Verification of SPL Feature Tree by Using Bayesian Network

Supervised by

Dr. Shamim Hasnat Ripon
Chairperson and Associate Professor
Department of Computer Science and Engineering

Submitted by

Md. Javedul Ferdous

and

Md. Delwar Hossain

A Project submitted in partial fulfillment for the degree of B.Sc. in Computer Science and Engineering

In the

Faculty of Science and Engineering
Department of Computer Science and Engineering

East West University

May 2015

DECLARATION BY CANDIDATE

We hereby declare that the work presented in this thesis is, to the best of our knowledge and belief, original, except as acknowledged in the text and that the material has not been submitted, either in whole or in part, for a degree at this or any other university.

Md. Javedul Ferdous

_____

Md. Delwar Hossain

_____

Letter of Acceptance

I hereby declare that this thesis is from the student's own work and effort, and all other sources of information used have been acknowledged. This thesis has been submitted with my approval.

SUPERVISOR and CHAIRPERSON: Dr. Shamim Hasnat Ripon

SIGNATURE: _____

DATE: _____

Abstract

Products with new features need to be introduced on the market in a prompt step and organizations need to speed up their development process. Reuse has been suggested as a solution, but to achieve effective reuse within an organization a planned and pre-emptive effort must be used. Software Product lines are the most promising technique and it increases productivity and software quality and decreases time-to-market. In SPL, a feature tree shows various types of features and seizures the relationships among them. Bayesian Network is gaining much interest in Software Engineering, mainly in calculating software defects and software reliability. It can make notable consequence on feature analysis & its component. This thesis applies BN in modeling and analyzing features in a feature tree. Many feature analysis are modeled and verified in Bayesian Network. The verification of the rules define the analysis rules & its correctness. Finally a tool is used for reduced human error to make result more efficient & precise.

Acknowledgements

First of all, we like to thank Almighty Allah for give us strength & proper knowledge to complete our task and to submit our work as Thesis. We are really grateful to our honorable chairperson & our supervisor Dr.Shamim Hasnat Ripon, Chairperson and Associate Professor of Department of CSE, East West University for his guidance & direction. He showing us the path to perform up to my potential and produce a good research work. Without his help, it is not possible to complete our research work in due time.

To the Father of Computer, Charles Babbage

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays, systems are designed by composing existing components that have been used in other systems in engineering discipline. Software engineering has been more focused on original development but it is now recognized that to achieve better & flexible software, more quickly and at lower cost, we need to adopt a design process that is based on systematic Software Reuse [1]. It can be the whole of an application system either by incorporating it without change into other systems or by developing application families. Basically, Software reuse use to reduce the cost of software production by replacing creation with recycling [2].

Software engineers involves investments in requirements analysis, architecture and design, documentation etc. which has been pressured to introduce new products and add functionality to existing products at a rapid pace to be able to compete at the market[3]. Most organizations today usually derive new systems from previous instances to speed up the process. But to reuse similarities between systems in the most efficient way a product line approach might be the right answer to an organization. The methodology uses a common set of core resources to modify, assemble, instantiate, or generate multiple products and is devoted to as a product line. Such a product line method includes building a product line as a product family [4].

Feature Models are a common language to represent Product Line Models which is describe the features and their dependencies for creating valid products. One of the familiar issues of FMs is that they may have defects that can significantly reduce the benefits of the product line approach. Two of these defects are dead features and false optional features [5]. Dead features are features absent from any valid product of the product line. False optional features are features declared as optional but actually required in all valid products.

1.2 Objectives

The objective of this paper is to implement Bayesian Network to analyze the defects in SPL feature models. This paper focuses on two specific analysis operations: false optional and dead features of feature diagram which based on Cardinality based diagram [6, 7]. To reaching the objective of our paper, we define several analysis rules for false optional and dead features by using First Order Logic & represent these rules by using Bayesian Networks. At first, we define BNs graphs to represent the scenarios of the analysis rules. Then we calculate Cumulative node probability after node probability tables of the variable has defined. After getting the result, we use a Bayesian network tools for verification. The calculated results & verification results match our First Order Logic analysis rules.

1.3 Contribution

As we want to represent an uncertain domain, Bayesian networks are introduce the key computer technology for dealing with probabilities in AI, where the nodes represent variables and arcs represent direct connection between them [8,9,10,11]. Our focus is to represent Cardinality based feature model & verify using tools to reduce human error where we use MSBNX as a tools. MSBNX is a Microsoft Windows software application that supports the creation, manipulation and evaluation of Bayesian probability models [5].

1.4 Outline

The entire report divided into 6 Chapter. In Chapter 1, it gives a short review of Software product line, Reuse & Feature Constraint. Chapter 2 shows the feature tree and BN and an example of feature tree of a Bayesian network. Chapter 3 covers Cardinality & its component. Chapter4 defines First Order Logic based analysis rules of false optional and dead features. Meanwhile, Chapter 5 models the analysis rules of Bayesian network & its probabilistic calculation. Plus all rules are verified by tools in this Chapter. In conclusion, in Chapter 6 we conclude the report by summarizing our contributions and outlining our future plans.

Chapter 2

Background

2.1 Software product line:

A software product line is a standard of software-demanding systems that share a mutual, managed set of features satisfying the specific needs of a particular market segment or mission which are developed from a common set of core assets in a prescribed way [12]. Software product lines are emerging as a viable and important development paradigm allowing companies to realize order-of-magnitude improvements in time to market, cost, productivity, quality, and other business drivers. Software product line engineering can also allow fast market entry and flexible response, and provide a capability for mass customization [3, 4].

2.2 Feature Tree:

A Feature Tree is a classified diagram that visually shows the features of a solution in groups of increasing levels of detail. In the Feature Tree, some features may be flagged as mandatory, some optional, and some as mutually exclusive [13]. Features may be further flagged for development cycles, business priority, dependencies, or other relevant information. They are most commonly used for planning the overall feature set of a single solution or product that will be evolved over time or for defining the differing features that will be included in a product line [14, 15].

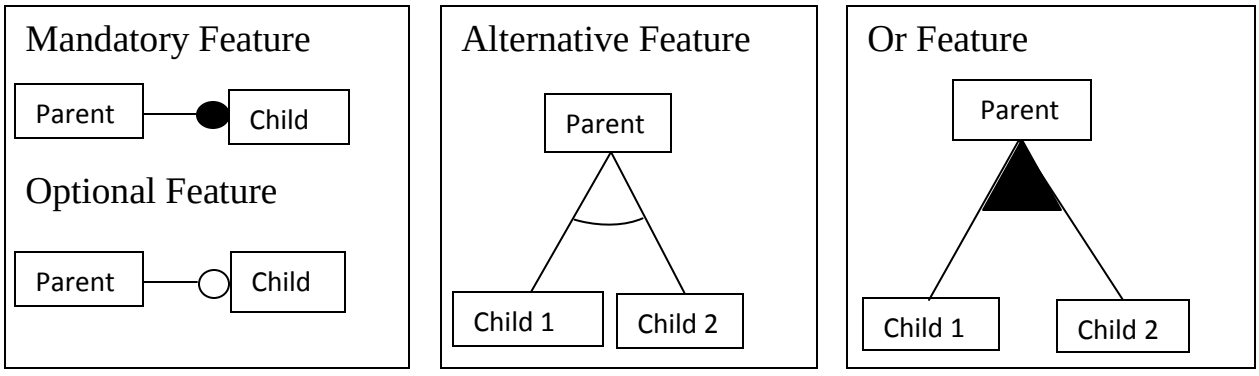


Figure 2.1: Feature of Feature Tree

2.3 Bayesian Network

Bayesian Network is one kind graphical representation which using cumulative probability distribution to make set of graph & make a relation with each node by arrow Assuming discrete variables, the strength of the relationship between variables is quantified by conditional probability distributions associated with each node [6]. The only constraint on the arcs allowed in a BN is that there must not be any directed cycles. In addition, BNs model the quantitative strength of the connections between variables, allowing probabilistic beliefs about them to be updated automatically as new information becomes available [8,9,10,11].

Example of Bayesian Network

A Bayesian network represents a set of random variables and their conditional dependencies via a graph. The nodes in a Bayesian network represent a set of random variables,
$X = \mathrm{X}_1 \ldots \ldots \mathrm{X}_i \ldots \ldots \mathrm{X}_n$ from the domain. A set of directed arcs (or links) connects pairs of nodes, $X_i \rightarrow \mathrm{X}_j$, representing the direct dependencies between variables [10].

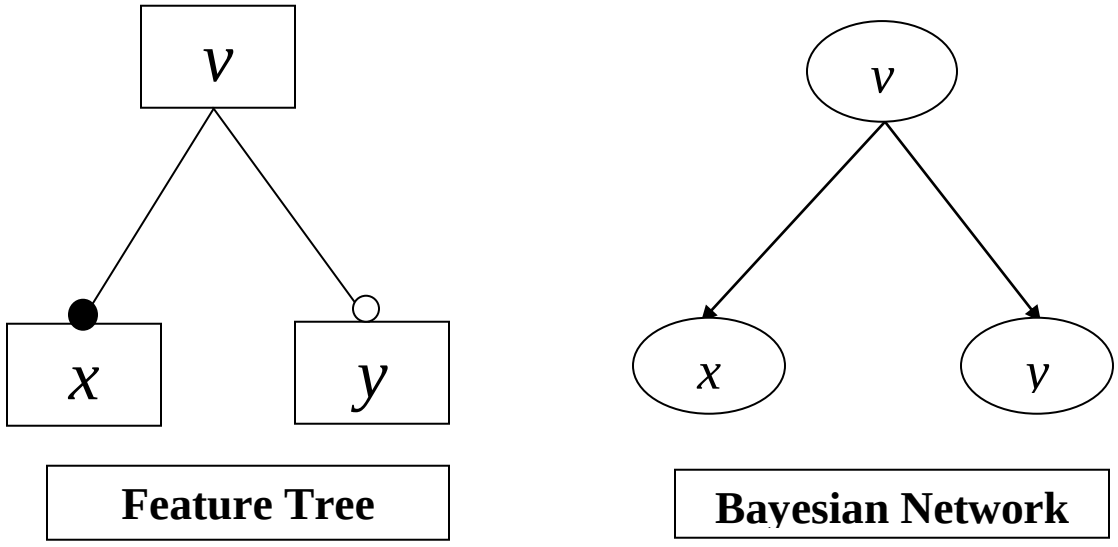


Figure 2.2: BN Representation of a Feature Tree

Figure 2.2 illustrates a Bayesian network. Its set of edges is$\mathrm{E} = \{(\mathrm{v}, \mathrm{x}), (\mathrm{v}, \mathrm{y})\}$. This constitutes a DAG because there are no undirected edges and there are no cycles. Furthermore, since x and y are conditionally independent of each other, so we can following: $P(x|v, y) = P(x|v)$,which means that, for the factorization represented by the Bayesian Network, the probability of x is conditioned only v and that the value of c is irrelevant for this local probability. Likewise, we can say $P(y|x, v) = P(y|v)$. The edges in the Bayesian network encode a particular factorization of the joint distribution [9]. In this example, the joint distribution of all variables, as factorized by this Bayesian network is,

$$P(x, v, y) = P(x|v)P(v)P(y|v)$$

In general, given nodes $A = A_1 \ldots\ldots, A_n$ the joint probability function for any Bayesian Network is

$$P(A) = \prod_{i=1}^{n} P(A_i|Parents(A_i))$$

Chapter 3

Cardinality Based SPL

3.1 Feature Model

A feature model is organized a set of features and makes a relationships between a parent feature and its child features which are important distinguishing characteristics, qualities, or features of a family of systems [6]. They are making a graphical illustration of a particular system using parent-child relation, rending the shared structure and behavior of a set of similar systems. All the various features of a set of similar/related systems are needed to compose into a feature model [15]. These entire features are grouped with cardinality where Cardinality can be defined with lower bound & upper bound. Lower bound indicates the minimum feature that can be selected within group cardinality. On the Contrast, Upper bound indicates the maximum feature that can be selected within group cardinality.

3.2 Cardinality

The concept of group cardinality was generalized in as a set of features annotated with a cardinality specifying an interval of how many features can be selected from that set [16 a feature cardinality specification may consist of a sequence of intervals [18].Features can be annotated with cardinalities, such as <1. ∗> or <3, 3>. Mandatory and optional features can be considered special cases of features with the cardinalities <1, 1> and <0, 1>, respectively. Feature cardinalities were motivated by a practical application. Furthermore, group cardinality is a property of the relationship between a parent and a set of sub features. Basically, Cardinality-based feature modeling integrates a number of extensions to the original Feature-Oriented Domain Analysis notation [14].

A cardinality-based feature model is an order of features, where each feature has feature cardinality. A feature cardinality is an interval of the form $<m, n>$, where $m \in \mathbb{Z} \wedge n \in \mathbb{Z} \cup \{*\} \wedge 0 \leq m \wedge (m \leq n \vee n = *)$ [17].More precisely, a feature cardinality is attached to the relationship between a solitary feature and its parent [14, 17]. Note that we allow a feature cardinality to have as an upper bound the Kleene star *. Such an upper bound denotes the possibility to take a feature an unbounded number of times. An example of a valid specification of a feature cardinality with more than one interval is [0...2], [6...6], which says that we can take a feature 0, 1, 2 or 6 times. Note that we allow the last interval in a feature cardinality to have as an upper bound the Kleene star $*$. Such an upper bound denotes the possibility to take a feature an unbounded number of times. For example, the feature cardinality [1...2], [7...$*$] requires that the associated feature is taken 1, 2, 5, or any number greater than 7 times. Semantically, the feature cardinality ε is equivalent to [0...0] and implies that the sub feature can never be chosen in a configuration. Additionally, features can be arranged into feature groups, where each feature group has group cardinality. Group cardinality <m, n>of a group is an interval with n as lower bound and n′ as upper bound. Given that k >0 is the number of grouped features in the group, we assume that the following invariant holds: $0 \leq n \leq n' \leq k$. Group cardinality denotes how many group members can be selected. For example, at least and at most one of the features Credit Card, Debit Card, and Purchase Order must be selected as a sub feature of Payment Type [14].

3.3 Example of Case Study

We used Graph Product Line as running example because it is well-known in the product line community, and it was proposed to be a standard case for evaluating product line methodologies. We used 9 artificial features to produce these defects. We identified artificial features with a capital AF. In addition, we identified original features of the model with their names. The members of the GPL are graphs either Directed or Undirected, their edges are Weighted or Unweighted, and their search algorithms are breadth-first search (BFS) or depth-first search (DFS).

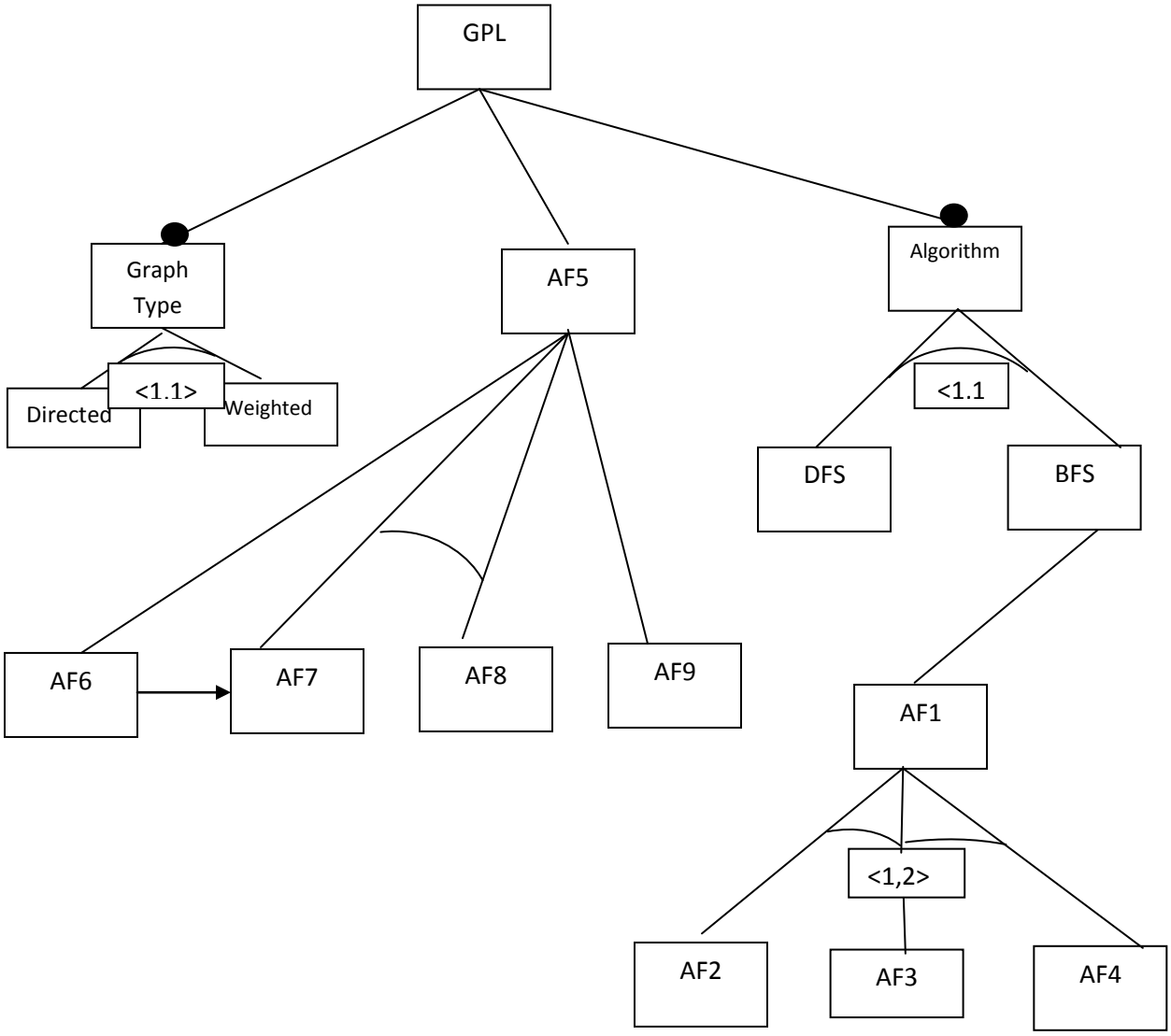


Figure 3.1: Graph Product Line

Chapter 4

Feature Analyses

4.1 First Order Logic Predicates

To implement our earlier work on logical representation of feature model and their analysis rules, we represent in this section a set of rules for dead and false optional features based on group of Cardinality [6, 7, 13].We use the following First Order Logic (FOL) predicates to define the rules.

- ✓ *Variation point (v): This predicate indicates that feature v is a variation Point, i.e., feature v has child feature(s).*
- ✓ *Mandatory variant (v, x): This predicate indicates that feature x is a mandatory Feature of feature v, i.e., x is a mandatory child of v.*
- ✓ *Optional variant (v, x): Here, x is an optional feature of v.*
- ✓ *Requires(x, y): It indicates that feature x requires feature y.*
- ✓ *Exclude(x,y): This predicate indicates that feature x and y are mutually exclusive.*
- ✓ *Select(x): Variant x is selected in the configuration.*

4.2 False Optional Feature

Rule 1

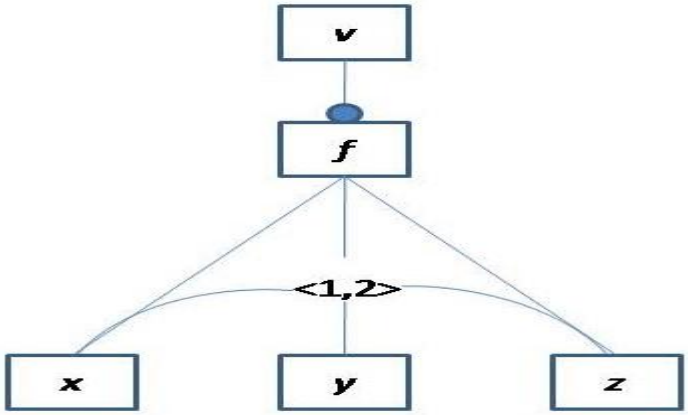


Figure 4.2.1: Feature Tree for FOF Rule 1

First Order Logic:

$\forall v,f,x,y,z.variation\_point(v) \wedge mandatory\_variant \wedge variation\_point(f)$^optional$_variant($
$x)$^optional$_variant(y)$^optional$_variant(z)$^$dead\_feature(x) \wedge dead\_feature(y) \wedge select(f)$
$\Rightarrow \neg select(x) \wedge \neg select(y) \wedge select(z).$

Rule 2

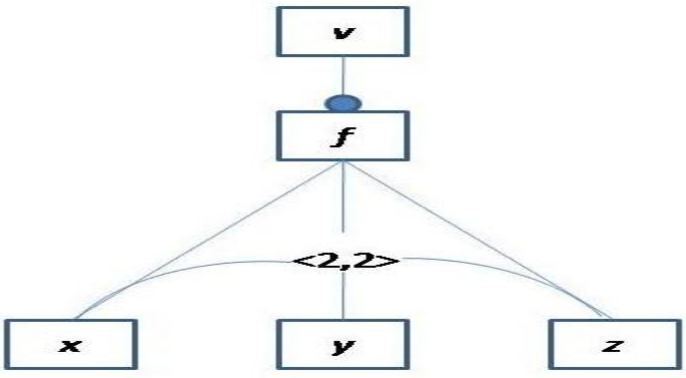


Figure 4.2.2: Feature Tree for FOF Rule 2

First Order Logic:

$\forall v,f,x,y,z.variation\_point(v) \wedge mandatory\_variant(f) \wedge variation\_point(f) \wedge optional\_variant(x) \wedge optional\_variant(y) \wedge optional\_variant(z) \wedge select(f) \Rightarrow \neg select(x) \wedge select(y) \wedge select(z)$

Rule 3

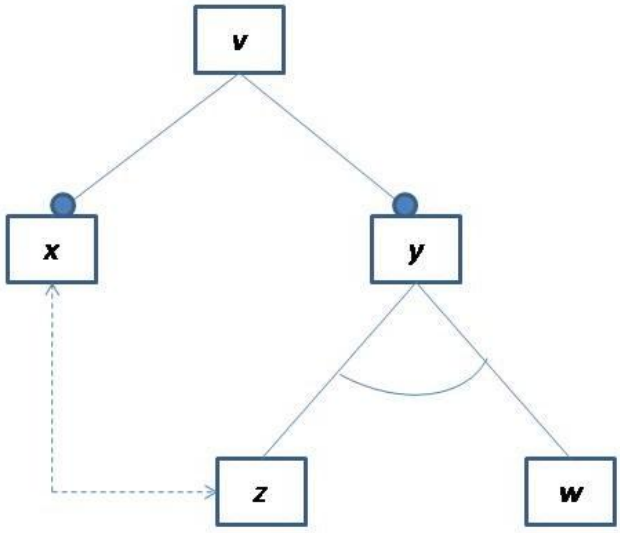


Figure 4.2.3: Feature Tree for FOF Rule 3

First Order Logic:

$\forall v,x,y,z.variation\_point(v) \wedge mandatory\_variant(x) \wedge mandatory\_variant \wedge variation\_point(y) \wedge optional\_variant(z) \wedge optional\_variant(w) \wedge exclude(x,z) \wedge select(x) \Rightarrow \neg select(z) \wedge select(w).$

4.3 Dead Feature

Rule 1

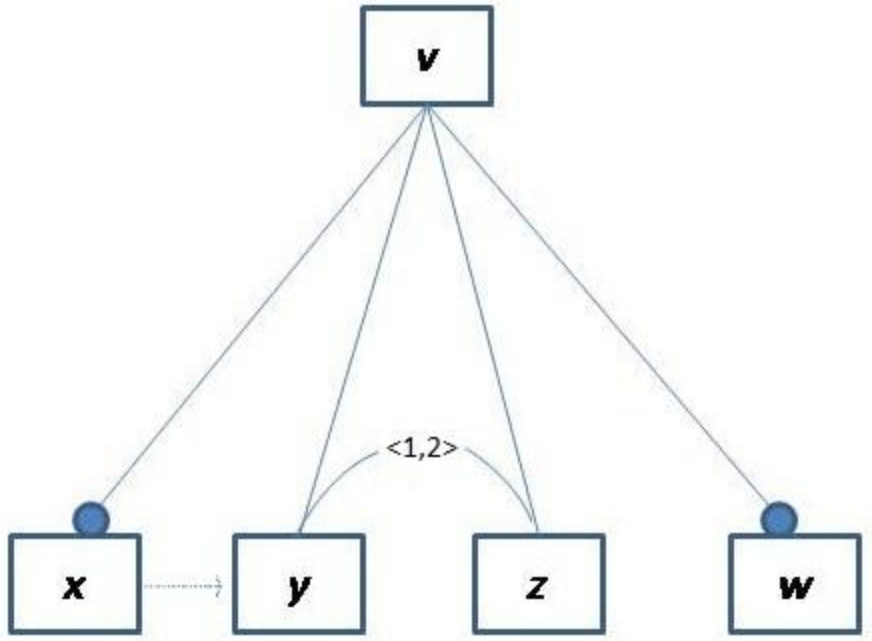


Figure 4.3.1: Feature Tree for DF Rule 1

First Order Logic:

$\forall v,f,x,y,z.variation\_point(v) \wedge mandatory\_variant(x) \wedge mandatory\_variant(w)$
$\wedge optional\_variant(y) \wedge optional\_variant(z) \wedge requires(x,y)$
$\wedge select(x) \Rightarrow select(y) \wedge select(w) \neg select(z).$

Rule 2

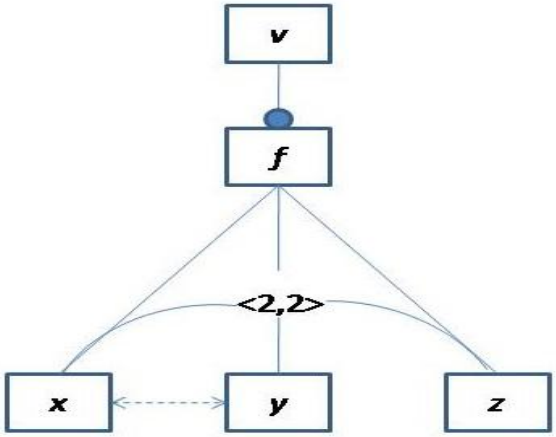


Figure 4.3.2: Feature Tree for DF Rule 2

First Order Logic:

$\forall v,f,x,y,z.variation\_point(v) \wedge mandatory\_variant(f) \wedge optional\_variant(x) \wedge optional\_variant(z) \wedge deadFeature(y) \wedge exclude(x,y) \wedge select(f) \Rightarrow select(x) \wedge \neg select(y) \wedge select(z)$.

Chapter 5

Bayesian Representation

5.1 Bayesian Modeling

After analyzing the scenarios for false optional and dead features, we define Bayesian Network representation of the scenarios. The key factor in defining the BN presentation is to identify the dependencies among the features. Our BNs are influenced by our logical representation shown I figure.

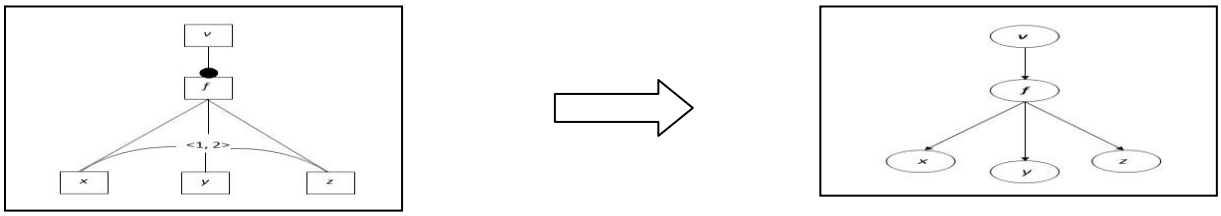
5.1.1 False Optional Feature:

An optional feature that constantly belongs to a mandatory feature is known a False Optional Feature [6].

Rule 1:

*False Optional, Rule 1:*One or more feature(s) become(s) false optional when it is (they are) grouped by a group cardinality with (with a mandatory father) having lower bound of "*m*" and upper bound of "*n*" & *m≠n* and [*k-n*] dead feature within the cardinality where "*k*" is the total number of features inside the group cardinality.

In the BN above, feature *x, y, z* are three variants of the variation point "*f*" and feature "*f*" is a variant of the variation point *v*. Thus the probability of features *x, y, z* being selected are directly dependent on whether *v* and *f* is selected or not. Additionally *f* is dependent on v.



Feature Tree Bayesian Network

Figure 5.1.1.1: Bayesian Network representation of FOF 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

Figure 5.1.1.2: NPT for False Optional Feature rule 1

From figure, based on the lower bound and 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.

Combination 1	*x*
Combination 2	*y*
Combination 3	*z*
Combination 4	*xy*
Combination 5	*yz*
Combination 6	*zx*

For combination 1,

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

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

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

$$= \frac{P(x=T,f=T,v=T)+P(x=T,f=T,v=F)}{P(f=T,v=T)+P(f=T,v=F)}$$

$$\therefore P(x = T, f = T, v = T) = P(x = T|f = T)P(f = T|v = T)P(v = T)$$

= 0* 1 * 1 = 0

$\therefore P(x = T, f = T, v = F) = P(x = T|f = T)P(f = T|v = F)P(v = f)$

=0 * 0 * 0 = 0

$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$

= 1 * 1 = 1

$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$

= 0 * 0 = 0

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

Thus combination 1 is an invalid combination for this scenario.

For combination 2

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

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

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

$$= \frac{P(y=T,f=T,v=T)+P(y=T,f=T,v=F)}{P(f=T,v=T)+P(f=T,v=F)}$$

$\therefore P(y = T, f = T, v = T) = P(y = T|f = T)P(f = T|v = T)P(v = T)$

= 0* 1 * 1 = 0

$$\therefore P(y = T, f = T, v = F) = P(y = T|f = T)P(f = T|v = F)P(v = f)$$

$$= 0 * 0 * 0 = 0$$

$$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$$

$$= 0 * 0 = 0$$

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

Thus combination 2 is an invalid combination for this scenario.

For Combination 3

$$\mathrm{P}(z, f) = P(z|f)P(f|v)P(v)$$

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

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

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

$$\therefore P(z = T, f = T, v = T) = P(z = T|f = T)P(f = T|v = T)P(v = T)$$

$$= 1 * 1 * 1 = 1$$

$$\therefore P(z = T, f = T, v = F) = P(z = T|f = T)P(f = T|v = F)P(v = f)$$

$$=1 * 0 * 0 = 0$$

$$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$$

$$= 0 * 0 = 0$$

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

Thus combination 3 is a valid combination for this scenario.

For combination 4

$$P(x, y, f) = P(x, y|f)P(f|v)P(v)$$

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

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

$$= \frac{P(x=T,y=T,f=T,v=T)+P(x=T,y=T,f=T,v=F)}{P(f=T,v=T)+P(f=T,v=F)}$$

$$\therefore P(x = T, y = T, f = T, v = T) = P(x = T|f = T)$$

$$P(y = T|f = T)\ P(f = T|v = T)P(v = T)$$

$$= 0 * 0 * 1 = 0$$

$$\therefore P(x = T, y = T, f = T, v = F) = P(x = T|f = T)$$

$$P(y = T|f = T)P(f = T|v = F)P(v = f)$$

$$= 0 * 0 * 0 = 0$$

$$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$$

$$= 0 * 0 = 0$$

$$P(x = T, y = T|f = T) = \frac{0 + 0}{1 + 0} = 0$$

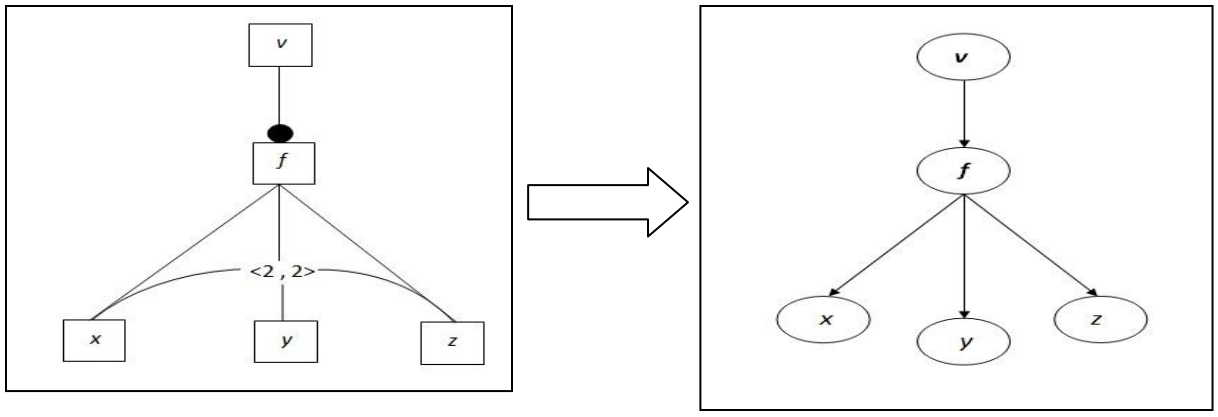
Thus combination 4 is an invalid combination for this scenario.

Similarly it can be shown that combination 5 and combination 6 are also invalid for this scenario making "*z"* a false optional feature.

Rule 2:

False Optional, Rule 2:One or more feature(s) becomes false optional when it is (they are) grouped by a group cardinality with (with a mandatory father) having lower bound of *"m"* is equal or greater than upper bound of *"n"* and [*k-m*] dead feature within the cardinality where "k" is the total number of features inside the group cardinality.

In the Bayesian Network above, feature *x,y,z* are three variants of the variation point *"f"* and feature *"f"* is a variant of the variation point *"v"*. Thus the probability of features *x,y,z* being selected are directly dependent on whether *"v"* and *"f"* is selected or not. Then "f" is dependent on *"v"*.



Feature Tree Bayesian Network

Figure 5.1.1.3: Bayesian Network representation of FOF Rule 2

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	1	0
F	0	1

z

f	T	F
T	1	0
F	0	1

Figure 5.1.1.4: NPT for False Optional Feature rule 2

From figure, the lower bound and upper bound of the group cardinality after *f* is selected. There is $3c_2 = 3$ possible combination of feature from the group cardinality.

Combination 1	*xy*
Combination 2	*yz*
Combination 3	*zx*

For combination 1:

$$P(x, y, f) = P(x, y|f)P(f|v)P(v)$$

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

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

$$= \frac{P(x{=}T, y{=}T, f{=}T, v{=}T) + P(x{=}T, y{=}T, f{=}T, v{=}F)}{P(f{=}T, v{=}T) + P(f{=}T, v{=}F)}$$

$$\therefore P(x = T, y = T, f = T, v = T)$$
$$= P(x = T|f = T)P(y = T|f = T)P(f = T|v = T)P(v = T)$$

= 0 * 1 * 1 = 0

$$\therefore P(x = T, y = T, f = T, v = F)$$
$$= P(x = T|f = T)P(y = T|f = T)P(f = T|v = F)P(v = f)$$

$$= 0 * 1 * 0*0 = 0$$

$$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$$

$$= 0 * 0 = 0$$

$$P(x = T, y = T|f = T) = \frac{0 + 0}{1 + 0} = 0$$

Thus combination 1 is an invalid combination for this scenario.

For combination 2:

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

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

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

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

$$\therefore P(y = T, z = T, f = T, v = T) = P(y = T|f = T)P(z = T|f = T)P(f = T|v = T)P(v = T)$$

$$= 1 * 1 * 1 = 1$$

$$\therefore P(y = T, z = T, f = T, v = F) = P(y = T|f = T)P(z = T|f = T)P(f = T|v = F)P(v = f)$$

$$=1* 1 * 0*0 = 0$$

$$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$$

$$= 0 * 0 = 0$$

$$p(y = T, z = T|f = T) = \frac{1 + 0}{1 + 0} = 1$$

Thus combination 2 is a valid combination for this scenario.

For combination 3:

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

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

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

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

$$\therefore P(z = T, x = T, f = T, v = T) = P(z = T|f = T)P(x = T|f = T)P(f = T|v = T)P(v = T)$$

$$= 1*0*1 * 1 = 0$$

$$\therefore P(z = T, x = T, f = T, v = F) = P(z = T|f = T)P(x = T|f = T)P(f = T|v = F)P(v = f)$$

$$= 1 * 0 * 0 * 0 = 0$$

$$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$$

$$= 0 * 0 = 0$$

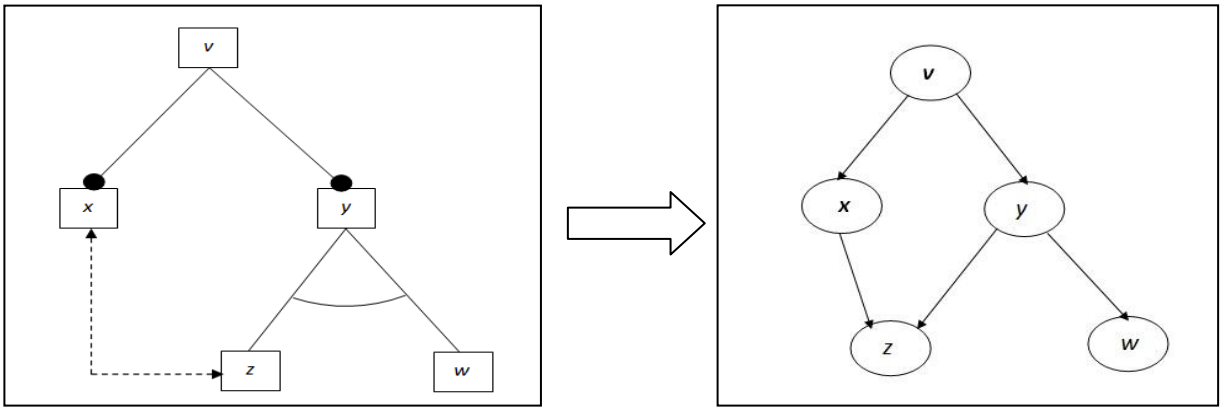
$$P(z = T, x = T | f = T) = \frac{0 + 0}{1 + 0} = 0$$

Thus combination 3 is an invalid combination for this scenario.

Rule 3:

False Optional, Rule 3: An optional feature becomes false optional when it is connected with a mandatory father by alternative relation. In the following graph *x* and *y* are mandatory feature. Feature *x* exclude feature *z* which is alternate with feature *w*. Feature *z* will be dead feature for exclude.

In alternate relation, only one feature can be selected. So *w* will be false optional. In the Bayesian Network above, both *x* and *y* are two variants of the variation point *v*. *z* is a variant of variation point *x* and *y*. w also a variant of variation point y.



Feature Tree Bayesian Network

Figure 5.1.1.5: Bayesian Network representation of FOF Rule 3

The NPT of these features are given below

v

T	F
1	0

x

v	T	F
T	1	0
F	0	1

y

v	T	F
T	1	0
F	0	1

z

x	*y*	T	F
T	T	0	1
T	F	I	I
F	T	I	I
F	F	0	1

W

y	T	F
T	1	0
F	0	1

Figure 5.1.1.6: NPT for False Optional Feature rule 3

For feature Z,

$$P(z,x,y) = P(z|x,y)P(x|v)P(y|v)P(v)$$

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

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

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

$$\therefore P(z=T,x=T,y=T,v=T)$$
$$= P(z=T|x=T,y=T)P(x=T|v=T)P(y=T|v=T)P(v=T)$$
$$= 0*1*1*1 = 0$$

$$\therefore P(z=T,x=T,y=T,v=F)$$
$$= P(z=T|x=T,y=T)P(x=T|v=F)P(y=T|v=F)P(v=F)$$

$$= 0 * 0 * 0 * 0 = 0$$

$$\therefore P(x = T, y = T, v = T) = P(x = T|v = T)P(y = T|v = T)P(v = T)$$

$$= 1 * 1 * 1 = 1$$

$$\therefore P(x = T, y = T, v = F) = P(x = T|v = F)P(y = T|v = F)P(v = F)$$

$$= 0 * 0 * 0 = 0$$

$$P(z = T|x = T, y = T) = \frac{0 + 0}{1 + 0} = 0$$

Thus z is an invalid combination for this scenario.

For feature W,

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

$$P(w = T|y = T) = \frac{P(w = T, y = T)}{P(y = T)}$$

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

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

$$\therefore P(w = T, y = T, v = T) = P(w = T|y = T)P(y = T|v = T)P(v = T)$$

$$= 1 * 1 * 1 = 1$$

$$\therefore P(w = T, y = T, v = F) = P(w = T|y = T)P(y = T|v = F)P(v = F)$$

$$= 1 * 0 * 0 = 0$$

$$\therefore P(y = T, v = T) = P(y = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(y = T, v = F) = P(y = T|v = F)P(v = F)$$

$$= 0 * 0 = 0$$

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

Thus w is a valid combination for this scenario

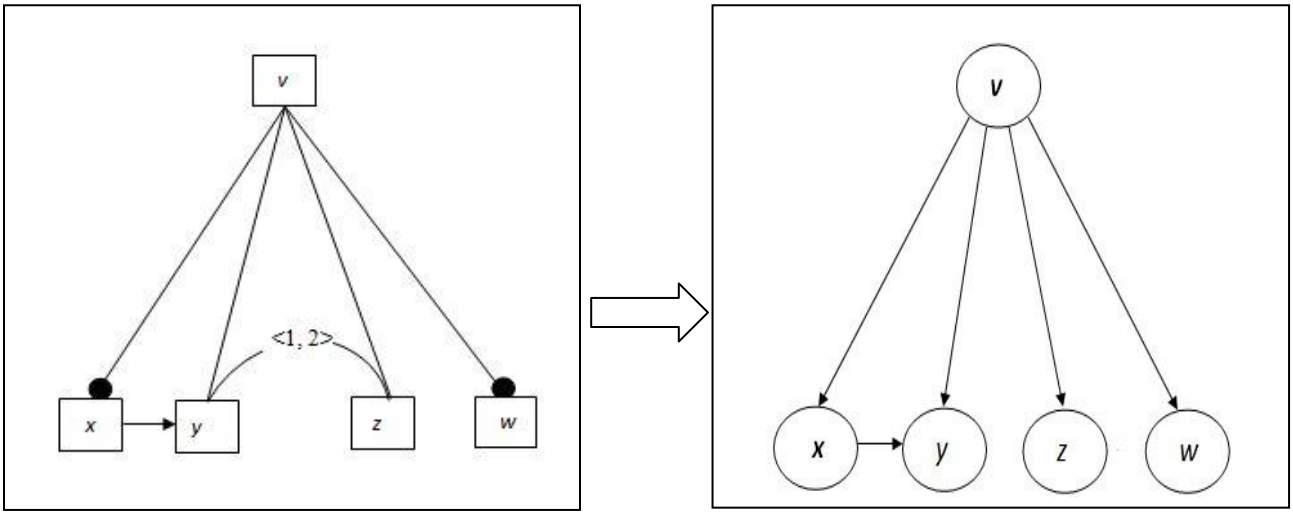
5.1.2 Dead Feature:

An optional feature that never gets selected in a valid product is known a Dead Feature [6].

Rule 1:

Dead Feature, Rule 1: An optional feature becomes dead if it belongs to group cardinality and the number of false optional feature is equal to the cardinality upper bound. In the following graph feature *y* and *z* connected with <1,1> group cardinality. *x* is a mandatory feature which requires *y* where *y* is a optional feature. This results into *y* being a false optional feature. Since upper bound is 1, *y* alone can be selected. Thus z is dead feature.

In the Bayesian network above , feature *x,y,z,w* are the four variant of the variation point v.Thus the probability of these variant feature being selected are directky depend on whether *v* is selected or not. Then *x* is only dependent on *v* but the probability of *y* being selected is also dependent on whether *x* is selected or not.Also *w* is only dependent on *v* . But *z* is dependent on whether *z* is selected or not.Hense both *y* and *z* are simultaneously dependent on (*v,x*) and (*v,w*).



Feature Tree Bayesian Network

Figure 5.1.2.1: Bayesian Network representation of DF Rule 1

The NPT are given below

v

T	F
1	0

X

v	T	F
T	1	0
F	0	1

y

v	X	T	F
T	T	1	0
T	F	I	I
F	T	I	I
F	F	0	1

z

v	T	F
T	0	1
F	0	1

w

v	T	F
T	1	0
F	0	1

Figure 5.1.2.1: NPT for dead feature rule 1

For Feature Y :

$$P(y,x) = P(y|x)P(f|v)P(v)$$

$$P(y = T|x = T) = \frac{P(y = T, x = T)}{P(x = T)}$$

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

$$=\frac{P(y=T,x=T,v=T)+P(y=T,x=T,v=F)}{P(x=T,v=T)+P(x=T,v=F)}$$

$\therefore P(y=T,x=T,v=T)=P(y=T|x=T)P(x=T|v=T)P(v=T)$

$= 1 * 1 * 1 = 1$

$\therefore P(y=T,x=T,v=F)=P(y=T|x=T)P(x=T|v=F)P(v=f)$

$=1 * 0 * 0 = 0$

$\therefore P(x=T,v=T)=P(x=T|v=T)P(v=T)$

$= 1 * 1 = 1$

$\therefore P(x=T,v=F)=P(x=T|v=f)P(v=f)$

$= 0 * 0 = 0$

$$P(y=T|x=T)=\frac{1+0}{1+0}=1$$

Thus y is a valid combination for this scenario.

For feature *Z*:

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

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

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

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

$\therefore P(z=T,w=T,v=T)=P(z=T|w=T)P(w=T|v=T)P(v=T)$

$= 0* 1 * 0 = 0$

$$\therefore P(z = T, w = T, v = F) = P(z = T|w = T)P(w = T|v = F)P(v = f)$$

$$= \text{I} * 0 * 0 = 0$$

$$\therefore P(w = T, v = T) = P(w = T|v = T)P(v = T)$$

$$= 0 * 0 = 0$$

$$\therefore P(w = T, v = F) = P(w = T|v = f)P(v = f)$$

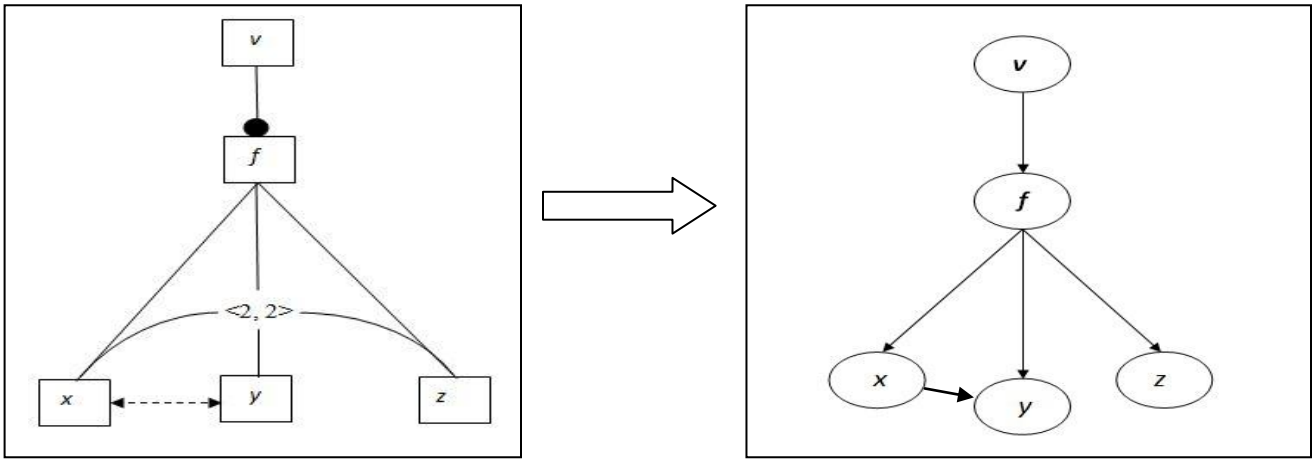
$$= 0 * 0 = 0$$

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

Thus *z* is an invalid combination for this scenario.

Rule 2:

Dead Feature, *Rule 2:* A feature within group cardinality becomes a dead feature when it is excluded by another feature inside the cardinality. In the Bayesian Network above, feature *x,y,z* are three variants of the variation point "*f*" and feature "*f*" is a variant of the variation point "*v*". Probability of *x* and *z* are directly dependent on *f* is selected or not but *y* is dependent on both *x* and *f*. The NPT of these feature are given below.



Feature Tree Bayesian Network

Figure 5.1.2.3: Bayesian Network representation of DF Rule 1

v

T	F
1	0

f

v	T	F
T	1	0
F	0	1

x

f	T	F
T	1	0
F	0	1

y

f	x	T	F
T	T	0	1
T	F	I	I
F	T	I	I
F	F	0	1

Z

f	T	F
T	1	0
F	0	1

Figure 5.1.2.4: NPT for dead feature rule 2

From figure, the lower bound and upper bound of the group cardinality after f is selected. There is $3c_2 = 3$ possible combination of feature from the group cardinality.

Combination 1	*xy*
Combination 2	*yz*
Combination 3	*zx*

For feature *x*,

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

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

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

$$= \frac{P(x{=}T, f{=}T, v{=}T) + P(x{=}T, f{=}T, v{=}F)}{P(f{=}T, v{=}T) + P(f{=}T, v{=}F)}$$

$\therefore P(x = T, f = T, v = T) = P(x = T|f = T)P(f = T|v = T)P(v = T)$

$$= 1^* 1 * 1 = 1$$

$$\therefore P(x = T, f = T, v = F) = P(x = T|f = T)P(f = T|v = F)P(v = f)$$

$$=1 * 0 * 0 = 0$$

$$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$$

$$= 0 * 0 = 0$$

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

Thus *x* is an invalid combination for this scenario.

For feature *y*,

$$P(y, x, f) = P(y|x, f)P(x|f)P(f|v)P(v)$$

$$P(y = T|x = T) = \frac{P(y = T, x = T)}{P(x = T)}$$

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

$$= \frac{P(y=T,x=T,f=T,v=T)+P(y=T,x=T,f=F,v=F)}{P(x=T,f=T,v=T)+P(x=T,f=F,v=F)}$$

$$\therefore P(y = T, x = T, f = T, v = T) = P(y = T|x = T, f = T)$$

$P(x = T|f = T)P(f = T|v = T)P(v = T)$

= 0* 1 * 1*1 = 0

$\therefore P(y = T, x = T, f = F, v = F) = P(y = T|x = T, f = F)$

$P(x = T|f = F)P(f = F|v = F)P(v = F)$

= I *0 * 0 * 0 = 0

$\therefore P(x = T, f = T, v = T) = P(x = T|f = T)P(f = T|v = T)P(v = T)$

= 1* 1 * 1 = 1

$\therefore P(x = T, f = F, v = F) = P(x = T|f = F)P(f = T|v = f)P(v = f)$

= 0*1* 0 = 0

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

Thus *y* is a valid combination for this scenario.

For Combination 3

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

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

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

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

$\therefore P(z = T, f = T, v = T) = P(z = T|f = T)P(f = T|v = T)P(v = T)$

$$= 1 * 1 * 1 = 1$$

$$\therefore P(z = T, f = T, v = F) = P(z = T|f = T)P(f = T|v = F)P(v = f)$$

$$=1 * 0 * 0 = 0$$

$$\therefore P(f = T, v = T) = P(f = T|v = T)P(v = T)$$

$$= 1 * 1 = 1$$

$$\therefore P(f = T, v = F) = P(f = T|v = f)P(v = f)$$

$$= 0 * 0 = 0$$

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

Thus *z* is an invalid combination for this scenario.

Now for combination 1

$$xy = P(x = T|f = T) * P(y = T|x = T)$$

$$= 1 * 0 = 0$$

Feature *xy* is not selected.

Now for combination 2

$$yz = P(y = T|x = T) * P(z = T|x = T)$$

$$= 0 * 1 = 0$$

Feature *yz* is not selected.

Now for combination 3

$$zx = P(z = T|f = T) * P(x = T|f = T)$$

$= 1 * 1 = 1$

Feature *zx* is selected.

Thus, *y* becomes a dead feature.

5.2 Tool Representation

To represent the vertices, or *nodes*, the variables and the edges or *arcs*, the conditional dependencies we use MSBNX as a tools. MSBNX is a Microsoft Windows software application that supports the creation, manipulation and evaluation of Bayesian probability models one of the primary roles of a Bayesian model is to allow the model creator to use commonsense and real-world knowledge to eliminate needless complexity in the model. The method used to remove meaningless relationships in a Bayesian model is to explicitly declare the meaningful ones.

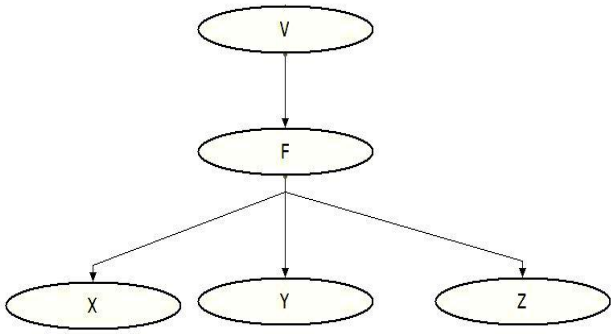


Figure 5.2.1: FOF Rule 1

These influences are represented by *conditioning arcs* between nodes. Each arc should represent a causal relationship between temporal antecedents. Basically it is a graphical representation of Bayesian Networks.We injects Boolean '0' and ‘1’ to each node regarding their dependency on vertices to vertices&represent truth table a below.

V		
Yes	No	bar charts
1.0	0.0	

Parent Node(s)	F		
V	Yes	No	bar charts
Yes	1.0	0.0	
No	0.0	1.0	

Parent Node(s)	X		
F	Yes	No	bar charts
Yes	0.0	1.0	
No	0.0	1.0	

Parent Node(s)	Y		
F	Yes	No	bar charts
Yes	0.0	1.0	
No	0.0	1.0	

Parent Node(s)	Z		
F	Yes	No	bar charts
Yes	1.0	0.0	
No	0.0	1.0	

Figure 5.2.2: NPT for FOF Rule 1

After setting the value on to the nodes , we take it to the "Inference Evaluation" for execution.

Then it shows a graphical representation of whole model.

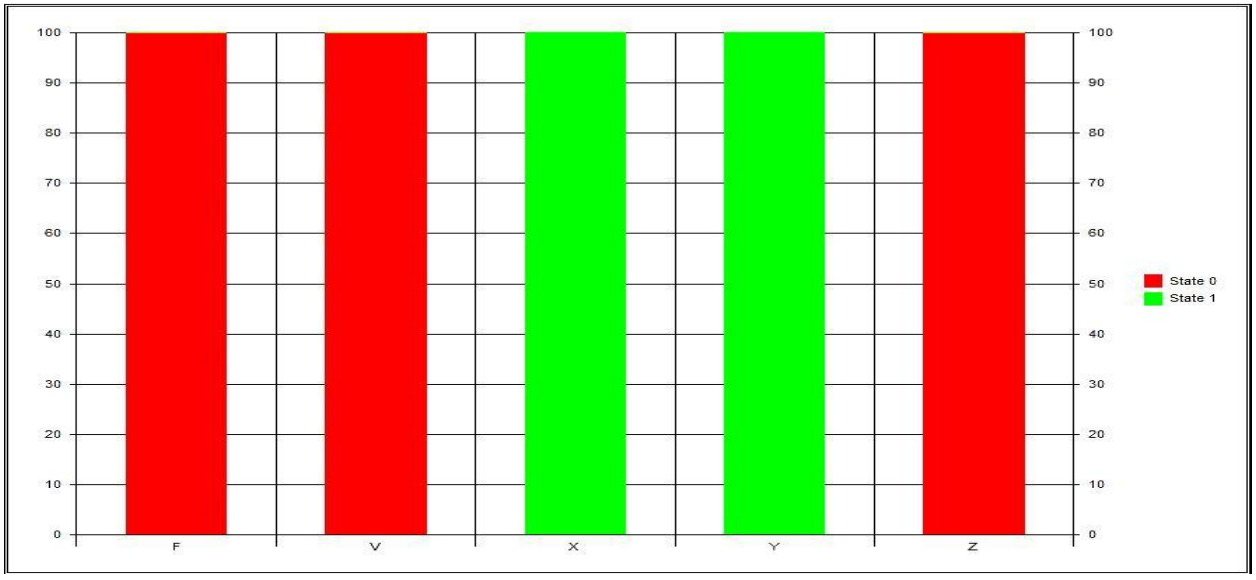


Figure 5.2.3: Bar Chart for FOF Rule 1

Figure shows that if *v* is varient of full manadory feature *f* , then *z* will be false optional feature where *x*&*y* feature are dead feature. We represent our all rules by following procedure & get the result which is equal to the result of Bayesian Networks.

Inconsistency: A model or system can be say that iit's consist inconsistency if it has any kind of anomaies that it shouldn't occur.To represent all rules in tools we found some inconsistency where by following procedure ,

Suppose, we are considering a scenario where *v* is root, *f* is the decender of *v*&*z* is the decender of *f*.If *z* is selected then f has to selested as well as *v*

We can get two kind of situation.

1) *z* is selected but *f* is not selected
2) *z* is selected but *v* is not selected

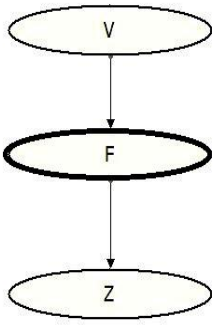


Figure: Situation 1

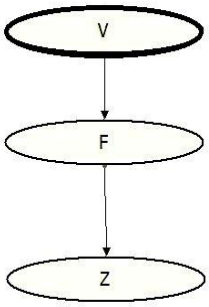


Figure: Situation 2

Figure 5.2.4: Inconsistency

If either one of this occur on the model then it will face inconsistency on the model. Basically inconsistency is one kind of anomalies in system that are rarely happen .

As usual we represent inconsistencyin the way we represent following rules.

Let's a have demonstration in *Dead Feature, Rule 2.* First, we take the portion of model where inconsistency exists.

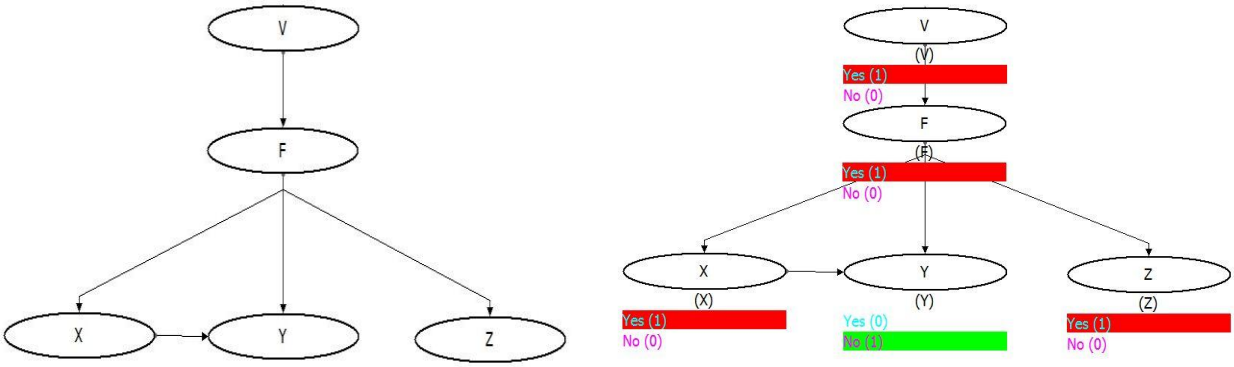


Figure 5.2.5: Bayesian representation in tools

Then, set the truth value regarding dependency on each node and set YES to '0' & NO to '1' for in *y* nodes for inconsistency.

When we evaluate the graphical representation of the output for Bar chart, it shows itswhich node is selected having inconsistency.

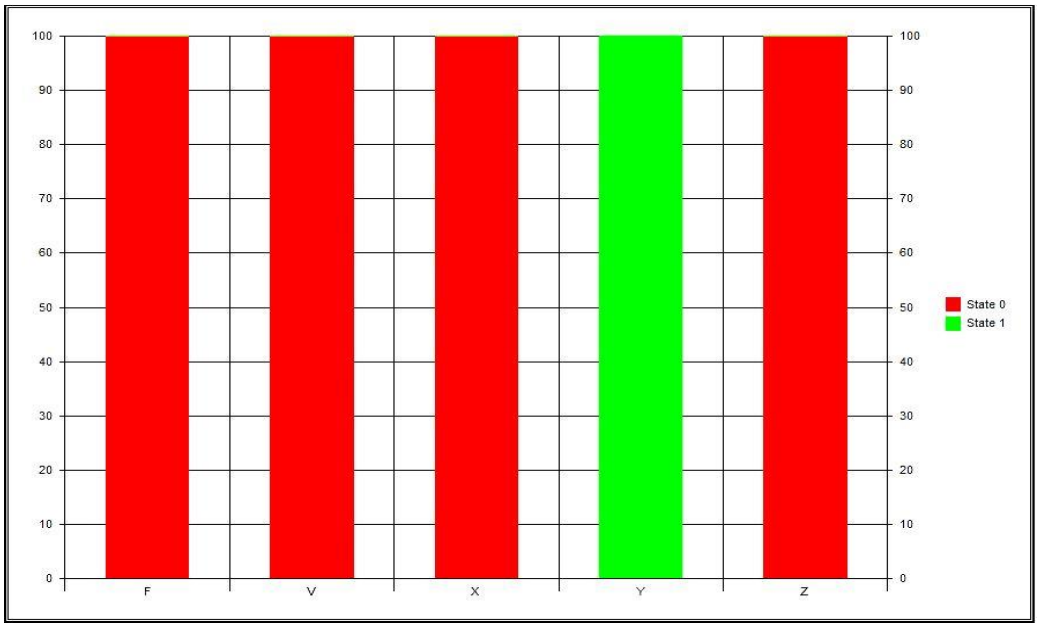


Figure 5.2.2: Bar Chart for Dead feature Rule 2

Chapter 6

Conclusion

6.1 Summery

This thesis presented an approach to defining and verifying software product line feature analysis rules by using Bayesian Networks. Our primary object was to define a set of rules for both dead and false optional features based on Cardinality based. Bayesian Network is used to model and verify the analysis rules. In association to our earlier work on applying First order logic, we represent them into Feature Diagram. To be able to know whether product lines are the right approach for Feature diagram or not, we represent it in Bayesian Network using cumulative node probability. After all this suggested work, we verify result in tools to check whether the result is valid or not.

6.2 Future Work

As we work with only two anomalies of feature model, we like to work with other anomalies & take those anomalies into some rules. We also interested in using probability in those rules.

Other anomalies are

1. Conditionally dead feature
2. Wrong cardinality
3. Redundancy

References

[1] Ian Somerville, "Software Reuse", Based on Software Engineering, 7th Edition.

[2] YijunYu, "Software Reuse", lecture 8, 2005.

[3] Caroline Nyholm, "Product Line Development – an Overview", Department of Computer Science, Mälardalen University, Vasteras, Sweden.

[4] A. Chaudhary, B. K. Verma, J.L. Raheja, "Product Line Development Architectural Model", In proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology, China, 9-11 July, 2010, pp.749-753.

[5] L. Rincón, G. Giraldo, R. Mazo and C. Salinesi, "An Ontological Rule-Based Approach for Analyzing Dead and False Optional Features in Feature Models", Electronic Notes in Theoretical Computer Science (ENTCS), Volume 302, February, 2014 Pages 111-132.

[6] Musfiqur Rahman and Shamim Ripon, "Using Bayesian Networks to Model and Analyze Software Product Line Feature Model", in book Multi-disciplinary Trends in Artificial Intelligence, LNCS 8875, pp. 220-231, 2014, Springer International Publishing, DOI: 10.1007/978-3-319-13365-2_20.

[7] Shamim Ripon, Sk. Jahir Hossain, Keya Azad and Meheedi Hassan, "Modeling and Analysis of Product Line Variants", International Workshop on Requirements Engineering Practices on Software Product Line Engineering (REPOS 2012), In conjunction with SPLC'12. (ACM).

[8] N Fenton, M Neil, and D Marquez, "Using Bayesian networks to predict software defects and reliability", Department of Computer Science, Queen Mary, University of London, London, UK, DOI:10.1243/1748006XJRR161.

[9] Todd A. Stephenson, "An Introduction to Bayesian Network Theory and Usage", IDIAP-RR-00-03,Dalle molle institute for Perceptual Artificial Intelligence, Martigny, Valais, Switzerland.

[10] Kevin B. Korb and Ann E. Nicholson, "Introducing Bayesian Networks", CRC Press, Monash University, Clayton, Victoria 3800 Australia.

[11] Ben-Gal I., "Bayesian Networks", in Ruggeri F., Faltin F. & Kenett R., Encyclopedia of Statistics in Quality & Reliability, Wiley & Sons (2007).

[12] Ingolf H. Kr¨uger, Reena Mathew ,Michael Meisinger ,"From Scenarios to Aspects: Exploring Product Lines",Department of Computer Science University of California, San Diego La Jolla, CA 92093-0114, USA.

[13] Don Batory, "Feature Models, Grammars, and Propositional Formulas", Department of Compute Sciences, University of Texas at Austin, Austin, Texas 78712.

[14] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker, "Formalizing Cardinality-based Feature Models and their Specialization", University of Waterloo, Canada, University of Applied Sciences Kaiserslautern, Zweibr¨ucken, Germany.

[15] Ebrahim Bagheri, Faezeh Ensan, Dragan Gasevic and Marko Boskovic, "Modular Feature Models: Representation and Configuration", Journal of Research and Practice in Information Technology (Australian Computer Journal) 43(2):109-140.

[16] Clément Quinton, Daniel Romero, and Laurence Duchien., “Cardinality-Based Feature Models With Constraints: A Pragmatic Approach”, SPLC - 17th International Software Product Line Conference –2013, Aug 2013, Tokyo, Japan. PP.162-166, 2013.

[17] Czarnecki, K., and C H P. Kim, “Cardinality-based feature modeling and constraints: a progress report”, International Workshop on Software Factories at OOPSLA’05, San Diego, California, USA.

[18] Krzysztof Czarnecki, Simon Helsen, Ulrich Eisenecker , “Staged Configuration Using Feature Models”, Software Product LinesLecture Notes in Computer Science Volume 3154, 2004, pp 266-283.