Indian Journal of Science and Technology, *Vol 8(17), DOI: 10.17485/ijst/2015/v8i17/55613, August 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

On Formalization of Extended Feature Model using Promotion Technique in Z

Shohreh Ajoudanian[1]* and Seyed-Hassan Mirian Hosseinabadi[2]

[1]Department of Computer Engineering, Islamic Azad University Science and Research Branch, Tehran, Iran; shajoudanian@pco.iaun.ac.ir, shajoudanian@yahoo.com
[2]Department of Computer Science, Sharif University of Technology, Tehran, Iran; hmirian@sharif.edu

Abstract

Objectives: To investigate automatic analysis of extended feature model with cardinality. **Methods/Analysis**: Presenting a formal approach to the specification of Ext-FM-WC. In this paper, we propose a promotion technique in Z language for formalizing Software Product Line Systems (SPLS). **Findings:** The main reason for choosing promotion technique for formalizing SPLS is that applying analysis operations that have promoting manner in large scale feature models is easier. **Conclusion/Application**: The structure of constructing Ext-FM-WC using promotion technique can has a lot of benefits on analysis operations that have naturally promoting structure.

Keywords: Automatic Analysis of SPLS, Extended Feature Model with Cardinality, Multiple Multi-level Promotions in Z

1. Introduction

Software product line engineering is about producing a set of related products that share more commonalities than variability. Feature models are widely used for variability and commonality management in software product lines[1]. The first formulation of a feature model language is due to Kang et al.[2]. A feature model captures software product line information about common and variant features of the software product line at different levels of abstraction. A feature model is represented as a hierarchically arranged set of features with different relationships among them. It models all possible products of a software product line in a given context. Unlike traditional information models, feature models not only represent a single product but also a family of them in the same model[1]. Extended Feature Models With Cardinality (Ext-FM-WC) are a kind of feature models with additional information about their features. Feature models only deal with characteristics related to the functionality offered by an SPL but Ext-FM-WC deal with extra functional features[3] (Ext-FM-WC are models with features, attributes, attribute domain and the relationships between different features and attributes of the different features).

Analysis operations on an Ext-FM-WC take as an input a model and extract information from models and return them as outputs. There are many works in this kind of operations[1,3–5] but to our knowledge, the best collection of analysis operations on feature models is the one presented in[1]. Analysis operations of Ext-FM-WC are still largely informal. The focus of this paper is on constructing Ext-FM-WC using promotion technique that has a lot of benefits on analysis operations that have naturally promoting structure. For example, because the method of formalizing, listing all features in a large scale feature model as an analysis operation can be broken into an operation in local levels of construction of the feature model and this modularization transforms time consuming analysis operations to operations that can be calculated faster.

This paper provides three main contributions to the study of automatic analysis of software product lines. First, we present a formalism that constructs the semantics of complex constraint-based Extended Feature Models With

*Author for correspondence

Cardinality (Ext-FM-WC) using promotion technique in Z. Second, the structure of the formalism is in the form that constructs the Ext-FM-WC in a hierarchical or promoting manner and it has benefits in all editing and some analysis operations. Third, we show an example of the analysis operation that has promoting nature and can be apply in promoting manner in our formalism.

The remainder of the paper is structured as follows: In section 2, we formalize Ext-FM-WC with the promotion technique in Z and provide an example of formalizing Ext-FM-WC in step manner and a complex constraint. Section 3 discusses the results obtained and section 4 presents some conclusions.

In[6], the traditional notion of refactoring has extended for the SPL context. Authors has defined a feature model refactoring as a transformation that improves the quality of a feature model by improving (maintaining or increasing) its configurability. In this paper, a set of refactorings presented for FMs. For example, how to convert an Alternative relation to an or relation in a feature model.

A feature assembly framework has presented in [7], which is a new approach for creating feature models through feature composition and feature assembly. Furthermore, it promotes feature reusability by storing features in a so-called feature pool, which acts as a feature repository. They used a hybrid methodology that combines both a top down and a bottom up approach for modeling features in a product line to overcome the problem of scalability, abstraction mechanisms for feature models.

Czarnecki[8] provided a formalization of cardinality-based feature modeling and its specialization. They translated cardinality-based feature models into context free grammar. However, it does not support attributes, and global constraints.

The MUSCLES[9], presented a formal model of multi-step SPL configuration. It uses simple feature model and its focus is on constraints optimization in a feature model.

To our knowledge[10], is the first paper using Z for formalizing feature model. It is a formalization of simple feature model without attribute and cardinality and complex constraints. It does not cover modifications of the feature model.

In this paper, we present a formalization of the semantics of complex constraint-based Extended Feature Models With Cardinality (Ext-FM-WC) using promotion technique in Z. This formalism constructs the cardinalities for features, the cardinalities for groups, the attributes and the hierarchy of the feature model by promoting local state into global state. Constraints (basic and complex) are separately formalized.

Table 1 compares some works in the feature model refactorings. The criteria are (i) support for attributes; (ii) feature and group cardinalities; (iii) basic constraints, i.e. requires, excludes; (iv) complex constraints, i.e. Boolean constraints involving values of attributes; (v) formal semantics.

2. Formalizing Structure

This paper proposes to formalize Ext-FM-WC using Z specification language. Our motivation for defining a formal semantics for Ext-FM-WC is twofold. First, this formal account aims at avoiding unnecessary misinterpretations and confusion about the meaning of the notation that may arise from an informal definition.

For example, as reported in [12], the "ambiguity problem" with respect to optional features in feature groups is actually a case of notational redundancy. This kind of confusion can be avoided by providing precise semantics. Second, the formalization process with an impact on design decisions provided us more detailed understanding of various issues related to the notation[13]. For example, the addition of feature attributes and feature cloning in Ext-FM-WC represents a significant extension.

In this paper, we propose a promotion technique for formalizing software product line systems. The main

Table 1. Comparison of Ext-FM-WC to other FM modifications works

	Attributes	Feature and group cardinality	Basic constraints	Complex constraints	Formal semantics
8		✓	✓		✓
11			✓	✓	
6		~			
7		✓	✓		
10			✓		✓
9		✓	✓	✓	✓
Ext-FM-WC	✓	✓	✓	✓	✓

reason for choosing promotion technique for formalizing SPLS is that with factor the specifications – constructing large specifications out of small components - time consuming editing operations can be applied faster. The pattern of promotion technique that is used in this paper is multiple multi-level promotions.

The pattern of multiple multi-level promotions that are used in this paper for modeling SPL is presented in Figure 1. In our work, an Ext-FM-WC is presented with two subsystems (*GSubModel1* and *GSubModel2*) and a schema that is related to feature's attributes (*LAttribute*) with the multiple promotion technique. *GSubModel1* and *GSubModel2* subsystems created with multi-level promotion technique. The AND relation of these subsystems creates a schema that is used as a local state and will be promoted to global State (Figure 1).

2.1 Formalizing of Ext-FM-WC

The Feature Names is a set of names of the features in the Ext-FM-WC. For example, in Ext-FM-WC of Figure 2, FeatureNames== {F1, F2, F3, F4, F5, and F6}.

[FeatureNames]

The Mandatory, Optional, or and Alternative relations are dependency relations in an Ext-FM-WC. A feature and an attribute related to each other through the AttributeRel relation. In Figure 3, different relations are shown.

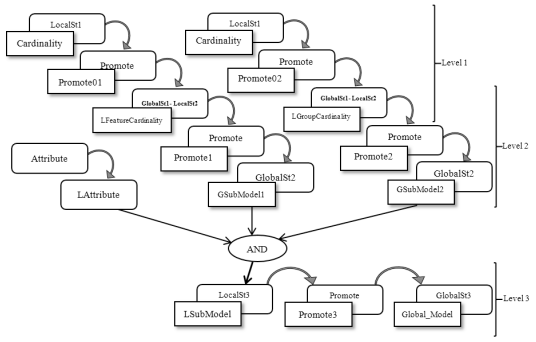

Figure 1. Structure of promotion technique in this paper.

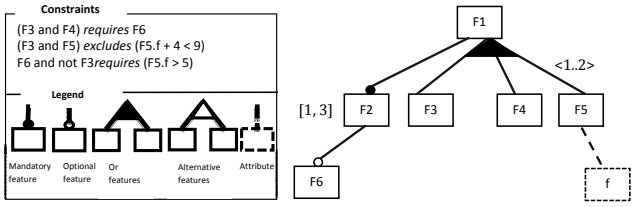

Figure 2. A sample Ext-FM-WC.

DRelation ::= Mandatory | Optional | Or | Alternative | AttributeRel

2.1.1 First Subsystem

In this section, the first subsystem in our multiple systems (*GSubModel1*) is created. *GSubModel1* in our multiple promotion patterns is one of the subsystems. *GSubModel1* is created with the multi-level promotion technique.

- First Level of *GSubModel1* subsystem

According to Figure 1, at the first level (*LocalSt1* – *GlobalSt1*) a cardinality is defined in the *LocalSt1* (*Cardinality* schema) and this cardinality is assigned to a feature in the *GlobalSt1* (*LFeatureCardinality* schema). *The Promote01* schema expresses the relationship between *LocalSt1* and *GlobalSt1*.

There are two types of cardinality in Ext-FM-WC: Feature cardinality and group cardinality. Feature cardinality defines how many times a given feature can be repeated. It's a kind of feature cloning. Group cardinalities define the number of features that can be chosen among a set of options under an 'Or' node. We can define cardinality as an ordered pair (lb, ub). Obviously, the lower bound must not exceed the upper one; at most they may have equal values.

In this subsystem only the feature cardinality, Mandatory and Optional relations are defined.

The *Cardinality* schema (*LocalSt1*) defines a set of bounds as below.

Cardinality

$$Bound: \mathbb{P}(\mathbb{N} \times \mathbb{N}_1)$$

$$\forall c: \mathbb{N} \times \mathbb{N}_1 \mid c \in Bound \bullet first\ c \leq second\ c$$

The *L FeatureCardinality* schema (*GlobalSt1*) assigns a cardinality to a feature. This schema is a local state at level two of subsystem *GlobalSubModel1*. The relationship between the state of a feature with cardinality in the

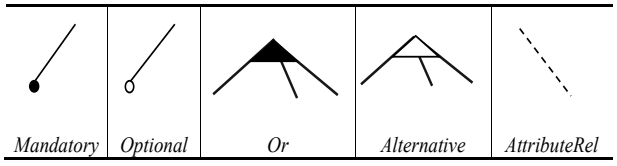

Figure 3. Different dependency relation in Ext-FM-WC.

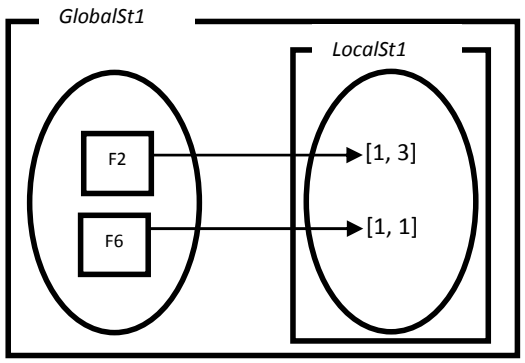

Figure 4. Some samples from *LocalSt1* and *GlobalSt1*of Figure 2.

global state and the state of a cardinality in the local state is illustrated in Figure 4.

LFeatureCardinality

FeatureCardinality: FeatureNames → Cardinality

The *Promote01* schema defines the relationship between the *Cardinality* schema (*LocalSt1*) and the *LFeatureCardinality* schema (*GlobalSt1*).

- Second Level of *GSubModel1* subsystem

According to Figure 1, at the second level (*LocalSt2* – *GlobalSt2*) a feature with cardinality is defined in *LocalSt2* (*LFeatureCardinality* schema) and this component is assigned to a *Mandatory* or *Optional* relation in *GlobalSt2* (*GSubModel1* schema). The *Promote1* schema expresses the relationship between *LocalSt2* and *GlobalSt2*.

Promote01

ΔCardinality
ΔLFeatureCardinality
FN?: FeatureNames

FN?∈domFeatureCardinality
θ Cardinality = FeatureCardinality FN?
FeatureCardinality′ = FeatureCardinality ⊕ {FN? ↦θ Cardinality′}

The local state (*LFeatureCardinality*) is defined before, but its global state is the *GSubModel1* schema. The *GSubModel1* schema associates each *DRelation* to a set of features with cardinality as a local state. In Ext-FM-WC, *Mandatory* and *Optional* relation can have only one child but an Or and *Alternative* relation can have more than one child. The relationship between the state of a feature with cardinality in the global state and the state of a feature with cardinality in the local state is illustrated in Figure 5.

GSubModel1

ChildFeature1: DRelation↔ LFeatureCardinality

---

dom ChildFeature1 = {Mandatory, Optional}

The *promote1* schema expresses the relationship between the *LFeatureCardinality* schema (*LocalSt2*) and the *GSubModel1* schema (*GlobalSt2*).

Promote1

ΔGSubModel1
ΔLFeatureCardinality
DR?: DRelation

DR? ∈ dom ChildFeature1
θLFeatureCardinality = ChildFeature1 DR?
ChildFeature1′ = ChildFeature1 ⊕ {DR? ↦ θ LFeatureCardinality′}

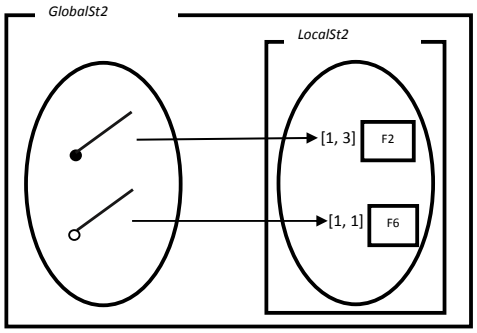

Figure 5. Some samples from the *LocalSt2* and *GlobalSt2* of Figure 2.

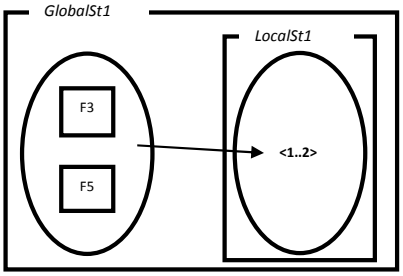

Figure 6. A sample of the LocalSt1 and GlobalSt1 of Figure 2.

2.1.2 Second Subsystem

In this section, the second subsystem in our multiple systems (*GSubModel2*) is created. The *GSubModel2* in our multiple promotion patterns is one of the subsystems. The *GSubModel2* is created with multi-level promotion technique.

- First Level of *GSubModel2* subsystem

According to Figure 1, at the first level (*LocalSt1* – *GlobalSt1*) a cardinality is defined in the *LocalSt1* (*Cardinality* schema) and this cardinality is assigned to a group of features in *GlobalSt1* (LGroupCardinality schema). The *Promote02* schema expresses the relationship between the *LocalSt1* and the *GlobalSt1*.

In this subsystem only Or and *Alternative* relations and group cardinalities are defined. The relationship between the state of a set of features with cardinality in the global state and the state of cardinality in the local state is illustrated in Figure 10.

- Second Level of GSubModel2 subsystem

According to Figure 3, at the second level (*LocalSt2* – *GlobalSt2*) a set of group features with cardinality is defined in the *LocalSt2* (*LGroupCardinality* schema) and this component is assigned to an Or or an Alternative relation in the *GlobalSt2* (*GSubModel2* schema). The *Promote2* schema expresses the relationship between *LocalSt2* and *GlobalSt2*.

The relationship between the state of a set of grouped feature with group cardinality in the global state and the state of a feature with cardinality in the local state is illustrated in Figure 7.

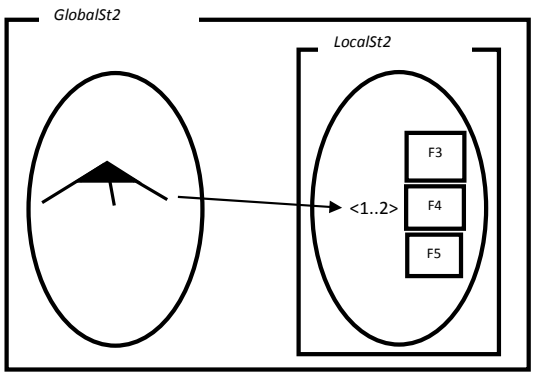

Figure 7. A sample of the LocalSt2 and GlobalSt2 of Figure 2.

2.1.3 Attributes in Ext-FM-WC

Attributes are concepts which denote properties of features (e.g., in Figure 2 f is an attribute of F5 feature). Each attribute has a set of values allowed for any particular attribute is referred to as its domain. *AttributeNames* is a set of names of attributes in the Ext-FM-WC.

$[AttributeNames]$

$Domain == \mathbb{F}\mathbb{Z}$

For example, in the Ext-FM-WC in Figure 2, AttributeNames == {f}, Domain = {{2, 3, 4, 5}, {10, 20, 30}}. Attributes have name, type (domain), and value. They are introduced with the *Attribute* schema. Each attribute name has a domain and a value. A Value is a member of the Domain.

Attribute

$$attrel: AttributeNames \rightarrow Domain$$
$$Value: AttributeNames \rightarrow \mathbb{Z}$$
$$\forall a: AttributeNames \bullet Value\ a \in attrel\ a$$

The *AttributeRel* relation can have one or more attributes as its children. These relations are shown in Figure 3.

LAttribute

$$ChildAttribute: DRelation \leftrightarrow \mathbb{F}\ Attribute$$
$$\forall ca: domChildAttribute \bullet ca = AttributeRel$$
$$\forall at: \mathbb{F}\ Attribute \bullet at \in ChildAttribute \ (\!|\ \{AttributeRel\}\ |\!) \Rightarrow \#at \geq 1$$

2.1.4 Ext-FM-WC

According to Figure 1, the *And* relation of subsystems *GSubModel1*, *GSubModel2* and *LAttribute* develop a new subsystem that we call it *LocalSubModel(LocalSt3).* The *LocalSubModel* is a subsystem of Ext-FM-WC that has features and group cardinality and attributes of features as the local states. In Figure 8 these local states are shown.

$LocalSubModel == GSubModel1 \wedge GSubModel2 \wedge LAttribute$

Some of the instances of *LocalSubModel* (*LocalSt3*) schemas are related to a parent feature for generating a *Global_Model* (*GlobalSt3*). The relationship between the *LocalSt3* and the *GlobalSt3* is illustrated in Figure 8.

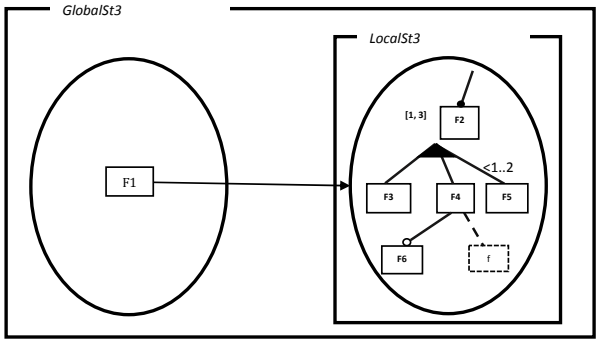

Figure 8. LocalSt3 and GlobalSt3 of Figure 2.

The *Promote* schema describes how the *GlobalSt3* is updated with the results of the local operations. This schema establishes the relationship of the state of a local instance to the *GlobalSt3*, and then shows how the after-state of an operation on that local instance is used to update the *GlobalSt3*.

Global_Model

$$Feature: FeatureNames \rightarrow \mathbb{P}_1 LocalSubModel$$

$$\forall f: dom\ Feature;\ lsm: ran\ Feature \bullet \forall\ a1: GSubModel1;\ a2:GSubModel2 \bullet$$
$$\forall\ b1: ran\ a1.ChildFeature1;\ b2: ran\ a2.ChildFeature2 \bullet$$
$$\forall\ c1: dom\ b1.FeatureCardinality;\ c2: dom\ b2.GroupCardinality \bullet \forall\ d: c2 \bullet f \neq c1 \land f \neq d$$

Promote

$$\Delta LocalSubModel$$
$$\Delta Global\_Model$$
$$f?: FeatureNames$$

$$f? \in dom\ Feature$$
$$\theta LocalSubModel \in Feature\ f?$$
$$Feature' = Feature \oplus \{f? \mapsto \{\theta LocalSubModel'\}\}$$

2.1.5 Constraints in Ext-FM-WC

An Ext-FM-WC is a Model that is introduced with *Global_Model* and a set of constraints between the features and attributes.

$$ConstModel == Global\_Model \times \mathbb{F}\ Constraints$$

Constraints in the Ext-FM-WC are in the form of *requires* and *excludes*. In *requires* constraints when feature X *requires* Y, it means that if X is included in a product, then Y should be included as well, but not vice versa. In *excludes* constraints when feature X *excludes* Y, it means that if X is included in a product, then Y should not be included and vice versa. These constraints can be written between two features, two attributes or a feature and an attribute. These constraints are defined in the following schema. RWFF is a Boolean relation that is actually the result of applying logical operator such as *and*, *or* and *not* on conditions. RWFF is defined in the recursive form because in the complex constriants, Boolean relations between features and attributes maybe appled in recursive form, e. g. A requires B and not C.

$$Constraints ::= requires \langle\!\langle\ RWFF \times RWFF\ \rangle\!\rangle \mid excludes \langle\!\langle\ RWFF \times RWFF\ \rangle\!\rangle$$

$$RWFF ::= and \langle\!\langle\ RWFF \times RWFF\ \rangle\!\rangle \mid or \langle\!\langle\ RWFF \times RWFF\ \rangle\!\rangle \mid not \langle\!\langle\ RWFF\ \rangle\!\rangle \mid cond \langle\!\langle\ Condition\ \rangle\!\rangle$$

Since in Ext-FM-WC, feature's attributes with integer domains are used, it is required that in addition to logical and relational operators, arithmetic operators is defined too. A condition is an attribute or a feature or is defined by applying relational operators on two arithmetic expressions.

$$Condition ::= Greater \langle\!\langle expre \times expre \rangle\!\rangle \mid EqualGreater \langle\!\langle expre \times expre \rangle\!\rangle \mid Less \langle\!\langle expre \times expre \rangle\!\rangle \mid EqualLess \langle\!\langle expre \times expre \rangle\!\rangle \mid Equal \langle\!\langle expre \times expre \rangle\!\rangle \mid notEqual \langle\!\langle expre \times expre \rangle\!\rangle \mid F \langle\!\langle FeatureNames \rangle\!\rangle \mid A \langle\!\langle AttributeNames \rangle\!\rangle$$

An Expression is an integer value, the value of an attribute or defined by applying arithmetical operators on two expression in the recursive form, e.g., "A.a + B.b – C.c" where A, B, and C are features and a, b, and c are the attributes of the features.

$$expre ::= add \langle\!\langle expre \times expre \rangle\!\rangle \mid sub \langle\!\langle expre \times expre \rangle\!\rangle \mid multi \langle\!\langle expre \times expre \rangle\!\rangle \mid divi \langle\!\langle expre \times expre \rangle\!\rangle \mid modu \langle\!\langle expre \times expre \rangle\!\rangle \mid A1 \langle\!\langle AttributeNames \rangle\!\rangle \mid val \langle\!\langle \mathbb{Z} \rangle\!\rangle$$

2.1.6 Validity of Constraints in an Ext-FM-WC

Constraints are used in many operations that are applied on an Ext-FM-WC. For example, if we have a product and we want to specify that is a valid product in Ext-FM-WC or not, we must examine that all features and attributes

in a product are in Ext-FM-WC or not and whether all constraints that are in Ext-FM-WC are satisfied in the product or not. For doing this operation it is better to examine the validity of a constraint with expressing the semantics of constraints that introduced in section 2.1.5.

Axiomatic definition is *ValidConstraint1* defines and validates requires and excludes constraints. The *requires* and *excludes* relations take two expressions that are a combination of arithmetic, logical and relational (complex constraints) relations.

- *Requires* (rwff1, rwff2) means that complex constraint rwff1 requires complex constraint rwff2, e.g., if rwff1 is in a product, rwff2 must be in the product too.
- *Excludes* (rwff1, rwff2) means that complex constraint rwff1 *excludes* the existence of complex constraint rwff2, e.g., if rwff1 is in a product, rwff2 must not exist in the product.

isValidConstraint1: Constraints

rwff1, rwff2: RWFF •

requires(rwff1, rwff2)= isValidConstraint1 ⇔ (rwff1= isValidConstraint2⇒rwff2=isValidConstraint2)

∧excludes(rwff1, rwff2)= isValidConstraint1 ⇔ (rwff1 = isValidConstraint2 ⇒¬rwff2=isValidConstraint2)

Axiomatic definition *isValidConstraint2* defines and validates *RWFF* complex constraint. *RWFF* can be a condition or a logical relation of two conditions (and, or, not).

is ValidConstraint2: RWFF

∀ r1, r2: RWFF; c1, c2: Condition •

and(r1, r2) = isValidConstraint2 ⇔ (r1 = isValid Constraint31 r1 ∧ r2 = isValidConstraint31 r2)

∧or(r1, r2) = isValidConstraint2 ⇔ (r1 = isValid Constraint31 r1 ∨ r2 = isValidConstraint31 r2)

∧not(r1) = isValidConstraint2 ⇔ (¬r1 = isValid Constraint31 r1)

∧cond(c1) = isValidConstraint2 ⇔ c1 = isValid Constraint32

Axiomatic definition *isValidConstraint32* defines and validates a condition. A condition can be a feature or the value an attribute or a relational relation of two expressions.

isValidConstraint31: RWFF → RWFF

∀ r1:RWFF• isValidConstraint31 r1 = r1

isValidConstraint32: Condition

∀ c1, c2: Condition; e1, e2: expre; f: FeatureNames; a: AttributeNames; c: Condition •

Greater(e1, e2) = isValidConstraint32 ⇔ (isVali Constraint4 e1 > isValidConstraint4 e2)

∧ EqualGreater(e1, e2) = isValidConstraint32 ⇔ (isValidConstraint4 e1≥ isValidConstraint4 e2)

∧ Less(e1, e2) = isValidConstraint32 ⇔ (isValid Constraint4 e1< isValidConstraint4 e2)

∧ EqualLess(e1, e2) = isValidConstraint32 ⇔ (isValidConstraint4 e1≤ isValidConstraint4 e2)

∧ Equal(e1, e2) =isValidConstraint32 ⇔ (isValid Constraint4 e1= isValidConstraint4 e2)

∧ notEqual(e1, e2) = isValidConstraint32 ⇔ (is ValidConstraint4 e1≠ isValidConstraint4 e2)

∧ F(f) = isValidConstraint32 ⇔ f = isValidConstraint5 f

∧ A(a) = isValidConstraint32 ⇔ Value(a) = isValid Constraint6 a

Axiomatic definition *isValidConstraint4* is the value of an attribute or takes an expression as the input and returns the arithmetical result of the expression as outputs.

isValidConstraint4: expre → ℤ

∀ e1, e2: expre; a, b: ℤ; a1, b1: AttributeNames•

isValidConstraint4 (add(e1, e2))= a+b

∧isValidConstraint4 (sub(e1, e2))= a−b

*∧ isValidConstraint4 (multi(e1,e2))= a*b*

∧ isValidConstraint4 (divi(e1,e2))= a div b

∧ isValidConstraint4 (modu(e1, e2))= a mod b

∧ isValidConstraint4 (A1(a1)) = isValidConstraint6 a1

∧ isValidConstraint4 (val(a)) = a

Axiomatic definition *isValidConstraint5* takes a feature as the input and returns that feature as outputs.

isValidConstraint5: FeatureNames → FeatureNames

∀f: FeatureNames • isValidConstraint5 (f) = f

Axiomatic definition *isValidConstraint6* takes an attribute as the input and returns the value of that attribute as outputs.

$$isValidConstraint6: AttributeNames \rightarrow \mathbb{Z}$$
$$Attribute$$
$$\forall a: AttributeNames \bullet isValidConstraint6\ (a) = Value(a)$$

2.2 A Sample Formalized Complex Constraint

The syntax and semantics presented in the previous sections are used in step manner in the Figures 6, 8, 10, 12, and 13 for formalizing the Ext-FM-WC in the Figure 2. In this section, we follow to formalize F6 ^ ¬ F3 requires (F5.f > 5) complex constraint that is related to the Figure 2 according to the syntax and semantics presented in the section 2.1.5 and 2.1.6. The syntax of this constraint according to the proposed formalism in the section 2.1.5 is as follows:

- = requires(RWFF, RWFF)
- = requires(and(RWFF, RWFF), cond(Condition))
- = requires(and(cond(Condition), not(RWFF)), cond(Greater(expre, expre)))
- = requires(and(cond(F(FeatureNames))), not(cond(Condition))), cond(Greater(A1(AttributeNames), Val(Z)))
- = requires(and(cond(F(FeatureNames))), not(cond(F(FeatureNames)))), cond(Greater(A1(AttributeNames), Val(Z)))

The semantics of this constraint according to proposed formalism in the section 2.1.6 is as follows:

According to ValidConst1, the validation semantic of formalized syntax in line 1 above is as:

$$rwff1 \Rightarrow rwff2$$

According to ValidConst2, and ValidConst32 , the validation semantic of formalized syntax in line 2 is as (rwff1 is a and relation, and rwff2 is a condition):

$$(rwff3 \wedge rwff4)\ ) \Rightarrow (exp1 > exp2)$$

According to ValidConst2, and ValidConst32 , the validation semantic of formalized syntax in line 3 is as:

$$(F(FeatureNames) \wedge not(rwff5)) \Rightarrow (exp1 > exp2)$$

According to ValidConst4, and ValidConst5, the validation semantic of formalized syntax in line 4 is as:

$$(F6 \wedge not(F(FeatureNames))) \Rightarrow (A1(a1) > val(z))$$

According to ValidConst4, and ValidConst6, the validation semantic of formalized syntax in line 5 is as:

$$F6 \wedge \neg F3 \Rightarrow (F5.f > 5)$$

3. Discussions and Evaluation

The structure of constructing an Ext-FM-WC using the (multiple-multilevel) promotion technique has a lot of benefits on analysis operations that have naturally promoting structure. In analysis operations, listing features in a large scale feature model as an analysis operation can be broken into an operation in local levels of construction of the feature model (bellow schemas) and this modularization transforms time consuming analysis operations to operations that can be calculated faster. Another example of analysis operations is the number of products of an Ext-FM-WC operation that is a very time consuming operation in the large scale feature models.

As a more automated solution, alloy analyzer is used to verify our model. Z/EVES can be used to verify Ext-FM-WC, however, as a theorem prover, the verification process requires, much user interaction[10]. Alloy Analyzer is a fully automated reasoning tool that requires no user interactions. It provides simulation and checking functionalities, a suitable way for finding counterexamples and identifying the origin of the inconsistencies in the model.

In this section, first we show some analysis operations that can be applied on Ext-FM-WC (section 3.1) and summarizes the analysis results in section 3.1.3 and then we compare our work with other's work (section 3.2).

3.1 Empirical Evaluation

In this section, the empirical evaluation examines the qualities of our approach in isolation. We show that the analysis operations with promotion pattern have better results in calculation time.

3.1.1 The Number of Products

In this section, we show the benefit of our approach on an analysis operation. For calculating the total number of products of Ext-FM-WC, we have to determine all valid products of Ext-FM-WC. It is useful to perform analysis on specific models, such as the Ext-FM-WC from Figure 2. Because we know exactly the number of elements of each signature declared in Alloy model, we can perform a complete analysis using the Alloy Analyzer.

We formalize this analysis operation on Ext-FM-WC in two ways: with the promotional manner and without the promotional manner.

3.1.1.1 The Number of Products Operation without Promotional Manner

In this analysis operation, first we examine that a product is a valid product of a constraint based Ext-FM-WC or no. After that we sum all valid products using Number of Products schema. A valid product is a product that satisfies all the constraints that are between different relations of Ext-FM-WC (dependency and cross tree constraints).

3.1.1.2 The Number of Products Operation in Promotional Manner

We modularize this operation in the level 2 and 3 of the Figure 1 for computing the number of products of the SPL. In this operation, we do not consider constraints because the constraints applied to the whole of the model (the number of products of the SPL is not naturally promoting structure because of the constraints that is applied to the whole of the model).

ValidProduct

VP: Product ConstModel
Product

$\forall p$: Product; cm: ConstModel; m: Global_Model; LFC: LFeatureCardinality; LGC: LGroupCardinality;
$A: \mathbb{P}$ FeatureNames • $(p, cm) \in VP \Leftrightarrow$
$\forall$(p1: p.Product1; p2: p.Product2; f: m.Feature; fc: LFC.FeatureCardinality
• ($\forall$lf: second f • ($\forall$cf1: lf.ChildFeature1; cf2: lf.ChildFeature2; ca: lf.ChildAttribute
• (first p1 = first f $\wedge$ first cf1 = Mandatoryfi $\Rightarrow$ ($\forall$c: dom second cf1• c $\in$ dom Product1))
$\wedge$ (first p1 = first f $\wedge$ first cf1 = Optional $\Rightarrow$ ($\forall$c: dom second cf1 • c $\in$ dom Product1 $\vee$ c $\notin$ dom Product1))
$\wedge$ (first p1 = first f $\wedge$ first cf2 = Or $\Rightarrow$ ($\forall$c: dom second cf2 • c $\in$ dom Product1 $\vee$ c $\notin$ dom Product1))
$\wedge$ (first p1 = first f $\wedge$ first cf2 = Alternative $\Rightarrow$ ($\exists_1$ c: dom second cf2 • c $\in$ dom Product1))
$\wedge$ ($\forall$c: second ca • second p2 $\in$ c.attrel(first p2))
$\wedge$ ($\forall$b: ran LFC.FeatureCardinality •
($\forall$d: b.Bound • (first p1 = first fc $\wedge$ first d $\leqslant$ second p1 $\leqslant$ second d)))
$\wedge$ ($\forall$a:domLGC.GroupCardinality;A:$\mathbb{P}$FeatureNames • ($\forall$b: a • ((first p1 = b $\Rightarrow$ b $\in$ A)
$\wedge$ ($\forall$b: ran LGC.GroupCardinality • ($\forall$d: b.Bound • (b = LGC.GroupCardinality a $\wedge$ first d $\leqslant$ #A
$\wedge$ #A $\leqslant$ second d)))))))))
$\wedge$ ($\forall$x: second cm • x = isValidConstraint1)

Product

Product1: FeatureNames $\rightarrow \mathbb{N} \setminus \{0\}$
Product2: AttributeNames $\rightarrow \mathbb{Z}$

allProducts

allProduct: ConstModelf $\rightarrow \mathbb{P}$ Product
ValidProduct

$\forall cm$: ConstModel $\mathbb{N}$ allProduct cm = {p: Product | (p, cm) $\in$ VP }

NumberOfProducts

NOP :ConstModelf $\rightarrow \mathbb{N}$
allProducts

$\forall cm$: ConstModel · NOP cm = #allProduct cm

- **The number of products in the first subsystem of the level 2.** First, in the level 2 of the first subsystem of the Figure 1, we evaluate that in a local level, a product is a valid product or not. The schema Valid Product Feature takes as an input a product and GSubModel1 and a feature with feature cardinality and evaluates that the cardinality is validated or no.

ValidProductFeature

VP1: product $\rightarrow$ GSubModel1

$\forall$p: product, m: GSubModel1, LFC: LFeature Cardinality ·
(p, m) $\in$ VP1 $\Leftrightarrow$ $\forall$p1: p.product1, fc: LFC.FeatureCardinality,
b:domLFC.FeatureCardinality, d1: b.Bound, k: d1.c
/ ran p1 =ran fc $\wedge$ ran k $\leqslant$ domp1 $\wedge$ domp1 $\leqslant$ dom k

The schema *all Product Feature Local* defines all of the valid product in the GSubModel1 level.

allProductFeatureLocal

$allProductLoc: GSubModel1 \rightarrow \mathbb{P}\ product$

$\forall gsm1: GSubModel1 \bullet$
$gsm1.allProductLoc = \{p: product / (p, gsm1) \in ValidProductFeature.VP1\}$

Finally, the schema *Numof Prod Local Feature* takes as an input a GlobalSubModel1 and returns as an output the number of valid products in this local model.

NumOfProdLocalFeature

$NOP1 : GSubModel1 \rightarrow \mathbb{N}$

$\forall gsm1: GSubModel1 \bullet$
$NOP1\ gsm1 = \#allProductFeatureLocal.allProductLoc$

- **The number of products in the second subsystem of the level 2.** Now, in the level 2 of the second subsystem of the Figure 2, we evaluate that in a local level, a product is a valid product or not. The schema Valid Product Group takes as an input a product and GSubModel1 and a feature with group cardinality and evaluates that the cardinality is validated or no.

ValidProductGroup

$VP2: product \rightarrow GSubModel2$

$\forall p: product, m: GSubModel2, LGC: LGroupCardinality \bullet$
$(p, m) \in VP2 \Leftrightarrow \forall p1: p.product1, a:ran\ LGC.Group\ Cardinality, A: \mathbb{P}\ Featurenames,$
$b:\mathbb{P}Featurenames, g:domLGC.GroupCardinality, d1: g.Bound, k: d1.c$
$|(ran\ p1 \in b \Rightarrow b \in A) \wedge g = LGC.GroupCardinality\ a \wedge rank \leqslant \#A \wedge \#A \leqslant domk$

The schema *all Product Group Local* defines all of the valid products in the GSubModel2 level.

allProductGroupLocal

$allProductLoc: GSubModel2 \rightarrow \mathbb{P}\ product$

$\forall gsm2: GSubModel2 \bullet$
$all\ Product\ Loc\ gsm2 = \{p: product \mid (p, gsm2) \in ValidProductGroup.VP2\}$

Finally, the schema *Num of Prod Local Group* takes as an input a GlobalSubModel2 and returns as an output the number of valid products in this local model.

NumOfProdLocalGroup

$NOP2 : GSubModel2 \rightarrow \mathbb{N}$

$\forall gsm2: GSubModel2 \bullet$
$NOP2\ gsm2 = \#allProductGroupLocal.allProductLoc$

- **The number of products in the level 3.** In the level 3 of the Figure 1, we evaluate that in the global level, a product is a valid product or not. The predicate Valid Product Global evaluates that the feature and attribute relations are validated or no.

ValidProductGlobal

$VP3: product \rightarrow Global\_Model$

$\forall p: product, m: Global\_Model, LFC: LFeatureCardinality, LGC: LGroupCardinality,$
$A: \mathbb{P}\ Featurenames, c1:Featurenames, d:FNSet, e:d.fns, s: \mathbb{P}\ AttributeSet, s1: s.atts$
$\bullet (p, m) \in ValidProductGlobal.VP3 \Leftrightarrow$
$\forall p1: p.product1, p2: p.product2, f: m.Feature, lf: f\ Featurenames, a: lf.lsms,$
$cf1: a.ChildFeature1, cf2: a.ChildFeature2, ca: a.ChildAttribute /$
$ran\ p1 = ran\ f \wedge ran\ cf1 = Mandatory \Rightarrow$
$(c1 \in ran\ cf1.FeatureCardinality \Rightarrow c1 \in ran\ p1) \wedge$
$ran\ p1 = ran\ f \wedge ran\ cf1 = Optional \Rightarrow$
$(c1 \in ran\ cf1.FeatureCardinality \Rightarrow c1 \in ran\ p1 \vee c1 \notin ran\ p1) \wedge$
$ran\ p1 = ran\ f \wedge ran\ cf2 = Or \Rightarrow$
$(d \in ran\ cf2.GroupCardinality \Rightarrow (c1 \in e) \Rightarrow c1 \in ran\ p1 \vee c1 \notin ran\ p1)$
$ran\ p1 = ran\ f \wedge ran\ cf2 = Alternative \Rightarrow$
$(d \in ran\ cf2.GroupCardinality \Rightarrow (c1 \in e) \Rightarrow c1 \in ran\ p1 \wedge$
$s \in domca \Rightarrow dom\ p2 \in s1.attrel(ran\ p2)$

Each of the schemas *Num of Prod Global Feature* and *Num of Prod Global Group* in a promoting manner return as an output the number of valid products in the global model.

NumOfProdGlobalFeature

$NOP3 : \mathbb{N}$

$\forall$ *lsm, lsm': LocalSubModel, lsms: LocalSubModelSet, gm, gm': Global_Model, finput: Featurenames, gsm1:GSubModel1, cf1:dom (GSubModel1.ChildFeature1), cf2: dom(GSubModel2.ChildFeature2), p:product, LFC: LFeatureCardinality, LGC: LGroupCardinality, A: $\mathbb{P}$ Featurenames, d: FNSet, e:d.fns, s: $\mathbb{P}$ AttributeSet,s1: s.atts* $\cdot$

$(Promote(lsms, lsm, lsm', gm, gm', finput) \wedge VProduct(p, gm, LFC, LGC, A, finput, d, e, s, s1)) \Rightarrow$ $NOP3 = NumOfProdLocalFeature\ gsm1 else\ 0$

NumOfProdGroupGlobal

$NOP4 : \mathbb{N}$

$\forall$ *lsm, lsm': LocalSubModel, lsms: LocalSubModelSet, gm, gm': Global_Model, finput: Featurenames, gsm2:GSubModel2, cf1:dom (GSubModel1.ChildFeature1), cf2: dom(GSubModel2.ChildFeature2), p:product, LFC: LFeatureCardinality, LGC: LGroupCardinality, A: $\mathbb{P}$ Featurenames, d: FNSet, e:d.fns, s: $\mathbb{P}$ AttributeSet,s1: s.atts* $\bullet$

$(Promote(lsms, lsm, lsm', gm, gm', finput) \wedge VProduct(p, gm, LFC, LGC, A, finput, d, e, s, s1)) \Rightarrow$ $NOP3 = NumOfProdLocalGroup\ gsm2\ else\ 0$

Finally, the total number of products of the global model is the sum of two previous functions.

totalnumofproduct

$NOP5 : \mathbb{N}$

$\forall$ *lsm, lsm': LocalSubModel, lsms: LocalSubModelSet, gm, gm': Global_Model, finput: Featurenames, gsm2:GSubModel2, cf1:dom (GSubModel1.ChildFeature1), cf2: dom(GSubModel2.ChildFeature2), p:product, LFC: LFeatureCardinality, LGC: LGroupCardinality, A: $\mathbb{P}$ Featurenames, d: FNSet, e:d.fns, s: $\mathbb{P}$ AttributeSet,s1: s.atts* $\cdot$

$NOP5 = NOP3 + NOP4$

Calculating the number of products in this promoting manner has a lot of advantages, for example we can calculate the number of products in a sub model of the global model. If we use this formalism for implementing a tool for analysis operations in the Ext-FM-WC, we can save the number of products in the local levels and then by adding them together have the total number of products in the global model.

3.1.2 Analysis Results

Due to the nature of Alloy Analyzer (i.e. counter example generation), the evaluations of the constraints in Ext-FM-WC are guaranteed to be correct in the case where the conditions actually hold (i.e. no false negatives are produced).

Table 2 (and Figure 9) shows some analysis operations results. The full analysis results are in another work of authors in this paper[14]. The evaluation was repeated 100 times on some Ext-FM-WCs with 6 (example in the Figure 2), 50, 100 features. Features have some attributes and cross-tree constraints such as Ext-FM-WC in the Figure 2. The table shows the average time (expressed in milliseconds).

Table 2. Some analysis operations results

		Valid Product		All Product	Number of Product	
	No. features	Without promotion technique	With promotion technique	With promotion technique	Without promotion technique	With promotion technique
SAT4J Solver	6	54 ms	40 ms	56 ms	55 ms	45 ms
MiniSat Solver		23 ms	13 ms	24 ms	30 ms	14 ms
SAT4J Solver	50	103 ms	60 ms	100 ms	110 ms	65 ms
MiniSat Solver		45 ms	25 ms	40 ms	56 ms	30 ms
SAT4J Solver	100	168 ms	143 ms	156 ms	183 ms	140 ms
MiniSat Solver		60 ms	44 ms	64 ms	80 ms	54 ms

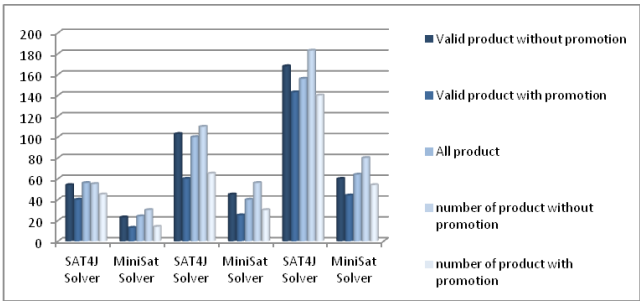

Figure 9. Some analysis operations results.

3.2 Comparative Evaluation

In comparative evaluation, we show the strengths of our approach with respect to other works that did edit on feature models. We are implementing a tool for automatic analysis of Ext-FM-WC with promotion structure. For solving and analysis our formalism, we use different solver and analyzer that Alloy provides. We use several small and medium-sized feature models that were publicly available, e.g., FAME-DBMS[15] with 21 features, decomposed Berkeley DB[16] with 38 features, e-Shop[17] with 287 features. We did not find large feature models for a thorough evaluation. Therefore, we perform an experiment using randomly generated feature models with different characteristics. The evaluation method that we use in this section is similar to[12]; Independent variables in our experiment are (a) the number of features, and (b) the number of features and attributes in a feature model. As a dependent variable, we measured the time needed to determine the classification. Extraneous variables such as hardware are kept constant by performing all measurements on the same Windows XP PC with 2.4 GHz and 2 GB RAM such as hardware that used in[12] for comparison.

3.2.1 Experimental Results

3.2.1.1 Number of Features

First, we measured how calculation time scales as feature models increase in size. Therefore, we varied the size of the generated feature model between 10 and 1000 features. For each feature model, we performed 10 edits. We performed 100 repetitions for each model size, as there is a fluctuation of calculation times depending on the randomly generated models (with the same size). For small feature models (< 200 features), the average calculation time is less than 0.02 second. For large feature models (1000 features), the average calculation time is 0.1 seconds. In Figure 10, we show the result of these measurements and compare it with the work of authors in[12]. In [11], the authors have been shown the result of their theory on feature models with 50 to 300 features; because their calculation time in 12 different refactoring operations take about 1 to 8 minutes, their work is not comparable to our work.

3.2.1.2 Number of Features and Attributes

First, we measured how calculation time scales as feature models increase in feature and attribute size. Therefore, we varied the size of the generated feature model between 10 and 1000 features and for every feature model the number of attributes is the half of the features. For each feature model, we performed 10 edits that are related to attributes. For example, adding an attribute, deleting an attribute, or updating an attribute in its domain or value. We performed 100 repetitions for each model size, as there is a fluctuation of calculation times depending on the randomly generated models (with the same size). For small feature models (< 200 features), the average calculation time is less than 0.02 second. For large feature models (1000 features), the average calculation time is 5 seconds. In Figure 11, we show the result of these measurements. We do not find any other work that we can compare it with our work.

4. Conclusion

This paper provides three main contributions to the study of software product lines. First, we present a formalism that constructs the semantics of extended feature models with cardinality (Ext-FM-WC) using promotion technique in the first-order logic in Z. Our motivation for defining a formal semantics for Ext-FM-WC is twofold. First, this formal account aims at avoiding unnecessary misinterpretations and confusion about the meaning of

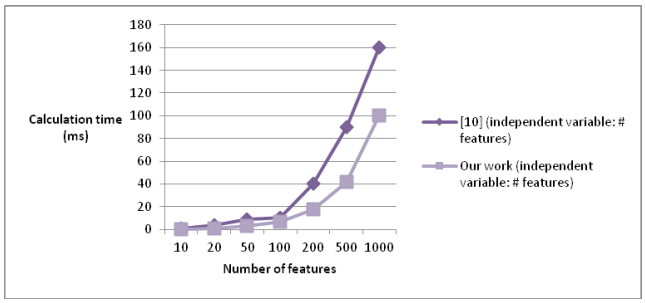

Figure 10. Calculation time in milliseconds for 10 random edits.

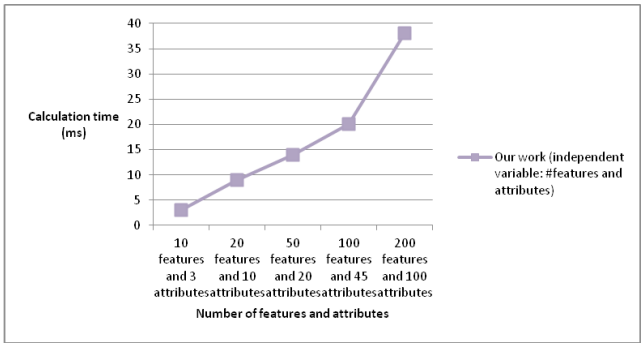

Figure 11. Calculation time in milliseconds for 10 random edits.

the notation that may arise from an informal definition. Second, the process of formalization gave us a deeper understanding of various issues concerning the notation, which had impact on design decisions. Second, the structure of the formalism that is multiple multi-level promotion technique is in the form that constructs the Ext-FM-WC in a hierarchical or promoting manner and it has benefits in some analysis operations. Third, we show an example of the analysis operation that has promoting nature and can be apply in promoting manner in our formalism.

5. References

1. Benavides D, Segura S, Ruiz-Corte's A. Automated analysis of feature models 20 years later: A literature review. Information Systems. 2010; 35(6):615–36.
2. Kang K, Cohen S, Hess H, Novak W, Peterson S. Feature-Oriented Domain Analysis (FODA) feasibility study. Software Engineering Institute, Carnegie Mellon University; 1990 Nov. Technical Report - CMU/ SEI-90-TR-21.
3. Benavides D, Trinidad P, Ruiz-Cortes A. Automated Reasoning on Feature Models. Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE'05); 2005.
4. Batory D, Benavides D, Ruiz-Cortes A. Automated analysis of feature models: challenges ahead. Communications of the ACM. 2006; 49(12):45–7.
5. Benavides D, Batory D, Heymans P, Ruiz-Corte's A. First Workshop on Analyses of Software Product Lines; 2008 Sep.
6. Alves V, Gheyi R, Massoni T, Kulesza U, Borba P, Lucena C. Refactoring Product Lines. Proceedings of the 5th international conference on Generative programming and component engineering (GPCE'06); ACM; 2006. p. 201–10.
7. Abo Zaid L, Kleinermann F, Troyer O. Feature assembly framework: Towards scalable and reusable feature models. Fifth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS '11); ACM; 2011.
8. Czarnecki K, Helsen S, Eisenecker U. Formalizing cardinality-based feature models and their specialization. Software Process: Improvement and Practice Journal. 2005; 10(1):7–29.
9. Whited J, Benavidese D, Saxenaf T, Doughertyd B, Schmidtf DC, Galindoe JA. Automated reasoning for multi-step feature model configuration problems. Proceedings of the 13th International Software Product Line Conference (SPLC '09); 2009. p. 11–20.
10. Sun J, Zhang H, Fang Li Y, Wang H. Formal Semantics and Verification for Feature Modeling. Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems; 2005. p. 303–12.
11. Thum T, Batory D, Kastner C. Reasoning about Edits to Feature Models. Proceedings of the 31st International Conference on Software Engineering (ICSE '09); 2009. p. 254–64.
12. Bontemps Y, Heymans P, Schobbens P, Trigaux J. Semantics of FODA feature diagrams. Proceedings SPLC 2004 Workshop on Software Variability Management for Product Derivation; 2004.
13. Czarnecki K, Eisenecker, UW. Generative Programming: Methods, Tools and Applications. Boston, MA: Addison-Wesley; 2000.
14. Ajoudanian SH. Mirian Hosseinabadi, SH. Automatic promotional specialization, generalization and analysis of Extended Feature Models with Cardinalities in Alloy. Journal of logical and algebraic methods in programming. 2014; 84(5):640–67.
15. Rosenmuller M et al. FAME-DBMS, Tailor-made Data Management Solutions for Embedded Systems. Proceeding of EDBT Workshop on Software Engineering for Tailor-made Data Management; ACM; 2008; p. 1–6.
16. Kastner C, Apel S, Batory D. A case study implementing features using AspectJ. Proceeding of International Software Product Line Conference; IEEE Computer Society; 2007. p. 223–32.
17. Mendonca M, Wasowski A, Czarnecki K, Cowan D. Efficient Compilation Techniques for Large Scale Feature Models. Proceeding of International Conference Generative Programming and Component Engineering; ACM; 2008. p. 13–22.

Appendix A: Z notation[5]

Declarations and Definitions	
a: A	Declarations
A | B	Definitions
Logic	
$p \wedge q$	Conjunction
$p \vee q$	Disjunction

¬p	Negation
p ⇒q	Implication
p ⇔q	Equivalence
Quantification	
∀ x : A | q(x) • p(x)	Universal quantification. Equivalent to the following: ∀x •x ∈ A ∧q(x) ⇒p(x)
∃ x : A | q(x) • p(x)	Existential quantification. Equivalent to the following: ∃x •x ∈ A ∧q(x) ∧p(x)
∃1 x : A | q(x) • p(x)	Unique existential quantification
Sets	
x ∈ A	Set membership
ϕ	Empty set
A⊆B	Set inclusion
{x, y, . . .}	Set of elements
(x, y)	Ordered pair
A × B	Cartesian product
PA	Powerset of A
FA	Finite powerset of A
{ x : A | q(x) • e(x)}	The set { e(x) | x ∈ A ∧q(x)}
A ∩ B	Set intersection
A∪B	Set union
A \ B	Set difference
∩ A	Generalized or distributive intersection
∪A	Generalized or distributive union
#A	Cardinal of set A
Relations and Functions	
A φ B	Relation
A → B	Total function
A →B	Partial function
domf , ran f	Domain and range of a function f
±	((R1, R2), S) ∈ ±⇔ S = (dom R_2ψ R_1) ∪ R_2
Schema expressions	
ΔS	Shortcut for S *f*Sə
ΞS	Shortcut for S *f*Sə*f*θS = θSə