# Reasoning about Feature Models in Higher-Order Logic

**Mikoláš Janota**     Joseph Kiniry

Systems Research Group,
University College Dublin, Ireland

SPLC '07

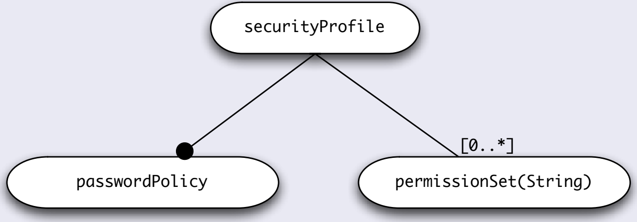
Mobius     lero

IST-15905

# Feature Oriented Domain Analysis

## Feature Models

- capture variability and commonality of a product line
- features represent the building blocks



# Why Formalize?

## Disambiguation

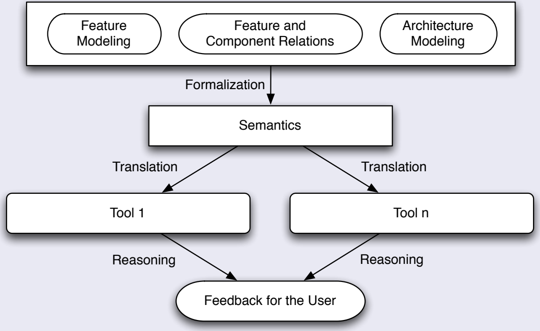
- informal explanation of the meaning might be ambiguous
- for example, absolute vs. relative meaning of *mandatory*

# Why Formalize?

## Disambiguation

- informal explanation of the meaning might be ambiguous
- for example, absolute vs. relative meaning of *mandatory*

## Reasoning about Feature Models

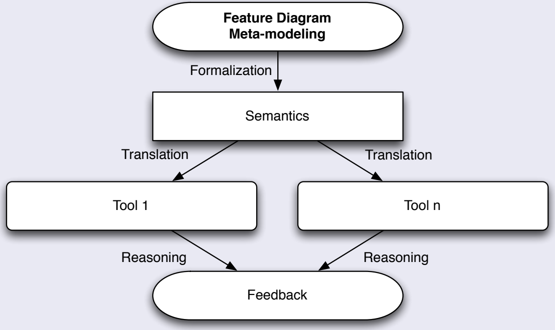


# Why Formalize?

## Disambiguation

- informal explanation of the meaning might be ambiguous
- for example, absolute vs. relative meaning of *mandatory*

## Reasoning at the Meta Level



# Mechanization of the Formalization

## PVS

- proof assistant widely used in computer science
- typed higher-order logic language

## Pros and Cons

🟩 reason about feature-models that have infinite number of configurations (e.g., feature cloning, attributes)

🟩 express and reason about constraints expressible in HOL

🟩 high level of trustworthiness of the formalization
as proofs are checked by a computer

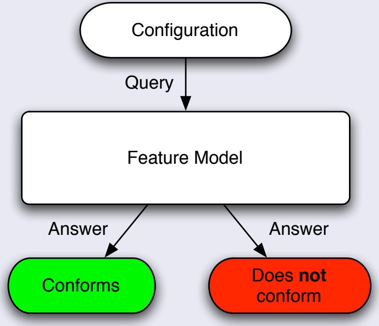
🟥 requires expertise in using a HOL proof-assistant

🟥 some tasks might be tedious

# Concepts

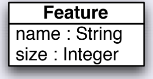
## Feature Models as Oracles

- the set of selected features and values of their attributes constitute a *configuration*
- a configuration either does or does not *conform* to the model



# Features and Configurations

## Features and Attributes



Feature

name : String

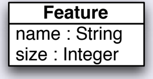
size : Integer

$$\text{Feature} \rightarrow \mathcal{P}(\text{AttributeIdentifier})$$

$$\text{AttributeIdentifier} \rightarrow \mathrm{Type}$$

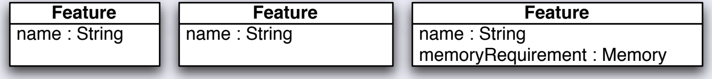
# Features and Configurations

## Features and Attributes



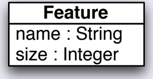
$$\text{Feature} \rightarrow \mathcal{P}(\text{AttributeIdentifier})$$
$$\text{AttributeIdentifier} \rightarrow \mathrm{Type}$$

## Feature Configurations



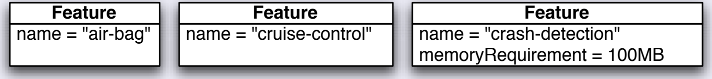
# Features and Configurations

## Features and Attributes



$$\text{Feature} \rightarrow \mathcal{P}(\text{AttributeIdentifier})$$
$$\text{AttributeIdentifier} \rightarrow \mathrm{Type}$$

## Feature Configurations

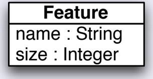


- *value assignment function* assigns values to attributes

$$\mathbb{A} \equiv \text{Feature} \rightarrow (\text{AttributeIdentifier} \rightarrow \text{AttributeValues})$$

# Features and Configurations

## Features and Attributes



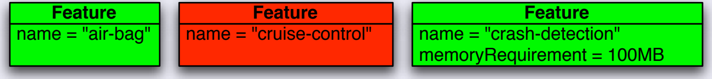
**Feature**

name : String

size : Integer

$$\text{Feature} \rightarrow \mathcal{P}(\text{AttributeIdentifier})$$
$$\text{AttributeIdentifier} \rightarrow \mathrm{Type}$$

## Feature Configurations



**Feature**

name = "air-bag"

**Feature**

name = "cruise-control"

**Feature**

name = "crash-detection"

memoryRequirement = 100MB

- *value assignment function* assigns values to attributes

$$\mathbb{A} \equiv \text{Feature} \rightarrow (\text{AttributeIdentifier} \rightarrow \text{AttributeValues})$$

- *selection function* determines the selected features

$$\textbf{select} \equiv \text{Feature} \rightarrow \mathrm{Boolean}$$

# Feature Models as Restriction Functions

## Feature Models as Restriction Functions

a *restriction function* determines whether the given feature selection and attributes' values conform to the model

$$\textbf{restr} \equiv \textbf{select} \times \mathbb{A} \to \mathrm{Boolean}$$

# Feature Models as Restriction Functions

## Feature Models as Restriction Functions

a *restriction function* determines whether the given feature selection and attributes' values conform to the model

$$\textbf{restr} \equiv \textbf{select} \times \mathbb{A} \rightarrow \mathrm{Boolean}$$

## Examples of Restriction Functions

- $f_1$ requires $f_2$:

$$r_1(s : \textbf{select}, a : \mathbb{A}) \equiv s(f_1) \Rightarrow s(f_2)$$

- $f_2$ requires $f_3$ with a specific version:

$$r_2(s : \textbf{select}, a : \mathbb{A}) \equiv s(f_2) \Rightarrow (s(f_3) \wedge a(f_3)(\mathrm{version}) = 7)$$

- restriction functions can be combined:

$$r_3(s : \textbf{select}, a : \mathbb{A}) \equiv r_1(s, a) \wedge r_2(s, a)$$

# Feature Models as Restriction Functions

## Feature Models as Restriction Functions

a *restriction function* determines whether the given feature selection and attributes' values conform to the model

$$\textbf{restr} \equiv \textbf{select} \times \mathbb{A} \rightarrow \mathrm{Boolean}$$

## More Examples in PVS Notation

- a restriction function that corresponds to a requires relation:

```
require(requiree, required: FEATURE) : RESTRICTION =
  LAMBDA (select: SELECT, da: DOMAIN_ASSIGNMENT):
    (select(requiree) IMPLIES select(required))
```

# Feature Models as Restriction Functions

## Feature Models as Restriction Functions

a *restriction function* determines whether the given feature selection and attributes' values conform to the model

$$\textbf{restr} \equiv \textbf{select} \times \mathbb{A} \to \mathrm{Boolean}$$

## More Examples in PVS Notation

- a restriction function that corresponds to a requires relation:

```
require(requiree, required: FEATURE) : RESTRICTION =
  LAMBDA (select: SELECT, da: DOMAIN_ASSIGNMENT):
    (select(requiree) IMPLIES select(required))
```

- combine two given restriction functions:

```
intersect(r1, r2: RESTRICTION) : RESTRICTION =
  LAMBDA (select: SELECT, da: DOMAIN_ASSIGNMENT):
    r1(select, da) AND r2(select, da)
```

# Meta-Level Property Example

## Specialization of a Feature Model via Restriction Functions

$$\begin{gathered} \mathit{specialization?}(\mathit{restr}_1, \mathit{restr}_2 : \mathbf{restr}) \equiv \\ \forall s : \mathbf{select}; a : \mathbb{A} \bullet \mathit{restr}_1(s, a) \Rightarrow \mathit{restr}_2(s, a) \end{gathered}$$

# Meta-Level Property Example

## Specialization of a Feature Model via Restriction Functions

$$\begin{aligned} &\mathit{specialization?}(\mathit{restr}_1, \mathit{restr}_2 : \mathbf{restr}) \equiv \\ &\quad \forall s : \mathbf{select}; a : \mathbb{A} \bullet \mathit{restr}_1(s, a) \Rightarrow \mathit{restr}_2(s, a) \end{aligned}$$

## Higher-Order Functions on Restriction Functions

assignment to an attribute value:

$$\begin{aligned} &\mathit{assign\text{-}value}(r : \mathbf{restr}) \equiv \\ &\quad \lambda s : \mathbf{select}, a : \mathbb{A} \bullet r(s, a) \wedge (a(f_1)(\mathrm{version}) = 3) \end{aligned}$$

# Meta-Level Property Example

## Specialization of a Feature Model via Restriction Functions

$$\begin{aligned} &\mathit{specialization?}(\mathit{restr}_1, \mathit{restr}_2 : \mathbf{restr}) \equiv \\ &\quad \forall s : \mathbf{select};\, a : \mathbb{A} \bullet \mathit{restr}_1(s, a) \Rightarrow \mathit{restr}_2(s, a) \end{aligned}$$

## Higher-Order Functions on Restriction Functions

assignment to an attribute value:

$$\begin{aligned} &\mathit{assign\text{-}value}(r : \mathbf{restr}) \equiv \\ &\quad \lambda s : \mathbf{select}, a : \mathbb{A} \bullet r(s, a) \wedge (a(f_1)(\mathrm{version}) = 3) \end{aligned}$$

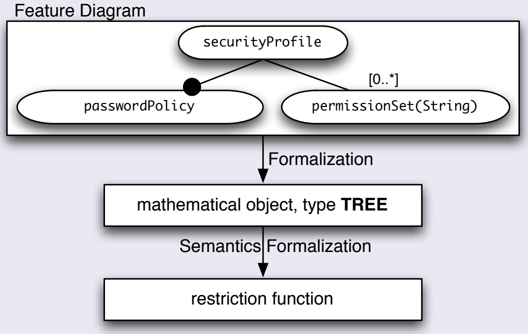
## Reasoning

the function *assign-value* returns a specialization:

$$\forall r \bullet \mathit{specialized?}(\mathit{assign\text{-}value}(r), r)$$

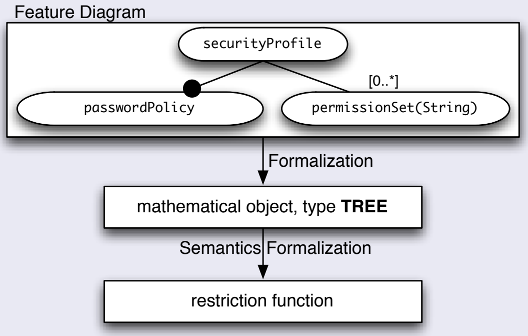
# From Feature Diagrams to Restriction Functions

## Schematically



# From Feature Diagrams to Restriction Functions

## Schematically



## A Function From Diagram to Restriction Function

$$\text{getRestriction} : \textbf{TREE} \rightarrow (\textbf{select} \times \mathbb{A} \rightarrow \mathrm{Boolean})$$

# Baking Restriction Functions

## Modeling Gradual Specialization of Restriction Function

- obtain a restriction function, e.g., from a feature diagram

$$r_0 \equiv \text{getRestriction}(\mathit{tree})$$

# Baking Restriction Functions

## Modeling Gradual Specialization of Restriction Function

- obtain a restriction function, e.g., from a feature diagram

$$r_0 \equiv \mathrm{getRestriction}(\mathit{tree})$$

- compose the functions defining each specialization:

$$\begin{aligned} r_1 &\equiv \mathit{spec}_1(r_0) \\ r_2 &\equiv \mathit{spec}_2(r_1) \\ &\dots \\ r_n &\equiv \mathit{spec}_n(r_{n-1}) \end{aligned}$$

# Baking Restriction Functions

## Modeling Gradual Specialization of Restriction Function

- obtain a restriction function, e.g., from a feature diagram

$$r_0 \equiv \mathrm{getRestriction}(\mathit{tree})$$

- compose the functions defining each specialization:

$$\begin{aligned} r_1 &\equiv \mathit{spec}_1(r_0) \\ r_2 &\equiv \mathit{spec}_2(r_1) \\ &\ldots \\ r_n &\equiv \mathit{spec}_n(r_{n-1}) \end{aligned}$$

## Bringing Specializations Together

$$r_n = \mathit{spec}_n(\ldots(\mathit{spec}_1(\mathrm{getRestriction}(\mathit{tree})))\ldots)$$

# Summary

## Feature Models as Oracles

- the oracle is an important characteristic of the feature model
- enables unified mathematical approach
  - meta-model level, e.g., what is specialization
  - model level, e.g., record constraints in mathematical notation
- oracles are compositional