

# Tolerating Inconsistency in Feature Models

Bo Wang\*, Zhenjiang Hu\*\*, Yingfei Xiong\*\*\*,  
Haiyan Zhao\*, Hong Mei\*

\* Peking University (China)

\*\* GRACE Center, National Institute of Informatics (Japan)

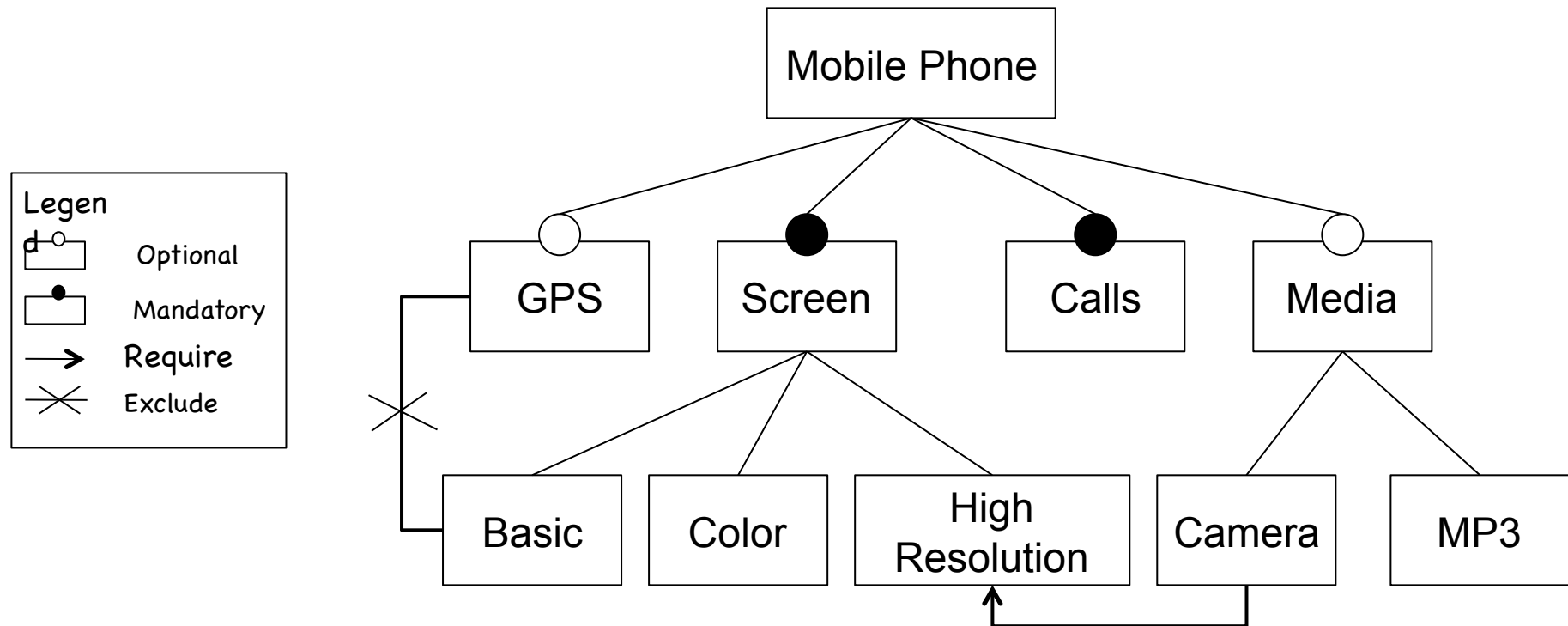
\*\*\* Waterloo University (Canada)

# Feature Models

- A Feature Model = Features + Relationships
  - Capturing the end-user's understanding of the general capabilities of products in a domain.
  - capturing the common and variable requirements of the products in the domain.

## Reuse the requirements

## Example: Mobile Phone

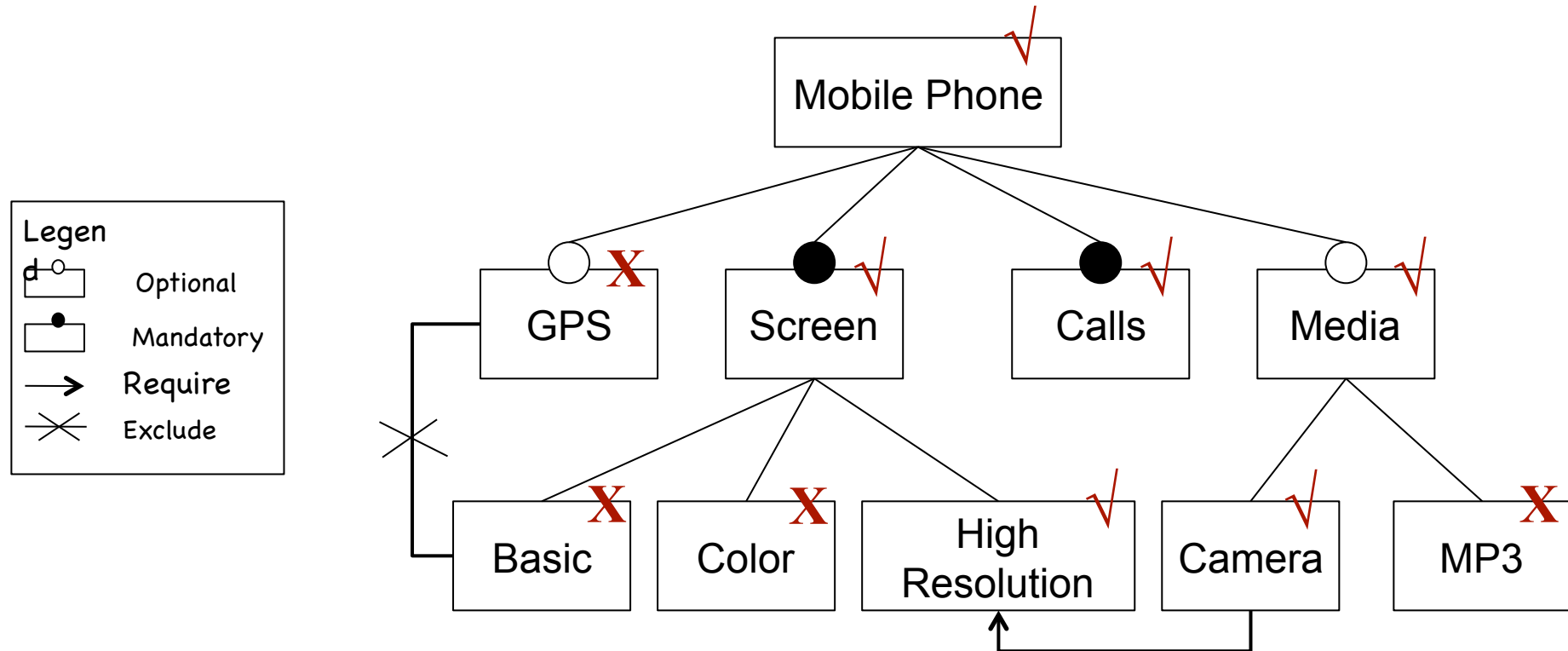


### Complex Constraints:

All-Group({*Screen*}) requires Single-Group({*Basic*, *Color*, *High Resolution*})

All-Group({*Media*}) requires Multi-Group({*Camera*, *MP3*})

## Example: A Product Configuration



### Complex Constraints:

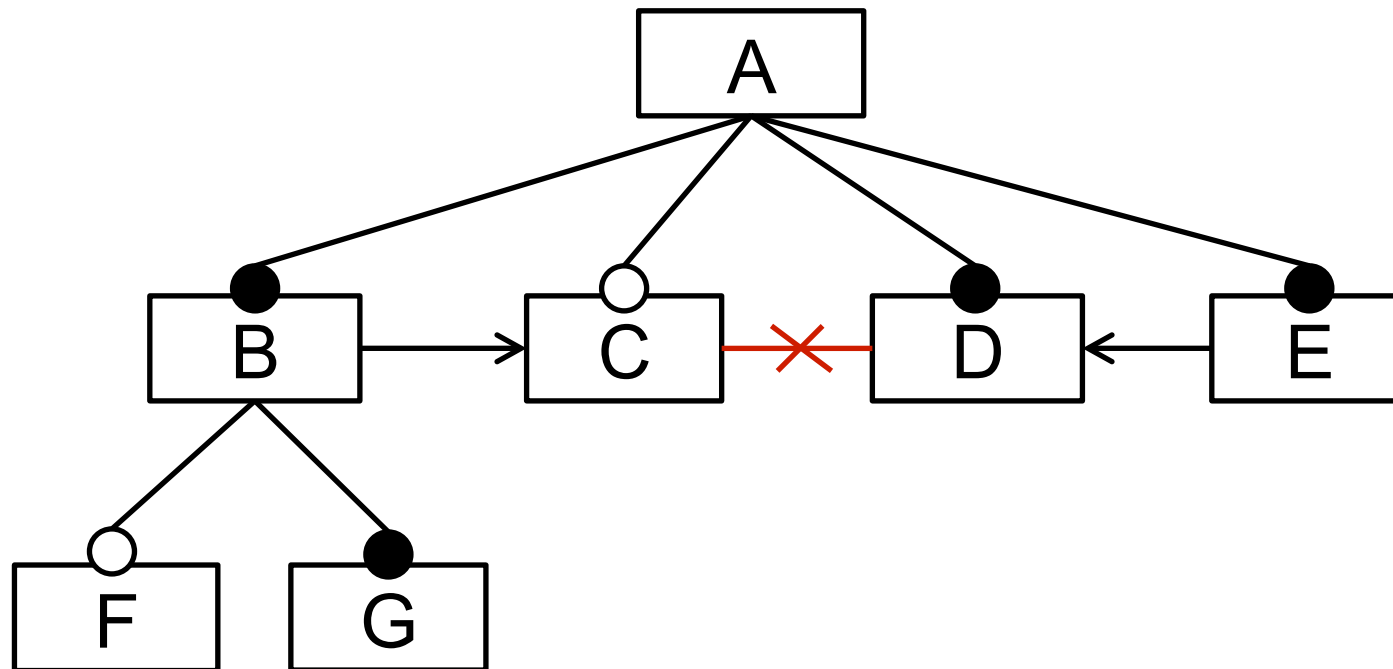
All-Group({*Screen*}) requires Single-Group({*Basic*, *Color*, *High Resolution*})

All-Group({*Media*}) requires Multi-Group({*Camera*, *MP3*})

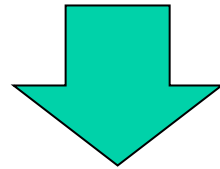
## Inconsistency in Feature Models

- A feature model contains inconsistencies, if the model includes contradictory information.
- Inconsistency leads to the fact that no consistent product configuration can be derived from the feature model.

## Example: An Inconsistent Feature Model



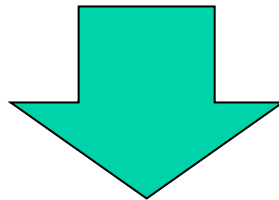
- When constructing feature models, it is difficult to **always** ensure the consistency of feature models.
  - Hard to find a suitable solution to the inconsistency
  - Existing constraints may be useful



Tolerating inconsistencies is important during the construction of feature models.

## Existing Approaches

The usual way of tolerating inconsistencies is to find **the minimal unsatisfied core**, and put it away from the system.



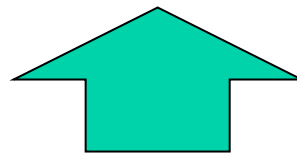
Two limitations:

- Time consuming
- No efficient mechanism to recover the constraints in the minimal unsatisfied core



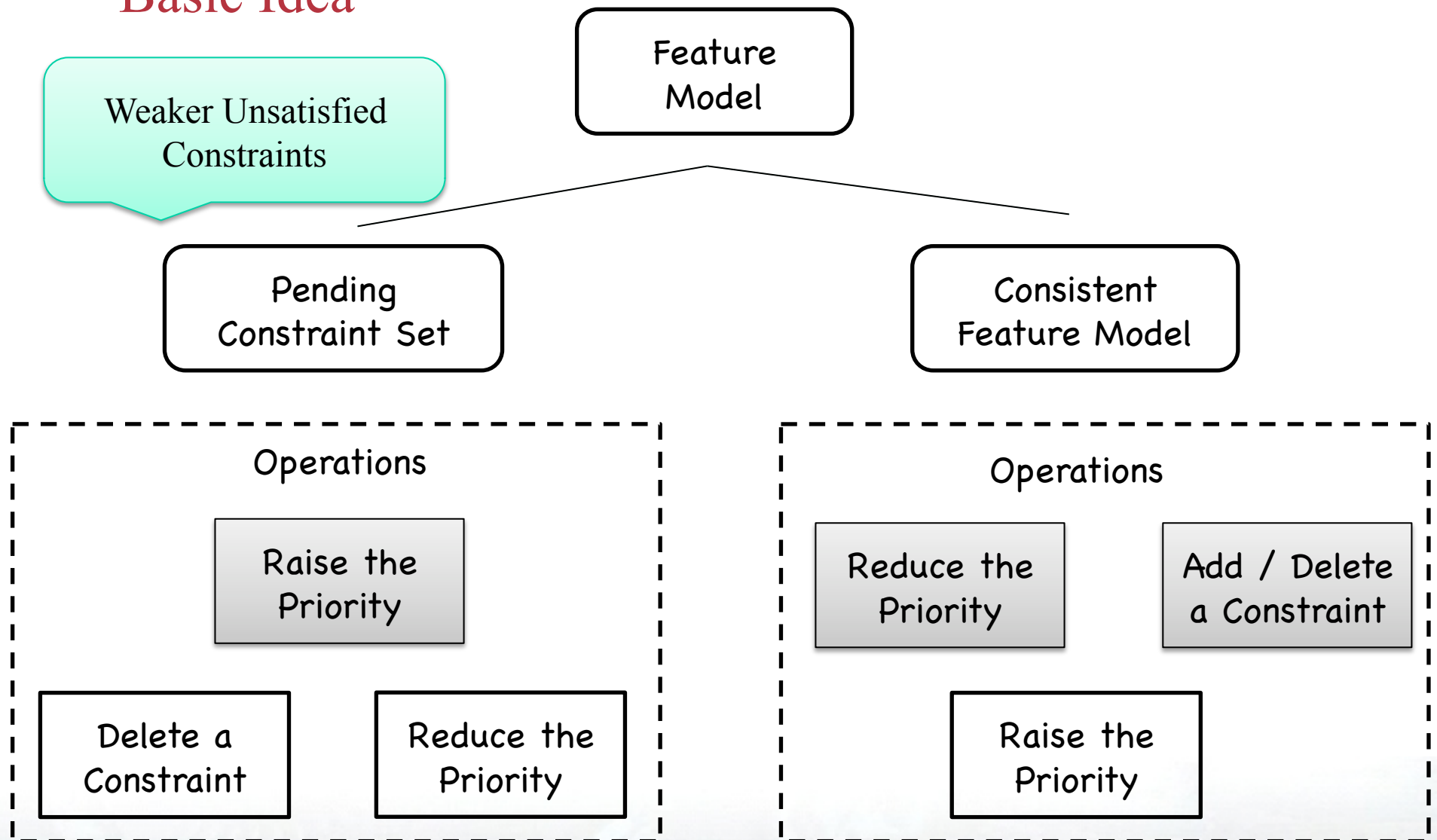
## Our Approach

- We propose a **priority-based approach** to **dynamically** tolerating inconsistencies in feature models.
  - **Priorities are dynamically assigned** to constraints according to the domain engineers' confidence on the constraints
  - Weaker unsatisfied constraints can be **incrementally updated**
  - The feature model is always **consistent** if the weaker unsatisfied constraints are excluded

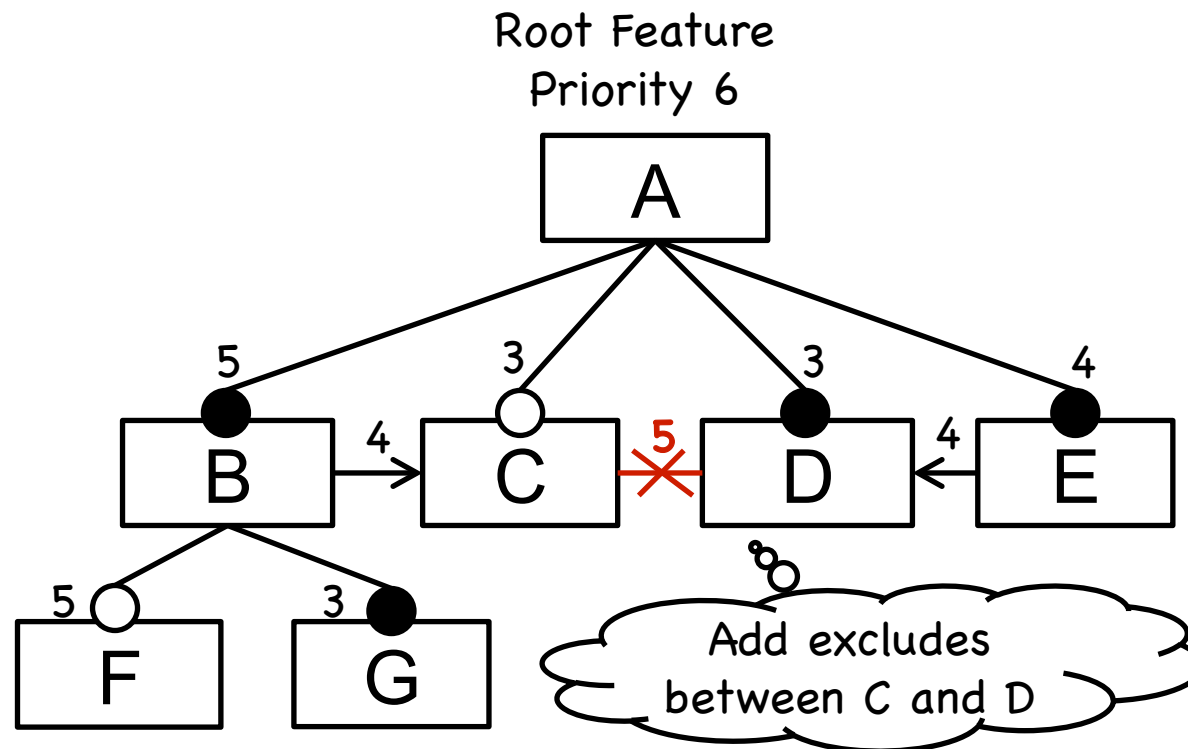


(Extnesion of) Constraint Hierarchy Method (**SkyBlue**)

## Basic Idea



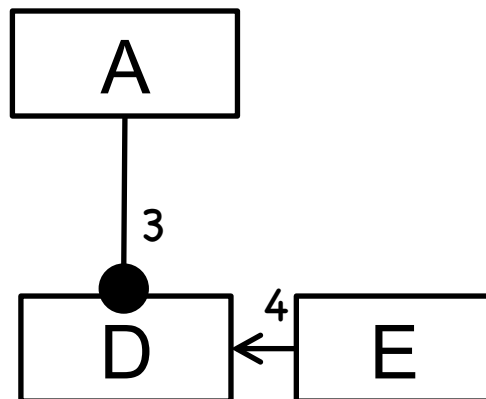
## An Example



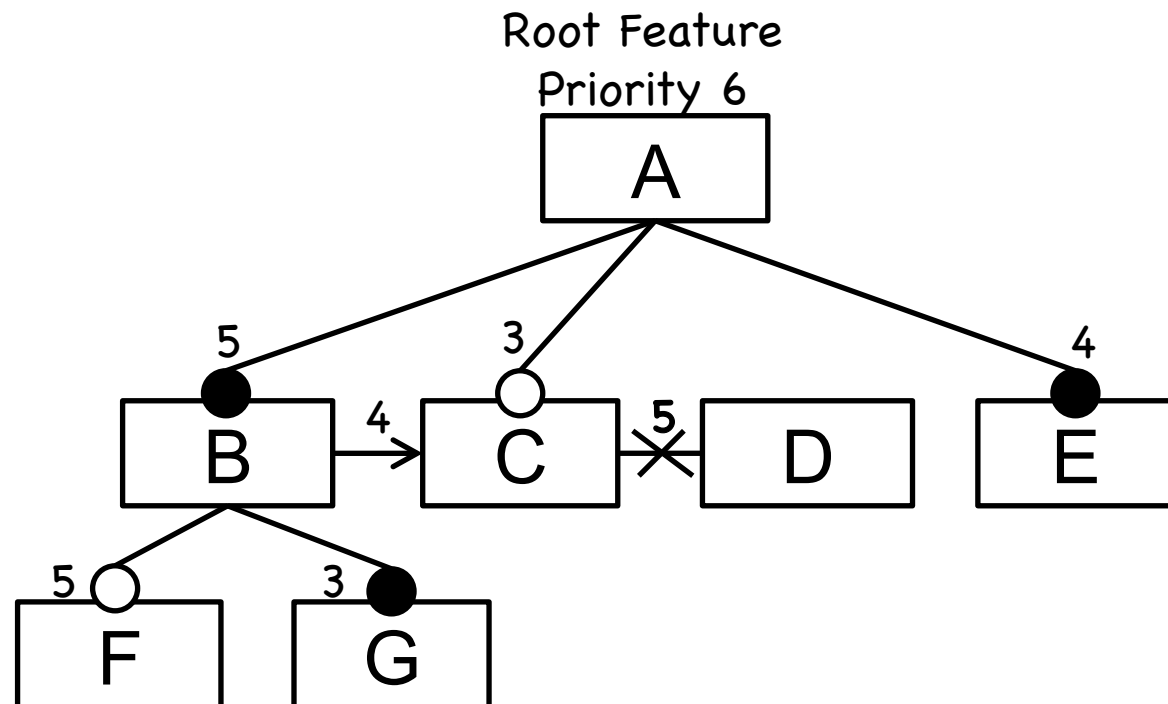
**Composite Constraint:**  
*All-Set(B) c-requires Single(F,G) Priority: 4*

## An Example

PCS



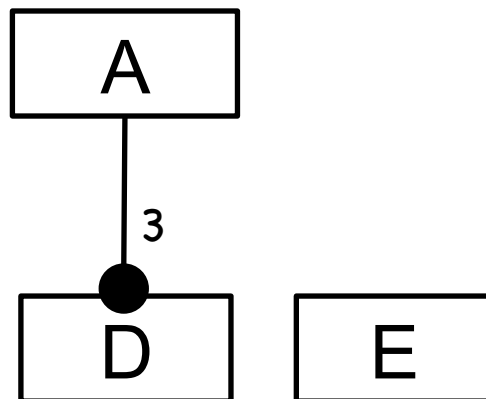
CFM



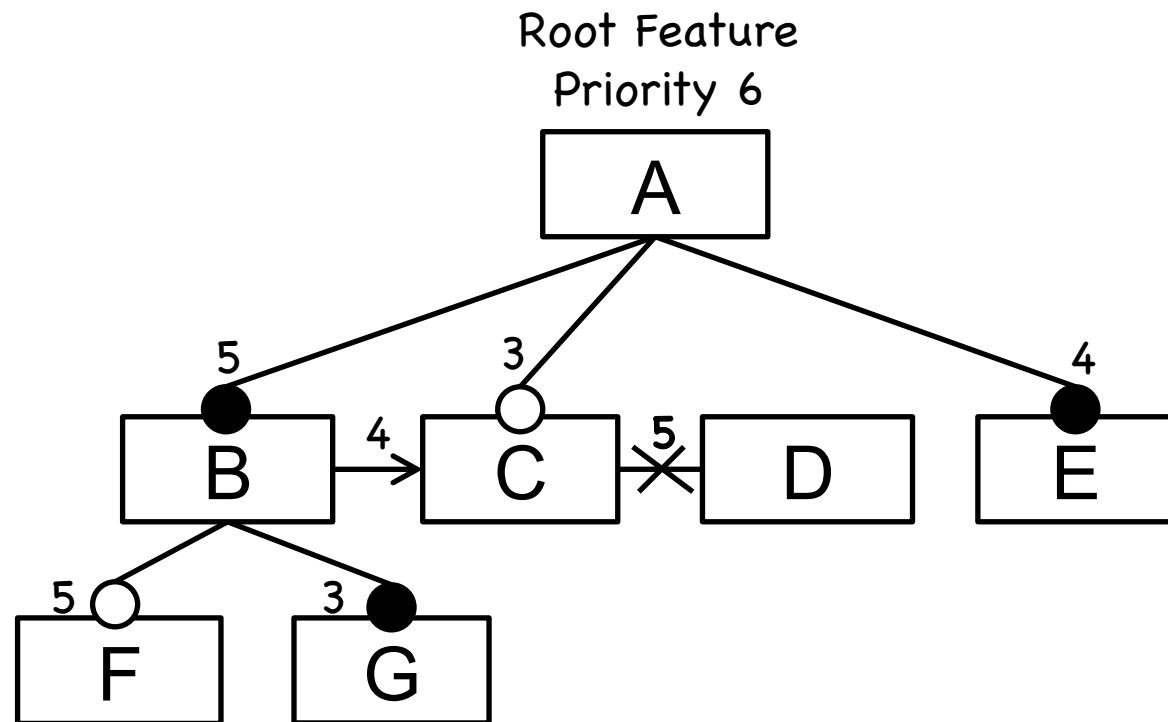
Composite Constraint:  
*All-Set(B) c-requires Single(F,G) Priority: 4*

## An Example

PCS



CFM

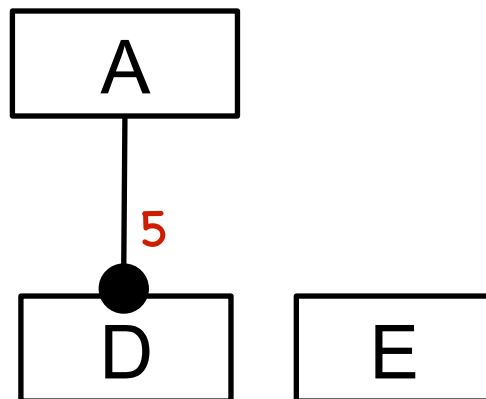


Composite Constraint:

*All-Set(B) c-requires Single(F,G) Priority: 4*

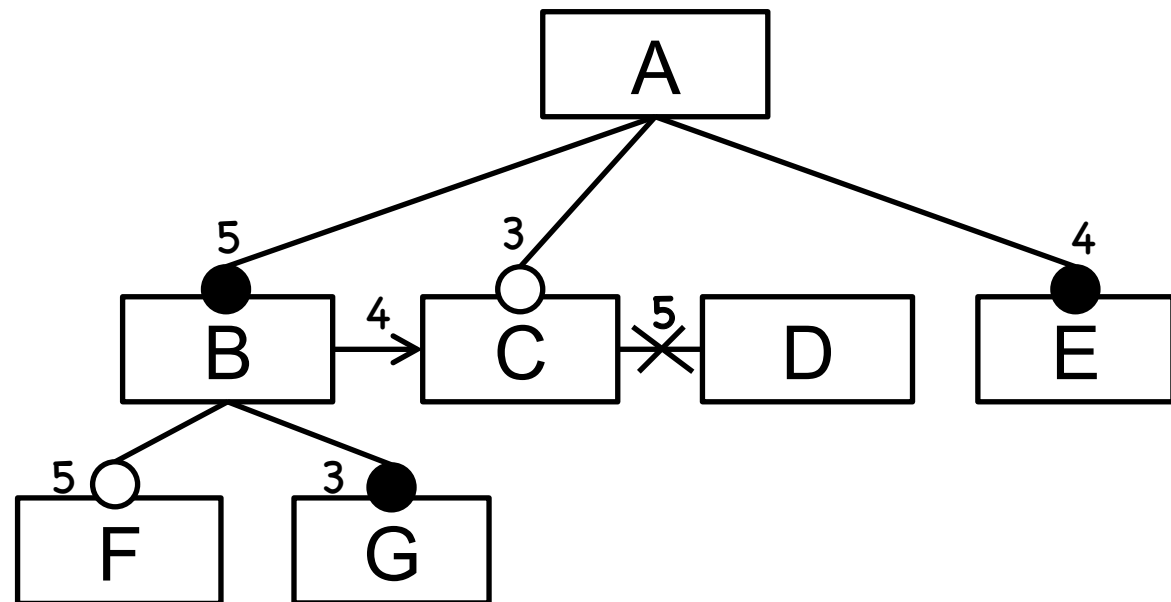
## An Example

PCS



CFM

Root Feature  
Priority 6

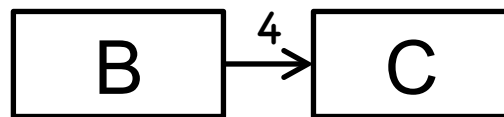


Composite Constraint:

*All-Set(B) c-requires Single(F,G) Priority: 4*

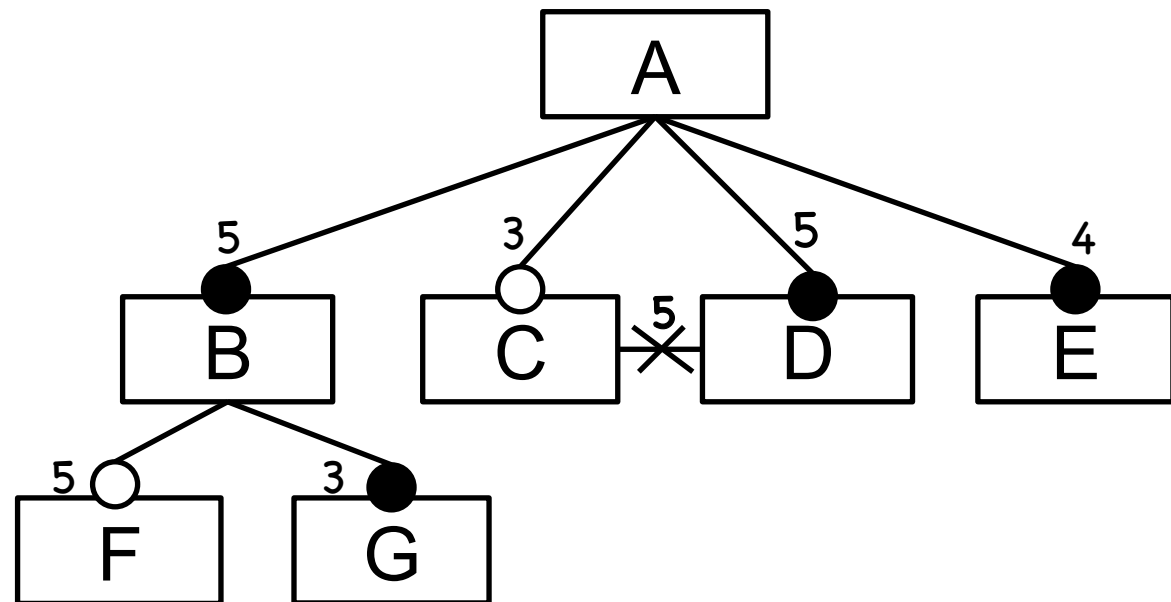
# An Example

PCS



CFM

Root Feature  
Priority 6



Composite Constraint:

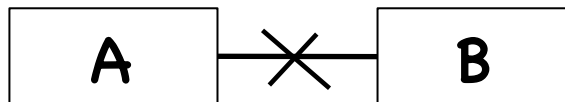
*All-Set(B) c-requires Single(F,G) Priority: 4*

## SkyBlue (1/2)

- A constraint solver widely used in GUI Construction.
  - Very efficient: Incremental Consistency Checking
  - Dynamic: Constraint Hierarchy Method

Each constraint is associated with a set of methods

- Constraint : Feature  $A$  excludes Feature  $B$
- Methods :
  - $A.\text{bindstate} := \text{unbind}$
  - $B.\text{bindstate} := \text{unbind}$



LWI 2010, September 21, 2010  
Feature Model



Constraint Graph

16



Peking  
University



大学共同利用機関法人 情報・システム研究機構  
国立情報学研究所  
National Institute of Informatics



## SkyBlue (2/2)

- SkyBlue checks the consistency by selecting a proper method for each constraint in the constraint graph.

### LGB (Local-graph-better) method graph

no method conflicts

no unenforced constraints that could be enforced by  
revoking one or more weaker constraints

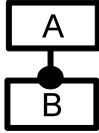
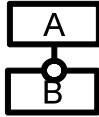
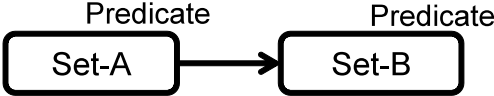
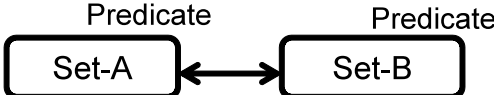
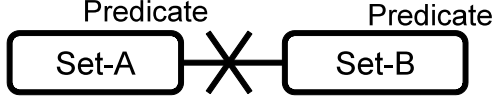
changing the selected methods for enforced constraints

The unenforced constraints in the LGB consist of the PCS.  
The enforced constraints in the LGB consist of the CFM.

## Our Implementation

- How to extend SkyBlue to support tolerating inconsistency in feature models
  - Define the methods for constraints in feature models
  - Revise the algorithm for constructing LGB method graphs

# Define the methods for constraints in feature models

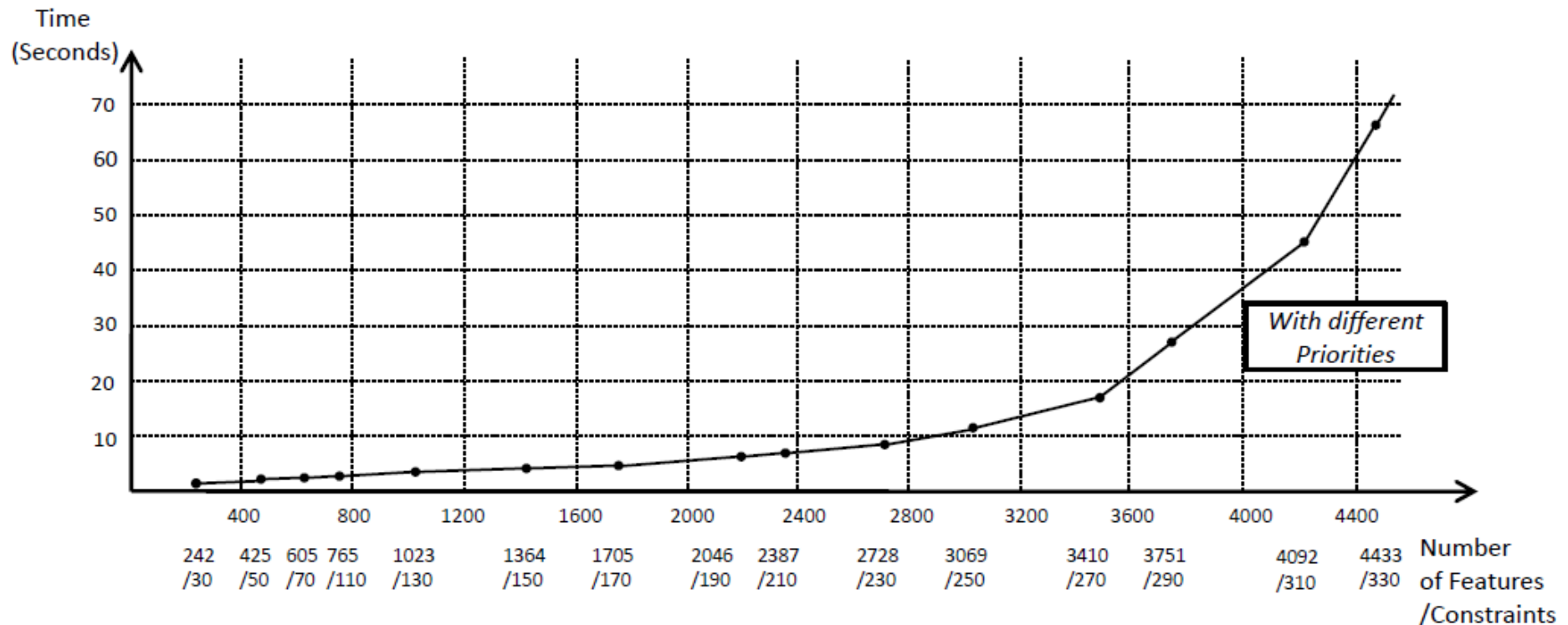
Relationship	Number of Methods	Methods
Mandatory 	2	{Bind(A), Bind(B)} or {Unbind(A), Unbind(B)}
Optional 	2	{Bind(A)} or {Unbind(B)}
Composite-Requires 	2	{Predicate(Set-A) = False} or {Predicate(Set-B) = True}
Composite-M-requires 	2	{Predicate(Set-A) = False, Predicate(Set-B) = False} or {Predicate(Set-A) = True, Predicate(Set-B) = True}
Composite-Excludes 	2	{Predicate(Set-A) = False} or {Predicate(Set-B) = False}

## Define the methods for constraints in feature models

Predicate	Value	Number Of Methods	Methods
All	True	1	{Bind( $A_1$ ), Bind( $A_2$ ) ... Bind( $A_n$ )}
Set-A			{Unbind( $A_1$ ), Unbind( $A_2$ ) ... Unbind( $A_n$ )}
<p><i>All-Set(A,B) composite-excludes Alternative-Set(C,D)</i>  Four methods:  1) {Unbind(A)}; 2){Unbind(B)};  3) {Unbind(C), Unbind(D)};  4) {Bind(C), Bind(D)}</p>			
Set-A { $A_1, A_2 \dots A_n$ }	False	1	{Unbind( $A_1$ ), Unbind( $A_2$ ) ... Unbind( $A_n$ )}

## Experimental Results

- Generate feature models randomly.



Up to 4000 features and 330 explicit constraints.

## Conclusion

- Adopt constraint hierarchy method to divide feature models into the PCS and CFM.
- Provide six priority-based operations to construct feature models.
- Extend SkyBlue to support tolerating inconsistency in feature models.
  - Introduction of compound features
  - A feature can be bounded multiple times

# Thank you!

## Q&A

LWI 2010, September 21, 2010



Peking  
University



大学共同利用機関法人 情報・システム研究機構  
国立情報学研究所  
National Institute of Informatics