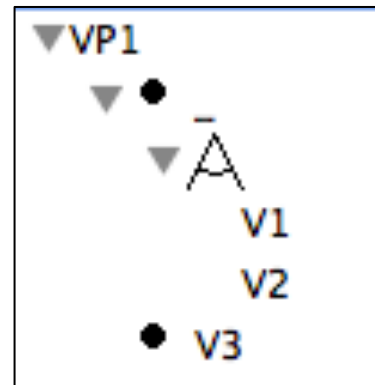
Realizability checking



1 aggregate



2 slice ("realizable part")



3 compare

Φ

$\{\{V1, V3, V2, VP1\},$
 $\{V1, VP1\},$
 $\{V3, VP1\},$
 $\{VP1\}\}$

4 merge diff ("unrealizable products")

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}}
```

Revisiting Merge: Aggregate + Slice

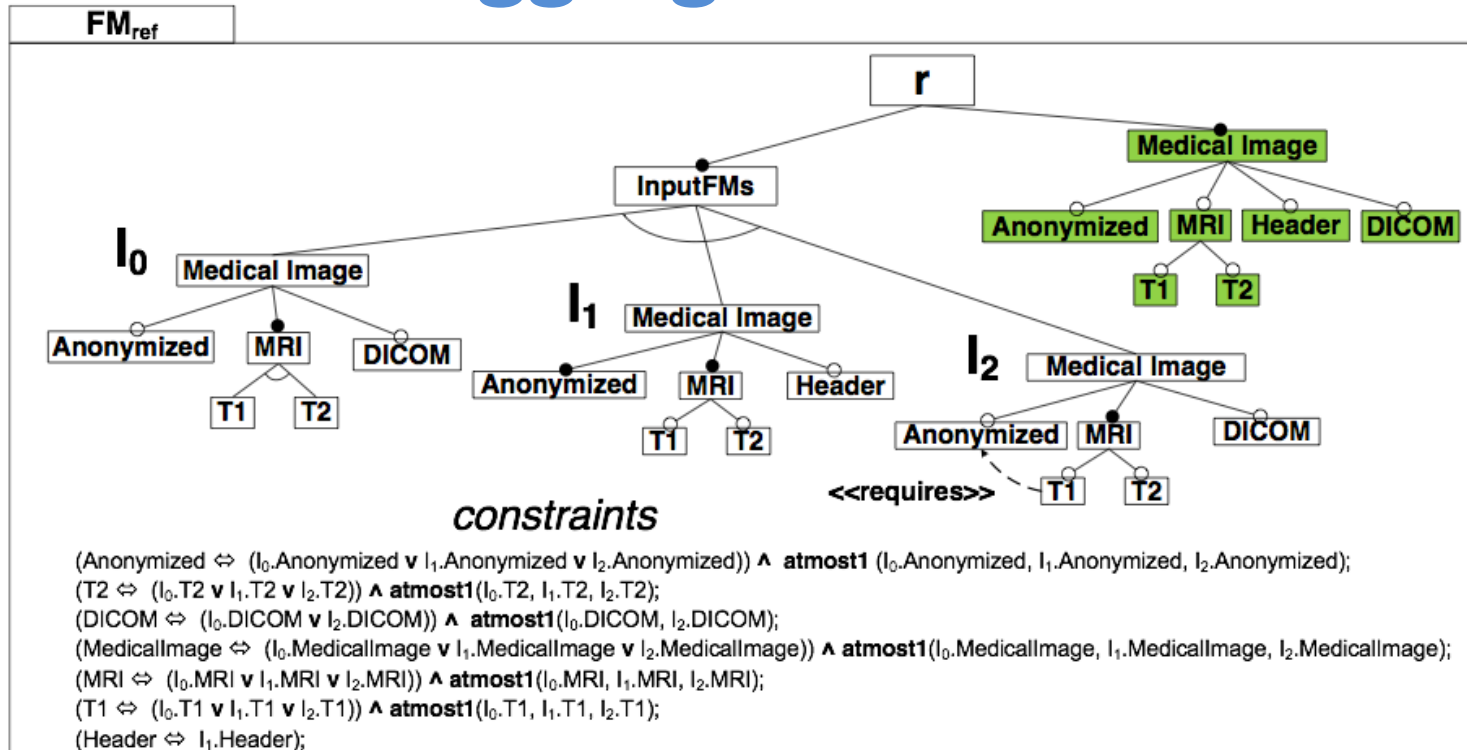
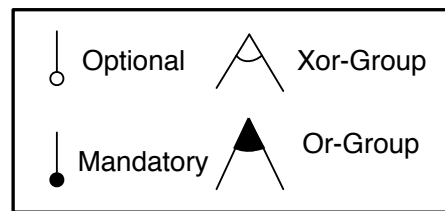
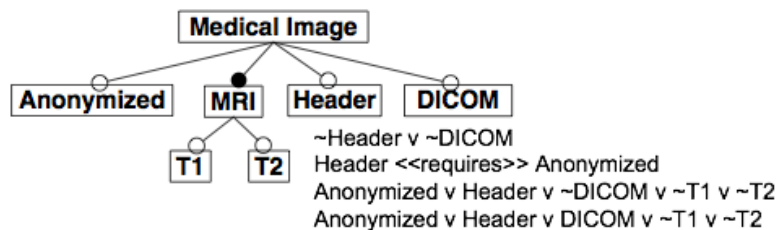


Figure 7.10: Merge of three feature models, I_0 , I_1 and I_2 using the slicing operator

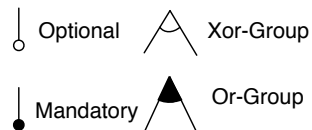


Revisiting Aggregate, Merge and Slice:

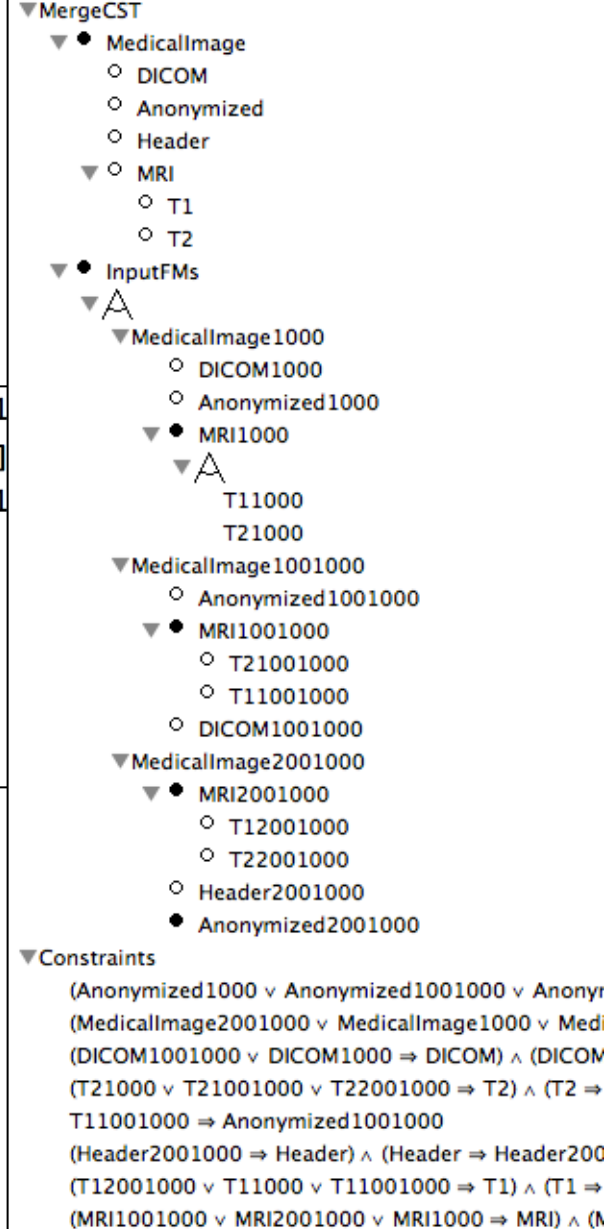
```
fmsupp1 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : (T1
fmsupp2 = FM (MedicalImage : Anonymized MRI [Header] ; MRI : [T1]
fmsupp3 = FM (MedicalImage : [Anonymized] MRI [DICOM] ; MRI : [T1]

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

fmSuppAggSlice = slice fmSuppAgg including fmSupp.*
```



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



 GitHub, Inc. [US] <https://github.com/FAMILIAR-project/familiar-documentation/blob/master/manual/composition.md>

  This repository [Explore](#) [Gist](#) [Blog](#) [Help](#)  FAMILIAR-project   

PUBLIC 

FAMILIAR-project / familiar-documentation

 Watch  Star 1  Fork 2

branch: master [familiar-documentation](#) / [manual](#) / [composition.md](#) 

 FAMILIAR-project 3 months ago Update composition.md

1 contributor

 file | 649 lines (536 sloc) | 29.216 kb

[Edit](#) [Raw](#) [Blame](#) [History](#)

Composing your Compositions of Variability Models

This document presents:

- a comprehensive tutorial on feature model composition, showing the equivalence of various operators and mechanisms offered by the FAMILIAR language
- numerous examples (toy examples or based on the revisit of existing works)

The associated FAMILIAR scripts are located in the [git repository](#) of the FAMILIAR scripts' repository.

There is an [appendix section](#) at the end of the document that demonstrates FAMILIAR sessions when executing the scripts.

Our ultimate goal is to provide solutions that fulfill the various needs of variability model composition.

Authors

- Mathieu Acher (University of Rennes 1, Inria / Irisa, Triskell team)
- Benoit Combemale (University of Rennes 1, Inria / Irisa, Triskell team)
- Philippe Collet (University of Nice Sophia Antipolis)
- Olivier Barais (University of Rennes 1, Inria / Irisa, Triskell team)
- Philippe Lahire (University of Nice Sophia Antipolis)
- Robert B. France (Colorado State University)

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

Quizz (open problem)

Composition of attributed
feature models

Conclusion

- Implementing variability (code clones, preprocessor, generative programming, ..., DSLs)
- Modeling variability
 - Syntax and semantics of feature models; synthesis (from semantics to syntax)
 - Reasoning and operations for feature models
- Reverse engineering variability
 - Code base, textual inputs, tabular data, configuration files, “variants”

