Article

Semantics of cardinality-based service feature diagrams based on linear logic

Ghulam Mustafa Assad[1], **Muhammad Naeem**[2], **Hafiz Abdul Wahab**[3], **Faisal Bahadur**[1], **Sarfraz Ahmed**[4]
[1]Department of Information Technology, Hazara University, Mansehra, Pakistan
[2]Department of Information Technology, Abbottabad University of Science and Technology, Abbottabad, Pakistan
[3]Department of Mathematics, Hazara University, Mansehra, Pakistan
[4]Department of Mathematics, Abbottabad University of Science and Technology, Abbottabad, Pakistan
E-mail: gm_assad@yahoo.com, naeem@aust.edu.pk, wahabmaths@yahoo.com, msosfaisal@gmail.com, surfy001@yahoo.com

Received 23 June 2015; *Accepted 1 August 2015*; *Published online 1 December 2015*

Abstract
To provide efficient services to end-user it is essential to manage variability among services. Feature modeling is an important approach to manage variability and commonalities of a system in product line. Feature models are composed of feature diagrams. Service feature diagrams (an extended form of feature diagrams) introduced some new notations to classical feature diagrams. Service feature diagrams provide selection rights for variable features. In our previous work, we introduced cardinalities for the selection of features from a service feature diagram which we call cardinality-based service feature diagrams (CSFD). In this paper, we provide semantics to CSFDs. These semantics are backed by the formal calculus of Linear Logic. We provide rules to interpret CSFDs into linear logical formula. Our results show that the linear formulas of CSFDs give the same results as expected from the CSFDs.

Keywords feature modelling; service feature diagrams; software product line; variability and similarity management; cardinality-based service feature diagrams.

Selforganizology
ISSN 2410-0080
URL: http://www.iaees.org/publications/journals/selforganizology/online-version.asp
RSS: http://www.iaees.org/publications/journals/ selforganizology /rss.xml
E-mail: selforganizology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Software product line engineering is one of the ways used by the researchers in industry to automate product development of the product line. One of the challenges for product development is the use of variability and commonality among the features of products. Feature modelling is established notation to deal with such type of challenges (Batory et al., 2006). Feature diagrams were introduced as a part of the Feature-Oriented Domain Analysis (FODA) in (Kang et al., 1990). Feature diagrams are used in number of domains including telecom systems (Griss et al, 1998), template libraries (Czarnecki and Eisenecker, 2000), network protocols (Barbeau and Bordeleau, 2002), and embedded systems (Czarnecki et al., 2002).

Feature models are, hierarchical models to record commonalities and variabilities among the products of a product line. In the model, each characteristic relevant to the problem space is said to be a feature. So in this sense, a feature is called a characteristic of a system. We can say that a feature can be a requirement, a quality, a technical function or a non-functional characteristic (Czarnecki and Kim, 2005). For example, colour, tires, and doors are features of a product line of a car.

The original feature diagrams have many extensions proposed by different authors. For example, the first extension of FODA diagrams is Feature-RSEB, proposed in (Griss et al., 1998). The second extension of FODA diagrams is Cardinality-based feature diagrams, proposed in (Czarnecki, 2005). Another extension of FODA diagrams is service feature diagrams (SFD), proposed by Naeem in (Naeem and Heckel, 2011).

This paper is an extension of our previous work in (Assad et al, 2015). In this paper, we argue to use cardinalities in service feature diagrams; we believe that the full benefit of service feature diagrams can be obtained by using cardinalities. CSFD, not only, reduces the feature types of SFD, but subsumes all the features of SFD, as well. Furthermore, we provide formal semantics to the idea of CSFD proposed in (Assad et al., 2015). These formal semantics are backed by the sequent calculus of Linear Logic (Girard, 1987, 1995; Troelstra, 1992).

The rest of the paper is arranged as follows: Section 2 elaborates on the information which is necessary to understand the technical contents of the paper, while in Section 3 we provided the semantics of our proposed scheme of CSFD in Linear Logic. Section 4 validates our scheme, and section 5 concludes the paper.

2 Background and Related Approaches

2.1 Classical feature diagrams

Usually, feature diagrams represent hierarchies of common and variable features in software product lines (Kang et al, 1990). Here, we use them to describe variability of services building on the notation provided in (Czarnecki and Eisenecker, 2000). For example, Fig. 1 shows a feature diagram D for an online travel agent.

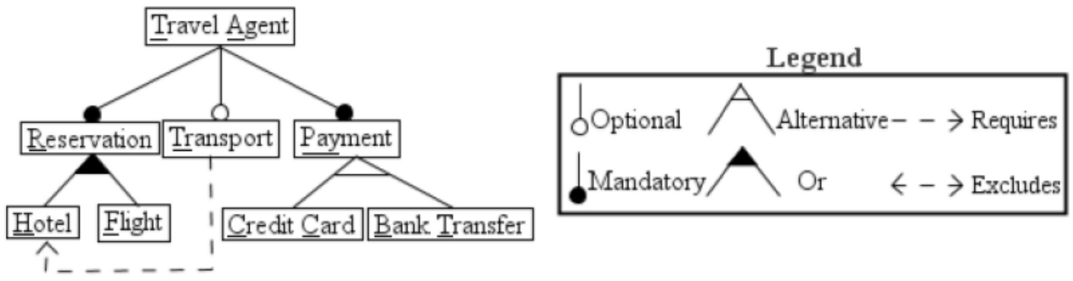


Fig. 1 Feature diagram of an on-line travel agent service.

An instance of D is a subset of features that is consistent with the constraints specified by D. For example, a valid purchase from the travel agent shown in Fig. 1 must obey the following rules:

1. The root feature is always selected
2. If a feature is selected, its parent must also be selected
3. If a feature is selected, all the mandatory features of its And-group are selected
4. If a feature is selected, exactly one feature of its Alternative-group must be selected
5. If a feature is selected, at least one feature of its Or-group must be selected
6. When two features are linked by requires constraint, the target feature must be included whenever the source is

7. A source and target of excludes constraint (not shown in Fig. 1) cannot be selected in an instance.

Thus, an instance of D is given by {TA, Res, H, Pay, BT}(for brevity, we use the underlined characters of the feature name in instances and logical formulas). A declarative way of defining the notion of instance is by means of Propositional Logic (Czarnecki and Wasowski, 2007). For example, a propositional equivalent of the feature diagram in Fig. 1 is:

$$(TA \leftrightarrow Res) \bigwedge (Res \leftrightarrow (H \vee F)) \bigwedge (Tr \leftrightarrow TA) \bigwedge (Tr \rightarrow H) \bigwedge (TA \leftrightarrow Pay) \bigwedge (CC \leftrightarrow (\sim BT \wedge Pay)) \bigwedge (BT \leftrightarrow (\sim CC \wedge Pay))$$

Valuations for which this formula is true characterise the valid instances. In our example, a possible instance is the valuation that assigns true to {TA, Res, H, Pay, BT} and false to {F, Tr, CC}.

2.2 Cardinality-based feature diagrams

Cardinality-based feature diagrams uses multiplicities on features. Cardinality-based feature modelling is an integration and extension of existing approaches. Czarnecki (Czarnecki and Kim, 2005) stated that a cardinality-based feature model is a hierarchy of features where each feature has feature cardinality, i.e., cardinality-based feature diagrams put constraints on features, provides a lower and upper limit for the selection of features.

Feature cardinality denotes the number of clones of sub-features which can be selected for a parent feature. Cardinalities are shown as [m....n], where m and n denote minimum and maximum number of selection for a feature, respectively. Feature with cardinality [1…1] are called mandatory, whereas features with cardinality [0…1] are called optional. Group cardinality is an interval of the form [m–n], where $n \in Z \wedge 0 \leq m \leq n \leq k$, where k is the number of features in the Group (Czarnecki and Kim, 2005).

Fig. 2 depicts a feature diagram showing seating capacity of a car manufacturer using cardinality-based feature diagrams. A possible instance of this feature diagram is {SC, F, LB, PS, R, H}.

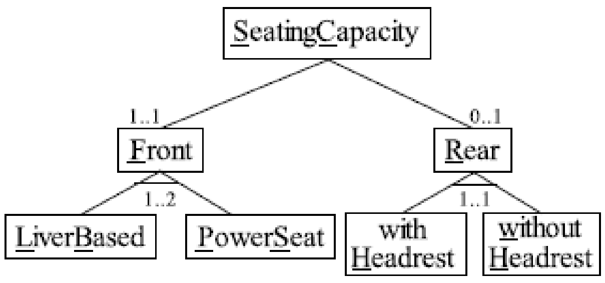


Fig. 2 Cardinality-based feature diagram showing seats of a car.

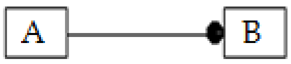
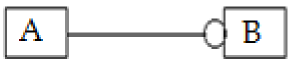
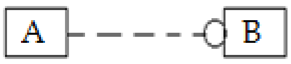
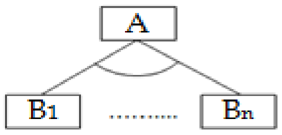
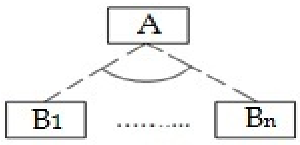
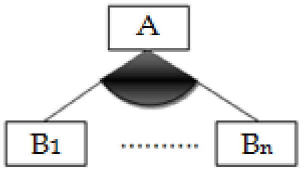
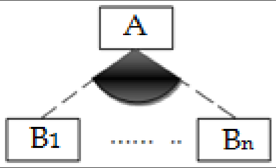
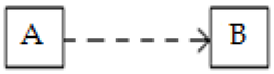
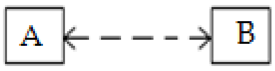
2.3 Service feature diagrams

Service feature diagrams introduced some new type of notations to the classical feature diagrams in the context of service specification and matching. These new notations include:

1. A solid edge: This edge is used when the selection of features is given to the requestor.
2. A dashed edge: A dashed edge is used when the selection rights of features are left with provider to choose from.

3. A resource feature: This feature can only be used once; whereas the classical representation is used for resource feature.
4. A shareable feature: This feature can be used multiple times. A rectangular box with gray background is used to represent shareable features.

Table 1 Notations used in service feature diagrams.

Features	**Feature Representation**	**Comments**
Mandatory		Feature B must be selected if A is, in an instance
Optional		Feature B may be selected or rejected with A in an instance depending on requester's choice.
		Feature B may be selected or rejected with A in an instance depending on provider's choice.
Alternative-group		Exactly one feature from the group of B1,...,Bn must be selected with A in an instance based on the requestor's preference.
		Exactly one feature from the group of B1,...,Bn must be selected with A in an instance based on the provider's preference.
Or-group		At least one feature from the group of B1,...,Bn must be selected with A in an instance based on the requestor's preference.
		At least one feature from the group of B1,...,Bn must be selected with A in an instance based on the provider's preference.
Implies		Target feature B must be selected if the source feature A is.
Exclude		Feature A and B cannot be selected in one instance.

Using the notations discussed above a service feature diagram for an entertainment system of a car manufacturer is shown in Fig. 3 below.

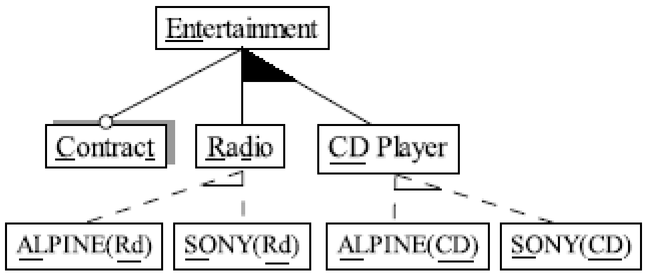
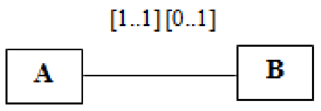
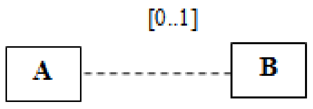
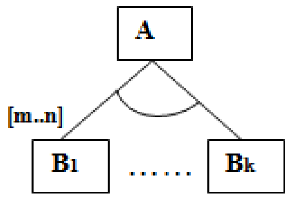
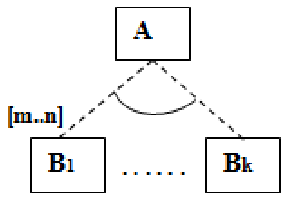


Fig. 3 SFD showing entertainment system of a car manufacturer.

2.4 Cardinality-based service feature diagrams

A cardinality-based service feature model is a hierarchy of features, where each feature has feature cardinality (Assad et al, 2015). A feature cardinality is an interval of the form [m..n], where m and n both are real number. A feature with cardinality [1…1] referred as "Mandatory" whereas feature with cardinality [0…1] referred as "Optional". A group cardinality is an interval of the form [m…n], where both m and n are real numbers and $0 < m \leq n \leq k$, where k is the number of features in the group. Group cardinality denotes how many group members can be selected.

Table 2 Notations used in cardinality-based service feature diagrams.

Features	Graphical Representation	Comments
Single Feature		If feature B is mandatory sub-feature then it must be selected on selection of A, otherwise it may be selected or rejected based on the requestor's preference in an instance.
		A feature B may be selected depending on provider's preference in an instance, if A is selected.
Group Feature		If the feature A is selected then features B_m to B_n must be selected from this group in an instance, where $0 \leq m \leq n \leq k$. This selection of features should be decided on the basis of requester's preferences.
		If the feature A is selected then features B_m to B_n must be selected from this group in an instance, where $0 \leq m \leq n \leq k$. This selection of features should be decided on the basis of provider's preferences.

The use of "Cardinality Based Service Feature Diagrams" simplifies the notations by

1. Eliminating the use of multiple feature types for representing alternative and or-group.

2. Combines optional feature with requestor's choice and mandatory.

We don't need to be confused with filled and unfilled Circle as well as with filled and unfilled arcs.

2.5 Linear logic

Linear logic was proposed by Girard in (1987). In contrast to the Propositional Logic, Linear logic can differentiate between the propositions which occurs multiple times from those which occurs once, in linear logical formula, i.e., $A \otimes A \neq A$, where A is a proposition in Linear Logic.

Linear logic provides three types of connectives: Multiplicative connectives, additive connectives, and exponential connectives. These connectives are used to form the fragments of Linear Logic, while the Classical Linear Logic (CLL) contains all the connectives of Linear Logic. The service feature diagrams can be transformed to Linear Logic. The encoding of a service feature diagram to linear logical formula gives the same result as expected from diagram. Following are the concepts that will be used in this paper. Two propositions A and B are representing features here.

1. Multiplicative Conjunction ($\otimes$). A linear formula $A \otimes B$ shows the selection of both features A and B (Naeem, 2012).
2. Additive Conjunction (&). A linear formula A & B is representing choice A or B (Naeem, 2012).
3. Linear Implication ($\multimap$). A linear expression $A \multimap B$ means that a feature B can only be selected if we have already chosen the feature A. We use linear implication to impose the condition where we want to select a feature before the other feature. For example, a sub-feature can only be selected if its parent is already chosen (Naeem, 2012).
4. Storage Operator (!). It is used to copy a linear proposition. A linear expression !A states the selection of a feature A as many times as required (Naeem, 2012).

Inference system

The basic linear inference system is a sequent, written in Gentzen's style (Cosmo and Miller, 2010). A sequent contains two sequences separated by turnstile $\vdash$ (also read as yields or derives). If Γ and Δ are the multi-sets of the finite sequences of formulas then $\Gamma \vdash \Delta$ represents a sequent in Linear Logic, which states that the multiplicative conjunction of the formulas inside Γ derives the multiplicative disjunction of the formulas in Δ (Lincoln et al, 1992).

An inference rule can be written as

$$\text{Rule}\ \frac{\text{Hypothesis1} \qquad \text{Hypothesis2}}{\text{Conclusion}}$$

In this rule, hypotheses and conclusion are represented in the form of sequent, while Rule represents the name of inference rule applied to Hypothesis1 and Hypothesis2 to get to the Conclusion (Naeem, 2012).

The deduction system of CLL consists of the basic rule and introduction rules for the connectives described above (Girard, 1987; Troelstra, 1992; Cosmo and Miller, 2010). We have only one basic rule, i.e., the identity rule

$$\text{id}\ \frac{}{A \vdash A}$$

which states that a formula A can be derived from the assumption of a formula A. Linear propositions can be moved from one side of a sequent to the other, as shown by the following rules

$$L(.)^{\perp}\ \frac{\Gamma \vdash A, \Delta}{\Gamma, A^{\perp} \vdash \Delta} \qquad R(.)^{\perp}\ \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash A^{\perp}, \Delta}$$

The multiplicative conjunction (⊗) has two introduction rules. First for introducing ⊗ on the left, second for introducing ⊗ on the right of a sequent:

$$L\otimes\ \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A\otimes B \vdash \Delta} \qquad R\otimes\ \frac{\Gamma \vdash A, \Delta \qquad \Gamma' \vdash A, \Delta'}{\Gamma \vdash A\otimes B, \Delta, \Delta'}$$

The additive conjunction (&) have two-introduction rules for the left and the right of a sequent, as shown below

$$L\&\ \frac{\Gamma, B_i \vdash \Delta}{\Gamma, B_1\&B_2 \vdash \Delta} \qquad R\otimes\ \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A\otimes B, \Delta}$$

The linear implication ⊸ also has two rules for introducing it on the left and right of the sequent

$$L\multimap\ \frac{\Gamma \vdash A, \Delta \qquad \Gamma' \vdash B, \Delta'}{\Gamma, A\multimap B, \Gamma' \vdash \Delta, \Delta'} \qquad R\multimap\ \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A\multimap B, \Delta}$$

In CLL, weakening and contraction rules are only allowed for the propositions having modalities.

$$W!\ \frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad C!\ \frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}$$

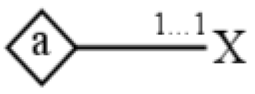
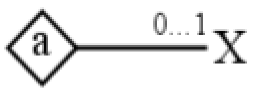
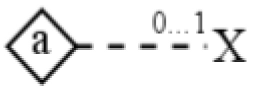
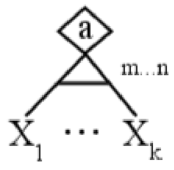
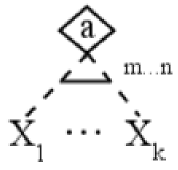
The !-modality can be introduced on the left and the right side of a sequent, as shown by the following rules

$$D!\ \frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad R!\ \frac{!\Gamma \vdash B}{\Gamma \vdash !B}$$

3 Semantics of Cardinality-based Service Feature Diagrams

A cardinality-based service feature diagram can be encoded into a logical formula which will give the same result as expected from the diagram. Table 3 shows the rules we provided to interpret a CSFD into Linear Logic. CSFD offers two types of relationship of feature with its sub features: 1) Single feature; 2) Group feature. Single feature is either mandatory or an optional feature, while the Group feature is the combination of multiple features.

Table 3 Encoding of different feature types in linear logic.

Rule	Graphical Representation	Linear Formulae	Comments
Rule::Res	$\boxed{a}$	LF($\boxed{a}$) = a	Feature *a* being resource means that we can derive *a* from *a*.
Rule::Share	$\boxed{a}$	LF($\boxed{a}$) = !a	Feature *a* being shareable means that we can derive *a*□…□*a* from !*a*.
Rule::Man		$LF(\alpha) \otimes (LF(\alpha) \otimes (LF(\alpha) \multimap LF(X)))$	Here X is a sub tree which is mandatory for feature *a*, i.e., X must be selected with feature a.
Rule::Opt$_R$		$LF(\alpha) \otimes (LF(\alpha) \otimes (LF(\alpha) \multimap (LF(X) \& LF(X)^{\perp})))$	Here sub tree X is optional for a, i.e., X may be selected or rejected with feature a, depends upon requestor's choice. If a is selected, one can derive both LF(X) and LF(X) $^{\perp}$
Rule::Opt$_P$		$LF(\alpha) \otimes (LF(\alpha) \otimes (LF(\alpha) \multimap (LF(X) \oplus LF(X)^{\perp})))$	Here sub tree X is optional for a, i.e., X be selected or rejected with feature a, depends upon provider's choice. If x is selected, one can derive both LF(X) $\oplus$ LF(X) $^{\perp}$
Rule::Group$_R$		$LF(\alpha) \otimes (LF(\alpha) \otimes (LF(\alpha) \multimap (\&_{i=1 \text{ to } e} (\&_{j=1 \text{ to } k}(LF(X_J) \otimes \bigotimes_{L \in S} LF(X_L)^{\perp})))$	The group of sub features $X_1, \ldots, X_k$ allow for requestor to make selection between all subsets of features in the range from m to n, and deselecting their respective complements.
Rule::Group$_P$		$LF(\alpha) \otimes (LF(\alpha) \otimes (LF(\alpha) \multimap (\oplus_{i=1 \text{ to } e} (\oplus_{j=1 \text{ to } k}(LF(X_J) \otimes \bigotimes_{L \in S} LF(X_L)^{\perp})))$	The group of sub features $X_1, \ldots, X_k$ allow for Provider to make selection between all subsets of features in the range from m to n, and deselecting their respective complements.

The 1st column of Table 3 shows the rule names used for the encoding. The 2nd column shows the graphical representation of the rule. The 3rd column shows the linear formulas of different CSFDs while the 4th column explains the encoding.

A diamond is used as a meta variable for representing features. LF(α) shows the meta proposition that will be replaced by *a* for α being resource and by !a for α being shareable feature. The linear implication $a \multimap b$ is used to express that we have to choose feature *a* to select feature *b*. This means that *a* gets consumed once *b* is obtained, the extra copy of the parent feature is used to keep the intermediate feature in possible instance formula. For example $a \otimes (a \otimes (a \multimap b))$ explain that it is required to choose feature *a* for the selection of the feature *b*. To encode a CSFD, it is required to keep three copies of parent feature before the linear implication.

The additive conjunction (a & b) represents the alternative occurrences of features a and b, depends on requestor choice. The additive disjunction $(a \oplus b)$ represents the alternative occurrences of features a and b, depends on provider choice. The subscripts (i=1 to e) denotes the number of features to be selected, where e is an element of a set t which represents cardinalities (m..n). l represents the set of rejected features from the set s.

$$s = \{1,2,\ldots..k\} \qquad t = \{t: m \le t \le n\}$$
$$e = \text{an element of } t \quad l = \{l: l \in s \wedge l \neq j\}$$

Let us explain the concepts discussed so far with the help of a general example, where a feature a is a parent feature of a group(R) of two sub-features b and c the groups has [1..1] cardinality. The feature b also represents a group(R) having two sub-features d and e, with [1..2] cardinality. The feature c has a mandatory sub-feature f and an optional sub-feature g, as shown in Fig. 4.

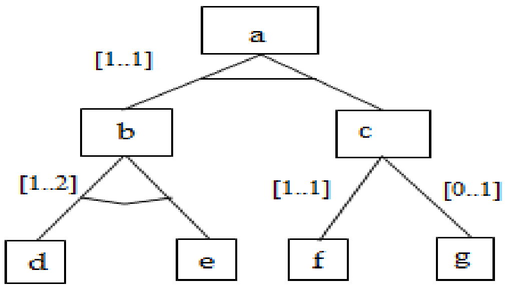


Fig. 4 An Example Cardinality-Based Service Feature Diagram

The step wise encoding of the cardinality-based service feature diagram using the rules given in Table 3 is shown below

1. $LF(a) \otimes (LF(a) \otimes (LF(a) \multimap ((LF(b) \otimes LF(c)^{\perp}) \& (LF(b)^{\perp} \otimes LF(c)))))$
2. $a \otimes (a \otimes (a \multimap ((LF(b) \otimes (LF(b) \otimes (LF(b) \multimap ((LF(d) \otimes LF(e)^{\perp}) \& (LF(d)^{\perp} \otimes LF(e)) \& (LF(d) \otimes LF(e))) \otimes c^{\perp} \otimes f^{\perp} \otimes g^{\perp})))) \& (LF(b)^{\perp} \otimes LF(d)^{\perp} \otimes LF(e)^{\perp} \otimes LF(c \otimes (LF(c) \otimes (LF(c) \multimap ((LF(f) \otimes LF(g)^{\perp}) \& (LF(f) \otimes LF(g))))))))$
3. $a \otimes (a \otimes (a \multimap ((b \otimes (b \otimes (b \multimap ((d \otimes e^{\perp}) \& (d^{\perp} \otimes e) \& (d \otimes e) \otimes c^{\perp} \otimes f^{\perp} \otimes g^{\perp})))) \& (b^{\perp} \otimes d^{\perp} \otimes e^{\perp} \otimes (c \otimes (c \otimes (c \multimap (f \otimes g^{\perp}) \& (f \otimes g))))))))$

The linear formula LF of Fig. 4 is

$$LF=a\otimes(a\otimes(a\multimap((b\otimes(b\otimes(b\multimap((d\otimes e^{\perp})\&(d^{\perp}\otimes e)\&(d\otimes e)\otimes c^{\perp}\otimes f^{\perp}\otimes g^{\perp}))))\&(b^{\perp}\otimes d^{\perp}\otimes e^{\perp}\otimes(c\otimes (c\otimes(c\multimap(f \otimes g^{\perp})\&(f \otimes g))))))))$$

Instance and instance formula

The diagrammatical representation of an instance of a feature diagram is called instance diagram (Naeem, 2012). An instance is defined as, a set of permissible selection of features from a feature diagram (Kang et al, 1990; Czarnecki and Eisenecker, 2000; Benavides et al., 2010). Instance diagram can be encoded into a logical formula, called instance formula, which will give the same result as expected from the instance diagram. The number of instance formulas of a CSFD depends on the level of variability captured. The CSFD must have at least one instance formula. The set of instance formulas IF for Fig. 4 are

$$IF= \{ (a\otimes b\otimes d\otimes e^{\perp}\otimes c^{\perp}\otimes f^{\perp}\otimes g^{\perp}), (a\otimes b\otimes d^{\perp}\otimes e\otimes c^{\perp}\otimes f^{\perp}\otimes g^{\perp}), (a\otimes b\otimes d\otimes e\otimes c^{\perp}\otimes f^{\perp}\otimes g^{\perp}), (a\otimes b^{\perp}\otimes d^{\perp}\otimes e^{\perp}\otimes c\otimes f\otimes(g\&g^{\perp})) \}$$

Here the symbol $\perp$ shows the rejection of a feature, the additive conjunction (g & $g^{\perp}$) is used to show that the selection or rejection of feature g depends upon requester's choice, whereas for provider's choice $\oplus$ is use. All these instances can be derived from linear formula LF of Fig. 4.

Note that rejection of a subtree X, leads towards the rejection of its sub features, that's why in above given formulas rejection of feature c means rejection of its subfeatures f and g, similarly rejection of feature b means rejection of its subfeatures d and e.

4 Validation

Validation of our work is based on derivation of instance formula IF from a linear formula LF of a feature diagram with the help of the formal framework of Linear Logic. Let us consider an example where a feature a is a parent feature of a group(R) of two sub-features b and c having [1..1] cardinality. The feature b also has a solitary feature d with [1..1] cardinality, as shown below in Fig. 5.

The linear formula LF and the set of instance formulas IF of this cardinality based service feature diagram are

$LF= a\otimes(a(\otimes a\multimap ((b\otimes c^{\perp}\otimes d^{\perp})\&(b^{\perp}\otimes(c\otimes(c\otimes(c\multimap d))))))$

$IF=\{(a\otimes b\otimes c^{\perp}\otimes d^{\perp}), (a\otimes b^{\perp}\otimes c\otimes d)\}$

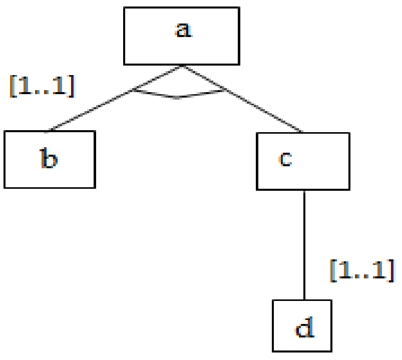


Fig. 5 An Example CSFD for validation.

The derivation of in instance formula IF from the Linear formula (LF) can be shown as LF ⊢ IF. This derivation is obtained by using an online prover for linear logic called llprover (Tamura, 1995). We derive all the instances from the corresponding formulas of the CSFD. The proof tree for the first instance formula of the set of instance formulas is

```
                                   ------------------ id
                                  b⊗c⊥⊗d⊥  ⊢  b⊗c⊥ ⊗d⊥
                ------- id  ------------------------------------ L&
                  a ⊢ a      (b⊗c⊥ ⊗d⊥ )&(b⊥ ⊗c⊗c⊗(c⊸d)) ⊢b⊗c⊥ ⊗d⊥
      ------- id  ------------------------------------------------ L⊸
      a ⊢ a          a,a⊸((b⊗c⊥ ⊗d⊥ )&(b⊥ ⊗c⊗c⊗(c⊸d)) ⊢ b⊗c⊥ ⊗d⊥
           ---------------------------------------------------- R⊗
           a,a,a⊸((b⊗c⊥ ⊗d⊥)&(b⊥ ⊗c⊗c⊗(c⊸d))) ⊢ a⊗b⊗c⊥ ⊗d⊥
           ---------------------------------------------------L⊗
           a,a⊗(a⊸((b⊗c⊥ ⊗d⊥)&(b⊥ ⊗c⊗c⊗(c⊸d))) ⊢ a⊗b⊗c⊥ ⊗d⊥
           --------------------------------------------------- L⊗
           a⊗a⊗(a⊸((b⊗c⊥ ⊗d⊥)&(b⊥ ⊗c⊗c⊗(c⊸d))) ⊢ a⊗b⊗c⊥ ⊗d⊥
```

The proof tree for the second instance formula of the set of instance formulas is

```
                             ------- id  ------- id
                             c ⊢ c       d ⊢ d
                   ------- id  ------------------ L⊸
                   c ⊢ c         c,c⊸d ⊢ d
                ------- id  -------------------------- R⊗
               b⊥ ⊢ b⊥        c,c,c⊸d ⊢ c⊗d
                 ---------------------------------- R⊗
                         b⊥,c,c,c⊸d ⊢ b⊥⊗c⊗d
                      ----------------------- L⊗
                        b⊥,c,c⊗(c⊸d) ⊢ b⊥⊗c⊗d
                      ----------------------- L⊗
                        b⊥,c⊗c⊗(c⊸d) ⊢ b⊥⊗c⊗d
                      ----------------------- L⊗
                        b⊥⊗c⊗c⊗(c⊸d) ⊢ b⊥⊗c⊗d
              ------- id  ------------------------------- L&
                a ⊢ a      (b⊗c⊥⊗d⊥)&(b⊥⊗c⊗c⊗(c⊸d)) ⊢ b⊥⊗c⊗d
          ------- id  ------------------------------------------ L⊸
           a ⊢ a      a,a⊸((b⊗c⊥⊗d⊥)&(b⊥⊗c⊗c⊗(c⊸d))) ⊢ b⊥⊗c⊗d
           ---------------------------------------------------- R⊗
              a,a,a⊸((b⊗c⊥⊗d⊥)&(b⊥⊗c⊗c⊗(c⊸d))) ⊢ a⊗b⊥⊗c⊗d
              ------------------------------------------- L⊗
              a,a⊗(a⊸((b⊗c⊥⊗d⊥)&(b⊥⊗c⊗c⊗(c⊸d))) ⊢ a⊗b⊥⊗c⊗d
              ------------------------------------------- L⊗
              a⊗a⊗(a⊸((b⊗c⊥⊗d⊥)&(b⊥⊗c⊗c⊗(c⊸d))) ⊢ a⊗b⊥⊗c⊗d
```

5 Conclusions and Future Work

Service feature diagrams were proposed by Naeem in (Naeem and Heckel, 2011; Naeem, 2012). Our previous paper (Assad et al, 2015) was the first step towards the development of a framework for the cardinality-based service feature diagrams. In this paper, we have provided formal rules to interpret cardinality based service feature diagrams into a linear logical formula. The encoding of cardinality based service feature diagrams to a linear logical formula gives the same results as expected from diagram. We have also validated our work with the help of examples given in Section 4. Our objective of formalizing Cardinality based service feature diagrams in linear logic (Girard, 1987; Troelstra, 1992; Girard, 1995) has been achieved.

In future we are looking for developing a tool support for our proposed approach.

References

Assad GM, Naeem M, Wahab HA. 2015. Towards cardinality-based service feature diagrams. Computational Ecology and Software, 5(1):

Barbeau M, Bordeleau F. 2002. A Protocol Stack Development Tool using Generative Programming. Proceedings of the ACM Conference on Generative Programming and Component Engineering (GPCE’02), Pittsburgh, Vol. 2487 of LNCS. 93-109, Springer-Verlag, Heidelberg, Germany

Batory D, Benavides D, Ruiz-Cortes A. 2006. Automated analysis of feature models: challenges ahead. Communications of the ACM, 49(12): 45-47

Benavides D,Segura S,and RuizCort’es A. 2010. Automated analysis of feature models 20 years later: A literature review. Information Systems, 35(6): 615-636

Cosmo R and Miller D. 2010. Linear Logic. In: The Stanford Encyclopedia of Philosophy (Zalta EN, ed)(Fall 2010 edition), USA

Czarnecki K, Bednasch T, Unger P, Eisenecker UW. 2002. Generative Programming for Embedded Software: An Industrial Experience Report. Proceedings of the ACM Conference on Generative Programming and Component Engineering (GPCE’02), Pittsburgh, Vol. 2487 of LNCS. 156-172, Springer-Verlag, Heidelberg, Germany

Czarnecki K, Eisenecker UW. 2000. Generative Programming: Methods, Tools, and Applications, Addison-Wesley, Boston, MA, USA

Czarnecki K, Helsen S, Eisenecker U. 2005. Formalizing Cardinality-based Feature Models and their Specialization. Software Process Improvement and Practice. 10(1): 7-29

Czarnecki K, Kim CHP. 2005. Cardinality-Based Feature Modeling and Constraints: A Progress Report. Proceedings of OOPSLA’05 Workshop on Software Factories, San Diego, California, USA

Czarnecki K, Wasowski A. 2007. Feature Diagrams and Logics: There and Back Again. Proceedinsg of International Conference on Software Product Lines (SPLC’07). 23-34

Girard J-Y, 1987. Linear logic. Theoretical Computer Science, 50: 1-102

Girard J-Y. 1995. Linear Logic: Its Syntax and Semantics. In Proceedings of the Workshop on Advances in Linear Logic (ALL’95), pages 1-42, New York, NY, USA

Griss M, Favaro J, d’Alessandro M. 1998. Integrating Feature Modeling with the RSEB. Proceedings of the Fifth International Conference on Software Reuse (ICSR). 76-85, IEEE Computer Society Press, Los Alamitos, CA, USA

Kang K, Cohen S, Hess J, Novak W, Peterson S. 1990. Feature-oriented Domain Analysis (FODA) Feasibility Study. Technical Report, Carnegie-Mellon University Software Engineering Institute, USA

Lincoln P, John C. Mitchell, Andre Scedrov, and Natarajan Shankar. 1992. Decision problems for propositional linear logic. Annals of Pure and Applied Logic, 56(1-3): 239-311

Naeem M, and Heckel R. 2011. Towards Matching of Service Feature Diagrams based on Linear Logic. Proceedings of Workshops of SPLC.

Naeem M. 2012. Matching of Service Feature Diagrams using Linear Logic. PhD Dissertation. Department of Computer Sciences, University of Leicester, UK

Troelstra AS. 1992. Lectures on Linear Logic. Center for the Study of Language and Information, Stanford, CA, USA

Tamura N. 1995. User's Guide of a Linear Logic Theorem Prover (llprover). Technical report, Faculty of Engineering, Kobe University, Japan. Available online at http://bach.istc.kobe-u.ac.jp/llprover/