Indian Journal of Science and Technology, *9(31), DOI: 10.17485/ijst/2016/v9i31/93731, August 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Verification of SPL Feature Model by using Bayesian Network

Shamim Ripon[1*], Musfiqur Rahman[2], Javedul Ferdous[1] and Md. Delwar Hossain[1]

[1]Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh; dshr@ewubd.edu
[2]Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada

Abstract

Feature Tree represents all the features along with their relationship of a Software Product Line. Any defect in feature model can diminish the benefits of product line approach. Hence, the analysis of feature model plays a key role towards the success of any Software Product Line. This paper presents various analysis rules for cardinality-based feature model of both dead and false optional features. These rules are then verified by using Bayesian Network Based inference mechanism. Such verification not only confirms the analysis rules of the feature trees but also ensures the applicability of probabilistic information into the feature trees.

Keywords: Bayesian Network, Dead Feature, False Optional Feature, Feature Analysis, Software Product Line

1. Introduction

Software engineering involves investments into requirements analysis, architecture, design, documentation etc., and it has been pressured to introduce new products and add functionality to existing products at a quick step to be able to compete the demand of market[1]. Although software development has been focused on original development but now it has been a recognized fact that to achieve better and flexible software at a lower cost it is needed to adopt a design process that is based on systematic Software Reuse[2].

A collection of software that is related to each other in terms of features they share among themselves is known as Software Product Line (SPL). Those related software are said to be belonging to a software family where each of the software in the family are characterized by the features that vary from product to product[3]. Among many other usefulness of the concept of SPL, reusability and increase in the quality of the software product with respect to the time to market are the most important ones. A model is created based on the common and varying features of a SPL. The fundamental objective behind a model-based representation of the SPL is mainly to facilitate the stakeholder to select their desired product configurations in terms of features. The common requirements among all the members of the software family can easily be handled since they are present in every member and thus, is included in the overall family architecture. However, dealing with the variant requirements is not that straightforward because modeling variants makes the domain analysis task more complicated by adding an extra level of complexity. This is why management of variants belongs to one of the critical areas in SPL.

Feature Model (a.k.a. Feature Tree) is used to describe the features and their dependencies for creating valid products in SPL. Any defects in feature model can significantly diminish the benefits of the product line approach. Among the various defects, this paper is focused on dead and false optional features. Dead features are features that are not present in any valid product configuration. False optional features, on the other hand, are optional features but they are present in all valid product configurations.

The requirements of software are becoming complex day by day and a lot of uncertainties are also present in the requirements description. Bayesian Network (BN)

*Author for correspondence

has already been applied in several software engineering problems to handle uncertainties. So far, significant amount of research has been carried out on the use of BNs for predicting defects in software, reliability of software etc. However, hardly any of those works have addressed the issues related to requirement management in SPL, particularly feature model analysis. In this paper we focus on addressing this area where we show that the inference mechanism based on the conditional probability of BNs can be leveraged in analyzing various defects in feature model.

The objective of this paper is to define various dead and false optional feature analysis rules by using first order logic. These rules are then analyzed and verified by using Bayesian Network. We describe how a feature tree can be represented by using BN and how the analysis rules can then be defined in BN. We then show our adopted approach to prove the analysis rules by using Bayesian inference mechanism. The significance of our analysis rules is that we defined the rules for cardinality[4,5] based feature model instead of regular feature model. The BN based verification shows how to deal with uncertainties in software requirements by using AI techniques[6].

The organization of the rest of the paper is as follows. A brief overview of various feature types and relationship as well as cardinality based feature tree is given in Section 2. Then Section 3 illustrates the relationship between feature diagram and BN and how a feature tree can be represented into a BN. In the following section, we give the first order logic definition of the analysis rules and draw scenario for each case of the analysis rules. Then, the analysis rules defined in first order logic have been modeled and verified using BNs in Section 5. Finally, Section 6 concludes the paper by summarizing our findings and outlining our future plan.

2. Feature Model and Cardinality

In the hierarchy of the feature model each node represents a feature while the edges represent the relationships or constraints among the features[8]. The relationships among the parent features and their child features along with cross-tree constraints that typically define the inclusion and exclusion of feature with respect to some other features[9]. Here, we briefly discuss the different notations of feature models.

Mandatory: There is a mandatory relationship between a child feature and its parent feature if- whenever the parent feature appears in a valid product the child feature is also included.

Optional: If the inclusion of the child feature is optional in a product in which its parent appears, i.e., the child feature may or may not be included, then we define this relationship among a child feature and its parent feature as an optional relationship.

Alternative: When only one of the child features originated from a parent feature can be a part of the product in which the parent feature has been included, we define that relationship between the set of child feature and the parent feature as an alternative relationship.

Or: There is an alternative relationship between a set of child features and their parent when only one or more of the child features can be a part of the product in which the parent feature has been included.

These four relationships are the basic parent-child relationships that are commonly dealt with in a feature model. Other than these parent-child relationships there are also two types of cross-tree constraints among the features. The cross-tree constraints are as follows:

Requires: This constraint defines that when the inclusion of one feature (say feature A) requires the inclusion of another feature (say feature B) in a valid product. In that case the constraint is verbally represented as, "Feature A requires feature B".

Excludes: This constraint is defines that the inclusion of one feature (say feature A) requires the exclusion of another feature (say feature B) in a valid product. In that case the constraint is verbally represented as, "Feature A excludes feature B".

Other than the relationships discussed above some experts proposed extension of Feature Oriented Domain Analysis (FODA) feature models with cardinalities in order to incorporate practical applications and conceptual completeness[10]. Basic two notations for cardinality based feature models are defined below:

Feature Cardinality: Feature cardinalitv is defined as a sequence of intervals denoted by [m,n m,n] where m as lower bound and n as upper bound. The number of instances the feature can be selected for a valid product is defined by these intervals[9]. By using this definition we can establish that when both lower and upper bound of feature cardinality are equal to 1 then that relationship becomes the original mandatory relationship defined as a non-cardinality based relationship. Similarly for lower bound equals to 0 and upper bound equals to 1 then that relationship becomes the original optional relation-

ship which has been defined as a non-cardinality based relationship.

Group Cardinality: Group cardinality is defined as a sequence of interval denoted $<^{m,n}>$ where m and n represents respectively the upper and the lower bounds of the interval[9]. The maximum and minimum numbers of child features from a group cardinality that can be a part of a valid product where the parent feature is present are denoted by the upper and lower bounds. By using this definition we can establish that when both lower and upper bound of group cardinality are equal to 1 then that relationship becomes the original alternative relationship defined as a non-cardinality based relationship. Similarly for lower bound equals to 0 and upper bound equals to 1 then that relationship becomes the original or relationship which has been defined as a non-cardinality based relationship.

3. Feature Diagram and Bayesian Network

The notation of feature modeling was proposed in[7]. Modeling the features of a software system family is considered as one of the most important parts of domain analysis. This is both formal and graphical representation of the common and varying features of a product family. Features are hierarchically organized in a feature tree where the root is the representation of the domain concept while other nodes represent various features. *Mandatory* and *Optional* are two classes of features. The relationships among the features which are originated from the same parents can be either *Or* relationship or *Alternative* relationship. Any cross-tree (or cross-hierarchy) relationship is denoted as a dependency.

BNs are the graphical representation of the dependencies among different random variables. BN is represented by a directed graph in which the nodes of the graph represent a random variable (or an event) whereas the directed edges among different nodes represent the relationships (or dependencies) among the random variables (or events). Each of these nodes has Conditional Probability Distributions (CPD) associated to it. The random variables can be discrete. In that case the representation of the CPD is renamed as Node Probability Table (NPT). This table lists all the possible probability values that a node can take based on different combinations of values that the parents of these nodes can take. Many cases may arrive where the nodes have two possible outcomes. This special condition is handled by representing the CPD binary probabilities of 0 and 1. This first step of constructing a BN is to identify the random variables associated with the domain we are studying. The next step is to model those variables which can be done trivially. By determining the relationships among the variables and finally the NPT is constructed for all the variables in separately.

Additionally, a BN is represented by a directed acyclic graph or a DAG. The direction of an edge represents the dependency between the two nodes which are connected by that edge. For example, if a directed edge connects two nodes X_i and X_j, it indicates that X_i is a parent node for the node X_j and, thus, X_j has a dependency on X_i. On the contrary, feature tree is nothing but a tree data structure with a root node and a number of non-root nodes. The features are represented by the non-root nodes whereas, the corresponding edges between features and sub-features represent the parent-child relationships. Because of the strong resemblance between BNs and feature trees, any information that is represented by a feature tree can be conveniently represented without any additional effort by BNs also. Furthermore, one of the most important and unique characteristics of BNs is that it is one of the most widely used techniques for incorporating probabilistic information when we are dealing with uncertainty. An example of feature tree and the corresponding BN for the tree is shown in Figure 1. For each of the variables in the BN a NPT is built from the dependency information of the variables.

4. Feature Analysis Rules

With the increase of the number of features in a feature model the potential complexity incorporated with the analysis of the model is becoming a task of extensive hardship. The additional level of complexity is making the analysis task error-prone and, hence, number of defects may get

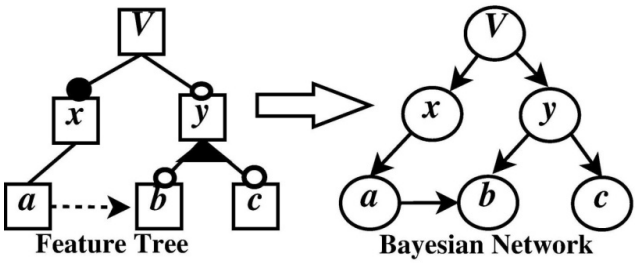

Figure 1. BN representation of a feature tree.

introduced at the modeling stage. Therefore, additional care is required while creating feature model so that we can precisely include the domain information within the model. It is worth noting that modeling the domain correctly with no information lost is a critical task. Moreover, identification of the defects at the proper stage of the software development lifecycle is a prerequisite for developing valid software that fulfills all the requirements of the stakeholders.

Defects found is a SPL feature model has been termed as *anomalies*. In order to detect and correct inconsistencies as well as redundancies addressing these anomalies by exploiting intelligent techniques and tools has been suggested by many experts[9,11]. Different kinds of approaches, such as defining corresponding *diagnostic task*[12] transforming feature models into an alternative representation of a Constraints Satisfaction Problem (CSP)[13] etc. have also been proposed by different authors. BNs has also been used for making predictions about software defects, reliability etc. in different research works. But hardly any of the works addressed how to use BNs in order to analyze the feature model. The work presented in this paper deals with two specific types of anomalies namely *False Optional*, *Dead Feature*.

False Optional: Features that are selected for all valid products in spite of being defined as optional features are call False Optional (FO) features.

Dead Feature: Features that are defined as optional features but never get selected for any of the valid products are known as dead features (DF)[14].

Due to the ability of Feature Models to derive a potentially large number of products, any defects in a Feature Model will inevitably affect many products of the product line. In this section, we represent a set of rules defined in first order logic for dead and false optional features based on group cardinality. The following First Order Logic (FOL) predicates are used to define the rules.

- $variation\_point(v)$: This predicate means, feature v has one or more child feature(s).
- $mandatory\_variant(v, x)$: This predicate means, feature x is a mandatory child feature of v.
- $optional_{variant(v,x)}$: This predicate means, feature x is an optional child feature of v.
- $requires(x, y)$: This predicates means, when x is selected y is also selected beacuse in a valid product x always requires y.
- $excludes(x, y)$: This predicates means, x excludes y from being selected for a valid product.
- $cardinality\ (G, k, m, n)$: This predicate indicates G is a group cardinality with total number of features k, lower bound m and upper bound n.
- $parent_{feature}\ (G, f)$: This predicate indicates f is the parent feature in the group cardinality G.
- $child_{features}\ (G, a, \ldots, d\ )$: This predicate indicates that $a, \ldots, d$ are the child features of the parent under the group cardinality G.
- $dead_{features}(a, \ldots, d)$: This predicate indicates that features $a, \ldots, d$ are dead features.
- $false_{optional}(a, \ldots, d)$: This predicate indicates that features $a, \ldots, d$ are false optional features.
- $select\ (x)$: This predicate means that the feature x is selected.

4.1 False Optional

Rule 1: One or more features become false optional when they are grouped by group cardinality with a mandatory parent having lower bound of m and upper bound of $n(m \leq n)$ with $[k - m]$ dead features within the cardinality, where k represents the total number of features within the cardinality (Figure 2).

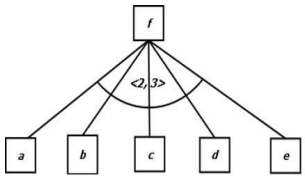

Figure 2. Scenario of cardinality based false optional feature Rule 1.

$$\forall v, f, a, b, c, d, e, G, k, m, n \cdot cardinality(G, k, m, n) \wedge parent\_feature(G, f) \wedge child\_feature(G, a, b, c, d, e) \wedge dead\_feature(a, b, c) \wedge select(f) \Rightarrow select(d) \wedge select(e)$$

Rule 2: In a group cardinality having total k features along with upper bound of n and lower bound of m, all of the features within the cardinality become false optional when k = m = n, where $k, m, n > 0$ (Figure 3).

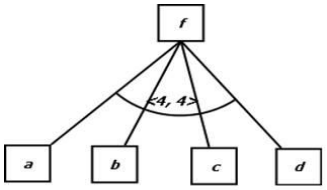

Figure 3. Scenario of cardinality based false optional feature Rule 2.

$$\forall v, f, a, b, c, d, G, k, m, n \cdot cardinality(G, k, m, n) \wedge parent\_feature(G, f) \wedge child\_feature(G, a, b, c) \wedge k = m \wedge m = n \wedge select(f) \Rightarrow select(a) \wedge select(b) \wedge select(c) \wedge select(d)$$

In the feature tree for this rule (Figure 3), *a, b, c, d* are ($k = 4$) features within the group cardinality having upper bound $n = 4$ and lower bound $m = 4$. Thus all of the features are going to be selected when the mandatory father f is selected.

Rule 3: One or more features become false optional when they are grouped by a group cardinality with a mandatory parent having lower bound of m and upper bound of n ($m \neq n$) with [k-m] dead features within the cardinality, where k represents the total number of features within the cardinality (Figure 4).

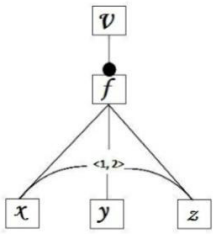

Figure 4. Scenario of cardinality based false optional feature Rule 3.

$$\forall v, f, x, y, z, G, k, m, n \cdot cardinality(G, k, m, n) \wedge variation\_point(v) \wedge mandatory\_variant(v, f) parent\_feature(G, f) \wedge child\_feature(G, x, y, z) \wedge dead\_feature(x, y) \wedge select(f) \Rightarrow \neg select(x) \wedge \neg select(y) \wedge select(z)$$

Rule 4: One or more features become false optional when they are grouped by a group cardinality with a mandatory parent having lower bound of m and upper bound of n and $m = n$ with [k-m] dead features within the cardinality, where k represents the total number of features within the cardinality (Figure 5).

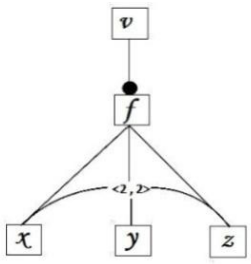

Figure 5. Scenario of cardinality based false optional feature Rule 4.

$$\forall v, f, x, y, z, G, k, m, n \cdot cardinality(G, K, m, n) \wedge m = n \wedge k > m \wedge variation\_point(v) \wedge mandatory\_variant(v, f) \wedge parent\_feature(G, f) \wedge child\_feature(G, x, y, z) \wedge variation\_point(f) \wedge child\_feature(G, x, y, z) \wedge dead\_feature(x) \wedge select(f) \Rightarrow \neg select(x) \wedge select(y) \wedge select(z)$$

4.2 Dead Feature

Rule 1: In a group cardinality having total k features along with upper bound of n and lower bound of m, [k-m] features within the cardinality become dead feature when m = n and there are n (or m) false optional features, where $k, m, n > 0$ and $k \geq n, m$ (Figure 6).

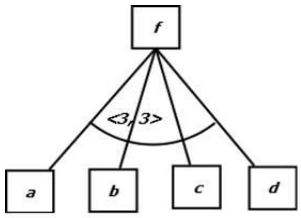

Figure 6. Scenario of cardinality-based dead feature Rule 1.

$$\forall v, f, a, b, c, d, e, G, k, m, n \cdot cardinality(G, k, m, n) \wedge parent\_feature(G, f) \wedge child\_feature(G, a, b, c, d, e) \wedge false\_optional(b, c, d) \wedge m = n \wedge select(f) \Rightarrow \neg select(a)$$

In the feature three for this rule (Figure 6), $k = 4$, $m = n = 3$. Let *b, c* and *d* be false optional features. According to the rule above (k-n) = (4-3) = 1 feature is dead feature which is feature *a* in this case.

Rule 2: An optional feature that does not belong to group cardinality becomes dead when it requires p features that belong to group cardinality with lower bound of m and upper bound of n, where $p, m, n > 0$ and $p > n \geq m$ (Figure 7).

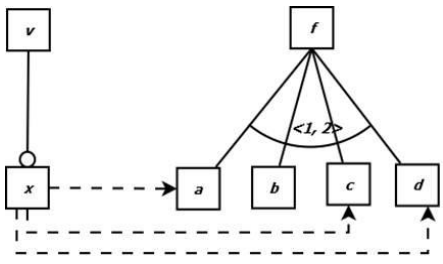

Figure 7. Scenario of cardinality-based dead feature Rule 2.

$$\forall v, x, f, a, b, c, d, G, k, m, n \cdot cardinality(G, k, m, n) \wedge variation\_point(v) \wedge$$

$$optional\_variant(x) \wedge parent\_feature(G, f) \wedge child\_feature(G, a, b, c, d)$$

$$\wedge requires(x, a) \wedge requi$$

In the feature three for this rule in Figure 7, x is an optional feature under the variation point v. The features a, b, c, and d are within the group cardinality under feature f with upper bound of 2 and lower bound of 1. Feature x requires 3 features that belong to the group cardinality. But, because of the fact that the upper bound of this group cardinality is 2, x can never get selected. Therefore, x is a dead feature.

Rule 3: A feature defined as an optional feature becomes a dead feature whenever it belongs to group cardinality

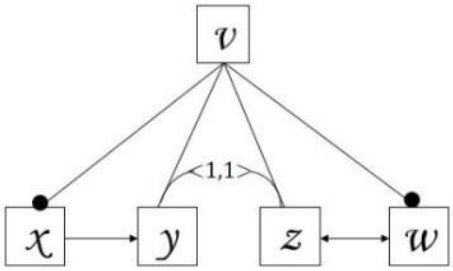

Figure 8. Scenario of cardinality-based dead feature Rule 3.

and the number of false optional feature is equal to the cardinality upper bound (Figure 8).

$$\forall v, x, y, z, G, m, n, k \cdot$$
$$cardinality(G, k, m, n) \wedge m = n$$
$$\wedge parent\_feature(G, v) \wedge child\_feature(G, y, z)$$
$$\wedge variation\_point(v) \wedge mandatory\_variant(v, x)$$
$$\wedge mandatory\_variant(v, w) \wedge requires(x, y)$$
$$\wedge exclude(w, z) \wedge select(x)$$
$$\Rightarrow select(y) \wedge select(w) \wedge \neg select(z)$$

In the graph in Figure 8, feature y and z connected with <1, 1> group cardinality. Feature x has been defined as a mandatory feature. Moreover, an optional feature y has a *requires* relation with feature x. Thus y is a false optional feature. Since upper bound is 1, y alone can be selected. Therefore z becomes a dead feature.

Rule 4: A feature belonging to group cardinality becomes a dead feature if some another feature within the cardinality excludes it (Figure 9).

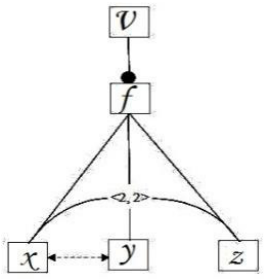

Figure 9. Scenario of cardinality-based dead feature Rule 4.

$$\forall v, x, y, z, f, G, k, m, n \cdot cardinality(G, k, m, n) \wedge (m = n) \wedge (k > m, n)$$
$$\wedge variation\_point(v) \wedge mandatory\_variant(v, f)$$
$$\wedge parent\_feature(G, f) \wedge child\_feature(G, x, y, z)$$
$$\wedge exclude(x, y) \wedge select(f)$$
$$\Rightarrow select(x) \wedge select(z) \wedge \neg select(y)$$

5. Bayesian Network based Analysis

In this section of the paper, the Bayesian Network representation for false optional and dead features is defined. We define BN by finding the all possible dependencies among the features based on our logical representation.

5.1 False Optional

Rule 1: In the feature tree for this rule in Figure 10(a), the total number of features $k = 5$, $m = 2$ and $n = 3$. Let there are $(k - m) = (5 - 2) = 3$ dead features a, b, and c. Therefore,

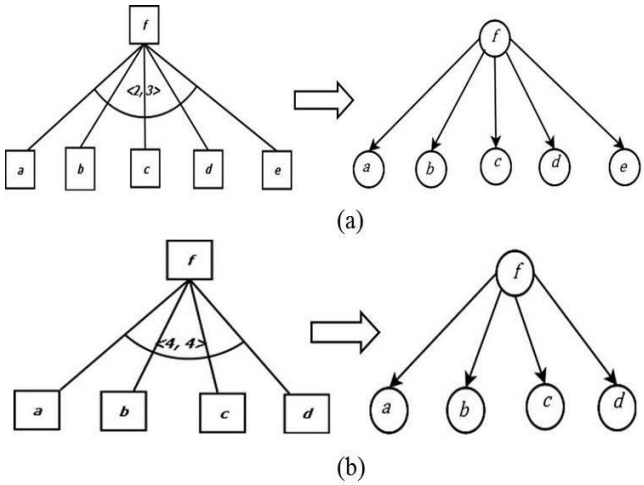

Figure 10. Bayesian representation of false optional Rule 1 and Rule 2.

to satisfy the lower bound of the group cardinality *d* and *e* are false optional features. In the BN, the child features *a, b, c, d* and *e* are dependent of the parent feature *f*. There is no other dependency. It has been declared that features *a, b* and *c* are dead features. Therefore, the probability of these features being selected is always 0. The NPT for this network is shown in Table 1.

Table 1. NPT of Rule 1 of cardinality based false optional feature

f	
T	*F*
1	0

a		
f	*T*	*F*
T	0	1
F	0	1

b		
f	*T*	*F*
T	0	1
F	0	1

c		
f	*T*	*F*
T	0	1
F	0	1

d		
f	*T*	*F*
T	1	0
F	0	1

e		
f	*T*	*F*
T	1	0
F	0	1

$$P(e,d,c,b,a,f) = P(e|f)P(d|f)P(c|f)P(b|f)P(a|f)P(f)$$

Now, if we choose to select 2 (lower bound) of the 5 features we have $\binom{5}{2} = 10$ combinations. Similarly, if we choose to select 3 (upper bound) of the 5 features we have $\binom{5}{3} = 10$ combinations. Thus, there are total 20 possible combinations that satisfy the lower and upper bound constraint of this given group cardinality. Let us consider some of the 20 combinations and determine the probability for those particular combinations of being valid.

$$P(a=T,b=T|f=T) = \frac{P(a=T,b=T,f=T)}{P(f=T)}$$

$$= \frac{\sum_{c,d,e\in\{T,F\}} P(a=T,b=T,c,d,e,f=T)}{P(f=T)}$$

By expanding each expressions of the above equation and putting appropriate probability value from the NPT for each variable we can draw the following conclusion.

$$P(a=T,b=T|f=T) = \frac{0+0+0+0+0+0+0+\mathbf{0}}{1} = \mathbf{0}$$

Similarly, we can show the following equations.

$$P(a=T,c=T|f=T) = \frac{0+0+0+0+0+0+0+\mathbf{0}}{1} = \mathbf{0}$$

$$P(b=T,c=T|f=T) = \frac{0+0+0+0+0+0+0+\mathbf{0}}{1} = \mathbf{0}$$

$$P(a=T,b=T,c=T|f=T) = \frac{0+0+0+\mathbf{0}}{1} = \mathbf{0}$$

$$P(c=T,d=T,e=T|f=T) = \frac{0+0+0+\mathbf{0}}{1} = \mathbf{0}$$

The similar procedure can be followed for any other combination and same result can be achieved with probability value of 0 except for the only valid combination of the child features within the group cardinality. The combination is as follows.

$$P(d=T,e=T|f=T) = \frac{P(d=T,e=T,f=T)}{P(f=T)}$$

$$= \frac{\sum_{a,b,c\in\{T,F\}} P(a,b,c,d=T,e=T,f=T)}{P(f=T)}$$

By expanding each expressions of the above equation and putting appropriate probability value from the NPT for each variable we can draw the following conclusion.

$$P(d=T,e=T|f=T) = \frac{0+0+0+0+0+0+0+\mathbf{1}}{1} = \mathbf{1}$$

Hence, *d* and *e*, although defined as optional features, turn out to be False Optional features.

Rule 2: In the feature tree for this rule (Figure 10 (b)), *a, b, c, d* are ($k = 4$) features within the group cardinality having upper bound $n = 4$ and lower bound $m = 4$. Thus all of the features are going to be selected when the mandatory father *f* is selected. From the BN representation we see that features *a, b, c,* and *d* depend on *f* (which is their parent feature). There is no other dependency. Thus, for each of the child features, whether that feature will get selected or not is dependent on f only. The corresponding NPT for this network is given in Table 2.

Table 2. NPT of Rule 2 or cardinality based false optional features

f	
T	*F*
1	0

a		
F	*T*	*F*
T	1	0
F	0	1

b		
f	*T*	*F*
T	1	0
F	0	1

c		
f	*T*	*F*
T	1	0
F	0	1

D		
f	*T*	*F*
T	1	0
F	0	1

$$P(d,c,b,a,f) = P(d|f)P(c|f)P(b|f)P(a|f)P(f)$$

Now, under the given scenario there is only $\binom{4}{4} = 1$ possible combination of the feature for satisfying the given constraint. We can prove that this combination is valid by doing some straightforward calculations based on the joint probability function and the values in the NPT.

$$P(a=T,b=T,c=T,d=T|f=T) = \frac{P(a=T,b=T,\ c=T,d=T,f=T)}{P(f=T)}$$

By evaluating the expressions in both numerator and denominator of the above equation and putting appropriate probability values from the NPT we can conclude the following:

$$P(a = T, b = T, c = T, d = T | f = T) = \frac{1}{1} = 1$$

Hence, all four child features of *a, b, c,* and *d* are False Optional features.

Rule 3: In the BN of Figure 11(a), feature *f* is a variant of the variation point *v* with three variants *x, y, z* of its own. Thus, whether or the the features *x, y, z* will get selected depends on the selection of the features *v* and *f*. Additionally, *f* is dependent on *v*. Table 3 shows the NPT for *v, x* and *y*.

Based on the lower and the upper bound of the group cardinality we can see that after *f* is selected. There are $3c_2 + 3c_1 = 6$ possible combinations of feature from the group cardinality.

We can use the Baye's rule for calculating the probability that feature z will be selected. NPT from Table 3 gives us the required CPD.

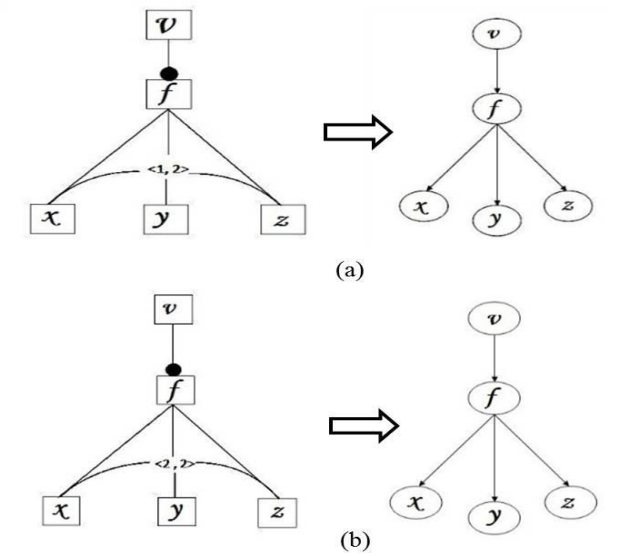

Figure 11. Bayesian representation of false optional Rule 3 and Rule 4.

Table 3. NPT for variables in false optional rule 1.

v	
T	*F*
1	0

f		
v	*T*	*F*
T	1	0
F	0	1

x		
f	*T*	*F*
T	0	1
F	0	1

y		
F	*T*	*F*
T	0	1
F	0	1

z		
f	*T*	*F*
T	1	0
F	0	1

$$P(z, f) = P(z|f)P(f|v)P(v)$$

$$P(z = T|f = T) = \frac{P(z = T, f = T)}{P(f = T)}$$

$$\frac{\sum_{v\in\{T,F\}} P(z = T, f = T, v)}{\sum_{v\in\{T,F\}} P(f = T, v)} = \frac{P(z = T, f = T, v = T) + P(z = T, f = T, v = F)}{P(f = T, v = T) + P(f = T, v = F)}$$

We can conclude the following,

$$P(z = T|f = T) = \frac{1 + 0}{1 + 0} = 1$$

We observe that if feature *f* is selected, the optional feature *z* gets selected. Due to the reason that feature *x* and *y* are dead, by the rules of cardinality, the remaining feature *z* automatically becomes false optional feature. The BN verification of Rule 4 can be performed in a similar manner (Figure 11(b)).

5.2 Dead Feature

Rule 1: In the feature three for this rule (Figure 12(a)), *k* = 4, *m* = *n* = 3. Let *b, c* and *d* be false optional features. According to the rule above (*k*-*n*) = (4-3) = 1 feature is dead feature which is feature *a* in this case. In the BN, all the child features *a, b, c, d* and *e* are dependent of the parent feature *f*. Features *b, c*, and *d* have been declared as false optional features. Therefore, the probability of these features being selected is always 1 as long as the parent feature *f* has been selected. As the number of features that have been selected from the group cardinality has reached

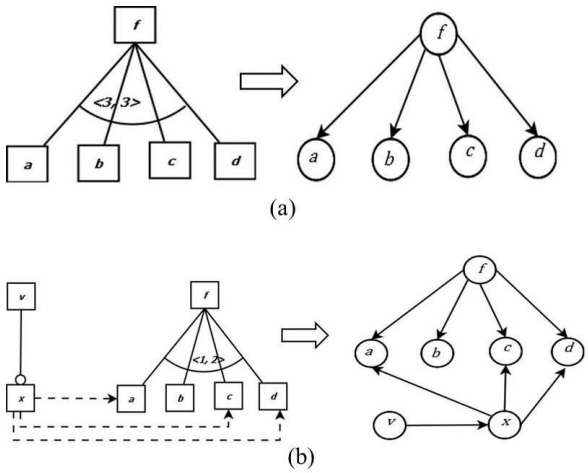

Figure 12. BN representation of cardinality based dead features Rule 1 and Rule 2.

the upper bound *n*, no other additional feature can ever be selected. Consequently, *a* will never be selected. The joint probability function for this rule is as follows:

$$P(d,c,b,a,f) = P(d|f)P(c|f)P(b|f)P(a|f)P(f)$$

Now, under the given scenario there are $\binom{4}{3} = 4$ possible combinations of the feature for satisfying the given constraint. In a similar way as shown in previous rules, after constructing the NPT for this rule we can show that,

$$P(a = T, b = T, c = T|f = T) = \frac{0+0}{1} = 0$$

Similarly we can show the following,

$$P(a = T, b = T, d = T|f = T) = \frac{0+0}{1} = 0$$

and

$$P(a = T, c = T, d = T|f = T) = \frac{0+0}{1} = 0$$

For the remaining combination we get,

$$P(b = T, c = T, d = T|f = T) = \frac{0+1}{1} = 1$$

Hence, *a* is a dead feature.

Rule 2: In the feature three for this rule in Figure 12(b), *x* is an optional feature under the variation point *v*. The feature *a, b, c,* and *d* are within the group cardinality under feature *f* with upper bound of 2 and lower bound of 1. Feature *x* requires 3 features that belong to the group cardinality. But, *x* can never get selected due to the fact that the upper bound of the group cardinality has been set to 2. Thus, x becomes dead feature. From the BN we can see, *x* is a variant of feature *v*. Child features *a, b, c,* and *d* are dependent on the parent feature *f*. Furthermore, *a, c,* and *d* also have dependencies on *x*. The selection of *a, c* and *d* will depend on both *x* and *f*. On the other hand, b is dependent on *f*. Another important point to be noticed here is that unlike the NPT of the earlier rules the following NPT has entries with values other than 0's and 1's. Because there is no constraint for False Optional (*FO)* or Dead Feature (*DF)* for the features within the cardinality, each of the features has equal probability of being selected. In this case ¼ = 0.25 is the value of probability for each of the features under the cardinality being selected. The NPT for this network is shown in Table 4.

The joint probability function is as follows:

Table 4. NPT for Rule 2 of cardinality based dead feature

v

T	F
1	0

x

v	T	F
T	1	0
F	0	1

f

T	F
1	0

a

f	x	T	F
T	T	1	0
T	F	0.25	0.75
F	T	0	1
F	F	0	1

b

F	T	F
T	0.25	0.75
F	0	1

c

f	x	T	F
T	T	1	0
T	F	0.25	0.75
F	T	0	1
F	F	0	1

d

f	x	T	F
T	T	1	0
T	F	0.25	0.75
F	T	0	1
F	F	0	1

$$P(x,v,a,c,d,b,f) = P(x|v)P(v)P(a|x,f)P(b|f)P(c|f,x)P(d|f,x)P(f)$$

Now, since there are total $\binom{4}{2} + \binom{4}{1} = 10$ valid combinations of features from the group cardinality. But none of those combinations satisfy the constraint for *x* to be a part of a valid product. Therefore, *x* becomes a dead feature. We can calculate the probability of *x* being selected for each of the valid combinations and we will find that the probability value is always 0.

One of the valid combinations is when only one of the features from the group cardinality is selected. An example is as follows:

$$P(x = T|a = T, b = F, c = F, d = F) :$$

$$= \frac{P(x = T, a = T, b = F, c = F, d = F)}{P(a = T, b = F, c = F, d = F)}$$

$$= \frac{\sum_{v,f\in\{T,F\}} P(x = T, v, a = T, c = F, d = F, b = F, f)}{\sum_{v,x,f\in\{T,F\}} P(x, v, a = T, b = F, c = F, d = F, f)}$$

By expanding each expressions of the above equation and putting appropriate probability value from the NPT for each variable we can draw the following conclusion.

$$P(x = T|a = T, b = F, c = F, d = F) = 0$$

By doing similar calculation we can find that the probability value of x being a part of a valid product for any other combination is always 0.

$$P(a = F, b = T, c = F, d = F) = 0$$

$$P(a = F, b = F, c = T, d = F) = 0$$

$$P(a = F, b = F, c = F, d = T) = 0$$

$$P(a = T, b = T, c = F, d = F) = 0$$

and so on. Hence, *x* is a dead feature.

Rule 3: In the Bayesian network in Figure 13, feature *x*, *y*, *z* and *w* are the four variants of *v*. Thus the outcome of the event of *v* being or not being selected directly influences the outcome of the events of these variant features being or not being selected. Thus, *x* depends only on *v*, however the probability that the feature *y* is selected depends on *x* also. Moreover, *w* is only dependent on *v*. But *z* is dependent on whether *v* is selected or not. Both *y* and *z* depend on (*v*, *x*) and (*v*, *w*) at the same time. The NPT are given in Table 5.

$$P(z,w) = P(z|w)P(w|v)P(v)$$

$$P(z=T|w=T) = \frac{P(z=T,w=T)}{P(w=T)} = \frac{\sum_{v\in\{T,F\}}P(z=T,w=T,f=T,v)}{\sum_{v\in\{T,F\}}P(f=T,v)}$$

$$= \frac{P(z=T,w=T,v=T)+P(z=T,w=T,v=F)}{P(w=T,v=T)+P(w=T,v=F)}$$

We can conclude the following after using the values from NPT:

$$P(z=T|w=T) = \frac{0+\mathbf{0}}{1+\mathbf{0}} = \mathbf{0}$$

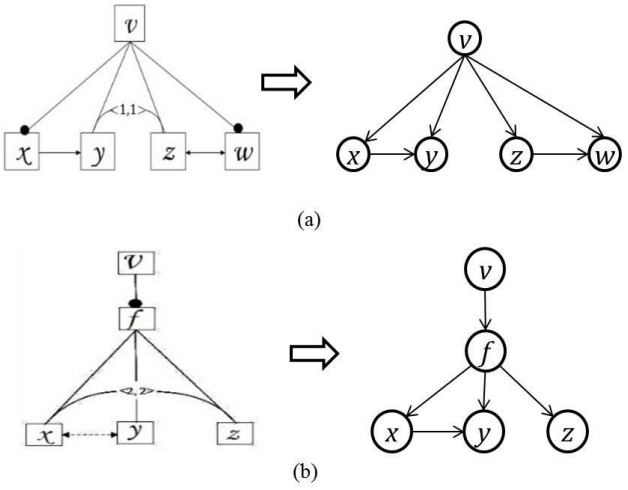

Figure 13. BN representation of cardinality based dead features Rule 3 and Rule 4.

Table 5. NPT for dead feature Rule 3

F

T	F
1	0

a

v	**T**	**F**
T	1	0
F	0	1

b

v	*x*	**T**	**F**
T	**T**	1	0
T	**F**	1	1
F	**T**	1	1
F	**F**	0	1

c

v	**T**	**F**
T	0	1
F	0	1

D

v	**T**	**F**
T	1	0
F	0	1

We observe from the result that optional feature *z* is selected. Similar verification can also be conducted for Rule 4.

6. Conclusions

A number of significant efforts have been made so far for modeling software product line feature model. Semantic web-based approach[18,19] and Rule-based approach[15,20] are two well-known approaches among these efforts. Semantic web technology is used to capture domain knowledge. Meaningful and shared ontological description of a specific domain can be expressed by this technology[18]. On the contrary, different rules are defined and verified by using first order logic in rule-based approach[17,20]. However, none of these approaches dealt with any kind of uncertainty in feature analysis.

We have exploited the theory of probabilistic reasoning by using Bayesian Networks. A feature tree can be represented by BN without going through any additional hardship as the structure of a feature tree resembles that of a BN. Since a feature tree is more like an And-Or tree, the interdependencies among the features can easily be found and thus, the conditional probability function as well as the node probability table can easily be constructed based on the feature dependencies.

In comparison to our earlier work[15] where only first order rules are defined but BN verification was not taken into account. Later we proposed a BN based verification method[16] but cardinality based feature tree is not considered there. Application of BN for such verification helps us to gain much knowledge into the feature tree as well as shows us how uncertainties can be handled in a feature tree.

Only a few scenarios are considered in this paper. We are currently developing rules for other scenarios of dead and false optional features. We are also interested to define and verify rules for wrong cardinality in a feature tree.

7. References

1. Nyholm C. Product line development– An overview. Building Reliable Component- Based Systems. Extended Report for Crnkovic I, Larsson M, editors. Artech House; 2002 Jul. p. 44–58.
2. Somerville I. Software Reuse. Software Engineering. 9th ed. Pearson; 2010.
3. Clements PC, Northrop LM. Software product lines: Practices and patterns. SEI Series in Software Engineering. Addison-Wesley; 2001.

4. Czarnecki K, Helsen S, Eisenecker U. Formalizing cardinality-based feature models and their specialization. Software Process: Improvement and Practice. 2005 Jan/Mar; 10(1):7–29.
5. Riebisch M, Bollert K, Streitferdt D, Philippow I. Extending feature diagrams with UML multiplicities. Proceedings of the 6th World Conference on Integrated Design and Process Technology (IDPT2002); 2002 Jun.
6. Fenton N, Neil M, Marquez D. Using Bayesian networks to predict software defects and reliability. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability. 2008 Dec; 222(4):701–12.
7. Kang KC, Cohen SG, Hess J, Novak W, Peterson A. Feature-Oriented Domain Analysis (FODA) feasibility study. Technical Report. Carnegie-Mellon University: Software Engineering Institute; 1990 Nov.
8. Benavides D, Felfernig A, Jose A, Reinfrank F. Automated analysis in feature modelling and product configuration. Proceedings of 13th International Conference on Software Reuse, ICSR; Pisa. 2013 Jun. p. 160–75.
9. Benavides D, Segura S, Ruiz-Cortes A. Automated analysis of feature models 20 years later: A literature review. Information Systems. 2010 Sep; 35(6):615–36.
10. Czarnecki K, Bednasch T, Unger P, Eisenecker U. Generative programming for embedded software: An industrial experience report. Proceedings of the 1st ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering; 2002. p. 156–72.
11. Batory D, Benavides D, Ruiz-Cortes A. Automated analysis of feature models: Challenges ahead. Communications of the ACM. 2006 Dec; 49(12):45–7.
12. Trinidad P, Benavides D, Duran A, Ruiz-Cortez A, Toro M. Automated error analysis for the agilization of feature modeling. Journal of Systems and Software. 2008 Jun; 81(6):883–96.
13. White J, Benavides D, Schmidt DC, Trinidad P, Dougherty B, Ruiz-Cortes A. Automated diagnosis of feature model configurations. Journal of Systems and Software. 2010 Jul; 83(7):1094–107.
14. Massen T, Lichter H. Deficiencies in feature models. Workshop on Software Variability Management for Product Derivation- Towards Tool Support; 2004.
15. Ripon S, Hossain SJ, Azad K, Hassan M. Modeling and analysis of product line variants. Proceedings of SPLC; 2012 Sept. p. 26–31.
16. Rahman M, Ripon S. Using bayesian networks to model and analyze software product line feature model. Multi-disciplinary trends in artificial intelligence. Murty MN, Xiangjian H, Chillarige RR, editors. Lecture Notes in Computer Science. Springer International Pub; 2014. p. 220–31.
17. Elfaki A, Fong S, Vijayaprasad P, Johar M, Fadhil M. Using rule-based method for detecting anomalies in software product line. Research Journal of Applied Sciences, Engineering and Technology. 2014; 7(2):275–81.
18. Ripon S, Piash MM, Hossain SMA, Uddin MS. Modeling product line variants – semantic web approach. Lecture Notes on Software Engineering. 2013 Feb; 1(1):84–8.
19. Ripon S, Piash MM, Hossain SMA, Uddin MS. Semantic web based analysis of product line variant model. International Journal of Computer and Electrical Engineering. 2014 Feb; 6(1):1–6.
20. Rincon LF, Giraldo GL, Mazo R, Salinesi C. An ontological rule-based approach for analyzing dead and false optional features in feature models. Electronic Notes in Theoretical Computer Science. 2014 Feb; 302:111–132.