Linear Time Computation of Variation Degree and Commonalities on Feature Diagrams

Mathieu Vavrille, Erwan Meunier, Charlotte Truchet, Charles Prud'Homme

IMT Atlantique
Département Automatique, Productique
et Informatique
Campus de Nantes
4, rue Alfred Kastler
CS 20722
44307 Nantes Cedex 3
France
T +33 (0)2 51 85 81 00
F +33 (0)2 99 12 70 08
URL : **www.imt-atlantique.fr**

Linear Time Computation of Variation Degree and Commonalities on Feature Diagrams

Collection des rapports de recherche d'IMT Atlantique (En ligne)

RR-2023-01-DAPI

Mathieu Vavrille[1], *mathieu.vavrille@univ-nantes.fr*
Erwan Meunier[2], *erwan.meunier@etu.univ-nantes.fr*
Charlotte Truchet[1], *charlotte.truchet@univ-nantes.fr*
Charles Prud'homme[1], *charles.prudhomme@imt-atlantique.fr*

[1] Nantes Université, École Centrale Nantes, IMT Atlantique, CNRS, LS2N, UMR 6004, F-44000 Nantes, France
2 Nantes Université, UFR Sciences et Techniques, F-44000 Nantes, France

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Linear Time Computation of Variation Degree and Commonalities on Feature Diagrams

Mathieu Vavrille[1], Erwan Meunier[2], Charlotte Truchet[1], and Charles Prud'homme[1]

[1]Nantes Université, École Centrale Nantes, IMT Atlantique, CNRS, LS2N, UMR 6004, F-44000 Nantes, France
{mathieu.vavrille,charlotte.truchet}@univ-nantes.fr,
charles.prudhomme@imt-atlantique.fr
[2]Nantes Université, UFR Sciences et Techniques, F-44000 Nantes, France
erwan.meunier@etu.univ-nantes.fr

January 2023

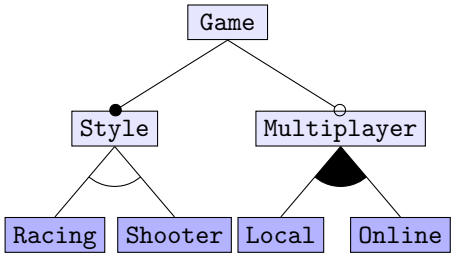
Abstract

The growth in Software Product Lines (SPLs) created a need for good and fast algorithms to analyse them. Extracting information, such as the number of products, may be crucial to generate a good test suite. However, due to the combinatorial complexity of SPLs, such algorithms may be intractable. Here, we focus on Feature Diagrams, and we use the tree structure to show linear time algorithms to compute the variation degree (number of products) and the commonalities (number of products containing each feature). We use the variation degree to show a uniform sampling procedure of configurations allowed by a feature diagram.

1 Introduction

Feature Models are representation of complex configuration systems. For example, an automotive product line allows to configure the system (a car being produced) with interacting features (such as the engine power, tyre dimensions, GPS, ...). The analysis of a feature model provides information on how to build an efficient test suite of the product line. In the case of Software Product Lines, a test suite is a way of finding bugs in the software. For example, the Linux kernel can be represented as a feature model containing thousands of features (libraries, options, hardware, ...). An important information for algorithms on feature models is the number of products allowed, called the variation degree. A finer information, the commonality of a feature, gives the number of products containing this specific feature.

In this report we show how to compute in linear time the variation degree and the commonalities of every feature on feature diagrams. The algorithms were introduced in [von der Maßen and Lichter, 2005] and [Fernández-Amorós et al., 2014]. This report is a synthesis of the algorithms on commonalities and variation degrees from these two papers. We then show how the variation degree can be used to design a fast uniform sampler on feature diagrams. We also provide formal proofs for all the theorems and formulas.



Cross-tree constraints:
`Shooter` $\implies$ `Online`

(a) Example of Feature Model

	1	2	3	4	5	6	7	8
Game	∗	∗	∗	∗	∗	∗	∗	∗
Style	∗	∗	∗	∗	∗	∗	∗	∗
Racing	∗	∗	∗	∗				
Shooter					∗	∗	∗	∗
Multiplayer	∗	∗	∗		∗	∗	∗	
Local	∗		∗		∗		∗	
Online	∗	∗			∗	∗		

(b) Allowed configurations. There is one column per configuration, and a $*$ means that the feature is present. The configurations 7 and 8 are allowed by the feature diagram but not the cross-tree constraint

Figure 1: A feature model and its set of allowed configurations.

This article is structured as follow : Section 2 defines feature models, Section 3 shows how to compute the variation degree, Section 4 shows how to compute the commonalities, and Section 5 shows how to use the variation degree to design a uniform sampler. The proofs are given in Appendix A.

2 Feature Models

A Feature Model is a graphical and condensed representation of the products of a Software Product Line. Given a fixed set of features $\mathcal{F}$, a feature model is a pair of, first, a feature diagram, which gives hierarchical structure of the features organization, and second, a conjunction of propositional formulas over $\mathcal{F}$.

Example. *Figure 1a shows a feature model representing games. It is mandatory that a game has a* `Style` *(black dot over the* `Style` *node), and optionally a* `Multiplayer` *mode (empty dot over the* `Multiplayer` *node). The* `Style` *can be either a* `Racing` *game or a* `Shooter` *game, but not both (represented by the circle arc between the two nodes). If there is a* `Multiplayer` *mode, it can be* `Local` *or* `Online`*, or both (represented by the black circle arc between the two nodes). On top of that, there is a constraint stating that if a game is a* `Shooter` *game, then there should be an* `Online` *mode.*

Table 1b shows the configurations allowed by this feature model. The last two configurations (7 and 8) are allowed by the feature diagram, but not by the cross-tree constraint.

We now give a formal definition of a feature model. Let $\mathcal{F}$ a set of features.

Definition 1 (Feature Diagram)**.** A *feature diagram* is an n-ary labeled tree, where the nodes can be of different types. A feature diagram D stores at its root a feature $D.feature \in \mathcal{F}$. The children can be from:

- a `mandatory/optional` group, with the sets $D.mand$ containing the mandatory children and $D.opt$ containing the optional children,
- an exclusive (`xor`) group, with the set $D.xor$ containing the children,
- an `or` group, with the set $D.or$ containing the children.

In addition, every feature appearing must appear only once in a feature diagram.

This definition is a recursive definition of feature diagrams. We call D' a sub-feature diagram of D if D is an ancestor of D' or D itself.

Definition 2 (Allowed Configuration)**.** A configuration is a subset of features. Given a feature diagram D, a configuration $C \subseteq \mathcal{F}$ is allowed iff:

- $D.feature \in C$
- for all D' sub-feature diagrams of D, $\forall D'' \in D'.children, D''.feature \in C \Rightarrow D'.feature \in C$
- for all D' sub-feature diagrams of D, if $D'.feature \in C$ then:
  - $\forall D'' \in D'.mand, D''.feature \in C$
  - $\exists D'' \in D'.or, D''.feature \in C$
  - $\exists! D'' \in D'.xor, D''.feature \in C$

We denote by $Sols\,(D)$ the set of allowed configurations.

Informally, all the features in $D.mand$ children have to be taken, at least one feature in the $D.or$ group has to be taken, and exactly one feature in the $D.xor$ group has to be taken. If a feature is taken then its parent feature is also taken.

To allow for more expressiveness when modeling feature interactions, feature diagrams are extended with propositional formulas allowing to model interactions, in particular between features that are not in the same sub-feature diagram.

Definition 3 (Feature Model)**.** A *Feature Model* FM is a pair $\langle D, \psi \rangle$ where D is a feature diagram and ψ is a boolean formula where variables are features included in $\mathcal{F}$. The constraints in ψ are called *cross-tree constraints*. A configuration is allowed by a feature model $\langle D, \psi \rangle$ if it is allowed by D and satisfies the boolean formula ψ.

The propositional formulas allow for more diverse constraints, but also make the problem much harder, because finding one configuration satisfying the propositional formulas is NP-complete.

In the following we forget about the cross-tree constraints and focus on the feature diagrams. The tree structure allows for polynomial (even linear) algorithms in the number of features. These results can then be used as approximations on the whole feature model.

3 Variation Degree

The variation degree is the number of configurations allowed by a feature model. It can be computed recursively thanks to the following formulas.

Theorem 1. *[Variation Degree of Feature Diagrams [von der Maßen and Lichter, 2005]] Let D be a feature diagram. Then*

- *If* $D.children = \emptyset$*, then* $|Sols\,(D)| = 1$

- *If* $D.mand \cup D.opt \neq \emptyset$,

$$|Sols(D)| = \prod_{D' \in D.mand} |Sols(D')| \times \prod_{D' \in D.opt} |Sols(D')| + 1$$

- *If* $D.xor \neq \emptyset$,

$$|Sols(D)| = \sum_{D' \in D.xor} |Sols(D')|$$

- *If* $D.or \neq \emptyset$,

$$|Sols(D)| = \left(\prod_{D' \in D.or} |Sols(D')| + 1 \right) - 1$$

From this theorem, a procedure to recursively compute the variation degree can naturally be derived. This procedure has a complexity linear in the number of features, and also computes the variation degree of every sub-feature diagram. All these results can be memoized for later access in constant time.

Example. *We show the computation of the variation degree on the example of Figure 1. We note by* D_f *the feature diagram rooted in feature* f.

- *The variation degree of all leaves is* 1 *(singleton product), so for all* D *in* $\{D_{Racing}, D_{Shooter}, D_{Local}, D_{Online}\}$, $|Sols(D)| = 1$.
- D_{Style} *is a xor node, so the variation degrees of children are added:* $\left|Sols\left(D_{Style}\right)\right| = 2$.
- $D_{Multiplayer}$ *is an or node, so the formula is* $\left|Sols\left(D_{Multiplayer}\right)\right| = (|Sols(D_{Local})| + 1) \cdot (|Sols(D_{Online})| + 1) - 1 = 3$.
- *The root node,* D_{Game} *is a mandatory/optional node. The formula gives* $|Sols(D_{Game})| = \left|Sols\left(D_{Style}\right)\right| \cdot (\left|Sols\left(D_{Multiplayer}\right)\right| + 1) = 8$

4 Commonalities

The variation degree is an important value to know before trying to enumerate all the solutions. However, it does not provide informations on specific features: depending on the structure of the feature model, some features may appear more often than others in the set of allowed configurations. The commonalities give an insight on the presence of features in the allowed configurations.

Definition 4 (Commonality)**.** The *commonality* of a feature f in a feature diagram D, noted $\varphi_f(D)$, is its frequency of appearance in the set of allowed configuration, i.e.

$$\varphi_f(D) = \frac{|\{C \in Sols(D) \,|\, f \in C\}|}{|Sols(D)|}$$

The commonality is heavily linked to random sampling approaches, such as uniform sampling in [Oh et al., 2019a], or BAITAL [Baranov et al., 2020]. The probability that a given feature appears in the solution returned by a uniform sampler is equal to the commonality, as stated in the following proposition.

Proposition 1 ([Oh et al., 2019b])**.** *Let $\mathcal{U}$ be a uniform sampler (i.e. $\forall C \in Sols(D), \mathbb{P}(\mathcal{U}(D) = C) = 1/|Sols(D)|$), then*

$$\forall f \in \mathcal{F}, \mathbb{P}(f \in \mathcal{U}(D)) = \varphi_f(D) \ .$$

Proof. We use the definition of the probability of a random event (positive cases divided by total cases), and the definition of the commonality

$$\begin{aligned}\mathbb{P}(f \in \mathcal{U}(D)) &= \frac{\#\text{positive cases}}{\#\text{total cases}} \\ &= \frac{|\{C \in Sols(D) \,|\, f \in C\}|}{|Sols(D)|} \\ &= \varphi_f(D)\end{aligned}$$

□

The commonalities is then equal to the probability of finding a feature in solutions found by a sampler. If there are features with very low commonality, a sampler may never return a solution containing it in a reasonable number of samples.

As for the variation degree, the commonality of a feature can be computed with a recursive function thanks to the following formulas.

Theorem 2. *[Commonalities on Feature Diagrams [Fernández-Amorós et al., 2014]] Let f be a feature and D be a Feature Diagram. We note $\phi_f(D) = |\{C \in Sols(D) \,|\, f \in C\}|$ the number of occurrences of a feature in the set of allowed configurations. Then*

$$\phi_f(D) = \begin{cases} |Sols(D)| & \text{if } D.feature = f \\ \frac{|Sols(D)|}{|Sols(D')|} \cdot \phi_f(D') & \text{if } f \in D' \text{ and } D' \in D.mand \\ \frac{|Sols(D)|}{|Sols(D')|+1} \cdot \phi_f(D') & \text{if } f \in D' \text{ and } D' \in D.opt \text{ or } D' \in D.or \\ \phi_f(D') & \text{if } f \in D' \text{ and } D' \in D.xor \end{cases}$$

The commonality of f in D can then be computed with $\varphi_f(D) = \frac{\phi_f(D)}{|Sols(D)|}$.

From this theorem, we naturally derive a recursive computation method for the number of occurrences for a single feature. The computation of the commonality of all features of D can also be done in a single traversal of D, leading to a complexity linear in the number of features. The algorithm for computing the commonality for every feature is given in Algorithm 1.

The commonality of every feature in the feature diagram is a rough approximation of the commonality in the whole feature model (including the propositional formulas). However the problem of computing the commonality (or even the variation degree) is much harder in the general case, and requires calls to a #-SAT solver. For example the strategy 3 of BAITAL [Baranov et al., 2020] makes $|\mathcal{F}|+1$ calls to a #-SAT solver to compute all the commonalities. On large feature models this may be prohibitive.

Function OCCURRENCES(R)
  Data: A feature diagram R
  Result: A mapping $\sigma : \mathcal{F} \to \mathbb{N}$ from features to their number of occurrences
  $\sigma \leftarrow \{\}$
  OCCURRENCESREC($R, 1, \sigma$)
  return σ

Procedure OCCURRENCESREC(R, κ, σ)
  Data: A feature diagram R, an integer κ for the recursive factor and a mapping σ.
  Result: Nothing is returned, but σ is filled with the features present in R.
  $\sigma[R.feature] \leftarrow \kappa \cdot |Sols(R)|$
  for $R' \in R.mand$ **do**
    OCCURRENCESREC($R', \frac{|Sols(R)|}{|Sols(R')|}\kappa, \sigma$)
  for $R' \in R.opt \cup R.or$ **do**
    OCCURRENCESREC($R', \frac{|Sols(R)|}{|Sols(R')|+1}\kappa, \sigma$)
  for $R' \in R.xor$ **do**
    OCCURRENCESREC(R', κ, σ)

Algorithm 1: Computation of the number of occurrences of every feature in the set of allowed configurations.

5 Uniform Sampling on Feature Diagrams

In addition to giving information on the SPL, the variation degree can also be used to perform uniform sampling.

Definition 5 (Uniform Sampler)**.** Given an input feature diagram D, an algorithm $\mathcal{U}$ is a *uniform sampler* iff

$$\forall s \in Sols\,(D)\,, \mathbb{P}\left[\mathcal{U}(D) = s\right] = \frac{1}{|Sols\,(D)|}$$

Remark. *We want to point out that in this definition, $\mathcal{U}$ is not a function in the mathematical sense because it returns different outputs (random configurations) given the same input (a feature diagram).*

The tree-like structure of the feature diagrams can be used to design a recursive sampler.

Proposition 2. *[Uniform Sampler of Feature Diagrams] Given a feature diagram D, the following algorithm $\mathcal{U}^{FD}$ recursively defined is a uniform sampler.*

- *If $D.children = \emptyset$, then $\mathcal{U}^{FD}(D) = \{D.feature\}$*
- *If $D.mand \cup D.opt \neq \emptyset$,*

$$\mathcal{U}^{FD}(D) = \{D.feature\} \cup \bigcup_{D' \in D.mand} \mathcal{U}^{FD}(D') \cup \bigcup_{D' \in D.opt} \begin{cases} \emptyset & \text{with probability } \frac{1}{|Sols(D')|+1} \\ \mathcal{U}^{FD}(D') & \text{otherwise} \end{cases}$$

- *If $D'.xor \neq \emptyset$, choose $D' \in D.xor$ with probability $\frac{|Sols(D')|}{|Sols(D)|}$, then*

$$\mathcal{U}^{FD}(D) = \{D.feature\} \cup \mathcal{U}^{FD}(D')$$

- *If $D.or \neq \emptyset$, we define*

$$C = \bigcup_{D' \in D.or} \begin{cases} \emptyset & \text{with probability } \frac{1}{|Sols(D')|+1} \\ \mathcal{U}^{FD}(D') & \text{otherwise} \end{cases}$$

and

$$\mathcal{U}^{FD}(D) = \begin{cases} \{D.feature\} \cup C & \text{if } C \neq \emptyset \\ \mathcal{U}^{FD}(D) & \text{otherwise} \end{cases}$$

Remark. *In the definition of the uniform sampler $\mathcal{U}^{FD}$, in the $D.or \neq \emptyset$ case, there is a recursive call with the same feature diagram. This is the case where $C = \emptyset$ that is forbidden (at least one child of $D.or$ has to be taken). In this case, we simply generate a new configuration C by calling recursively $\mathcal{U}^{FD}(D)$. The probability that C is empty (i.e. the probability to call $\mathcal{U}^{FD}(D)$ again) is $\frac{1}{|Sols(D)|+1}$, so it is very unlikely to happen.*

Example. *We apply the sampling algorithm to the same example of Figure 1a. Recall that the algorithm does not consider the cross-tree constraint. The algorithm starts at $D_{\texttt{Game}}$ (the feature diagram rooted in the feature* `Game`*). This node is a mandatory/optional node:*

- *A sub-configuration is sampled in the mandatory child ($D_{\texttt{Style}}$). This child is an alternative group, hence only one child is chosen. All the children have the same variation degree (equal to 1), they are all likely to be chosen.*
  - *Suppose that child $D_{\texttt{Racing}}$ is chosen. This child is a leaf node, hence the returned sub-configuration is $\{\texttt{Racing}\}$.*

  The feature `Style` *is added, hence the returned sub-configuration is $\{\texttt{Style}, \texttt{Racing}\}$.*
- *With probability $\frac{3}{4} = 1 - \frac{1}{|Sols(D_{\texttt{Multiplayer}})|}$ the sub-feature diagram $D_{\texttt{Multiplayer}}$ is sampled. Let's suppose that this probability is met. The $D_{\texttt{Multiplayer}}$ feature diagram is an* `or` *node:*
  - *With probability $\frac{1}{2}$ the child $D_{\texttt{Local}}$ is sampled. We suppose that this event does not happen, and that $D_{\texttt{Local}}$ is not chosen, hence the sub-configuration returned is $\{\}$.*
  - *With probability $\frac{1}{2}$ the child $D_{\texttt{Online}}$ is sampled. We suppose that this event does not happen, and that $D_{\texttt{Local}}$ is chosen, hence the sub-configuration returned is $\{\texttt{Online}\}$.*

  We construct the sub-configuration $C = \{\} \cup \{\texttt{Online}\} \neq \emptyset$, hence the returned sub-configuration is $\{\texttt{Multiplayer}, \texttt{Online}\}$

The final configuration returned is $\{\texttt{Style}, \texttt{Racing}, \texttt{Multiplayer}, \texttt{Online}, \texttt{Game}\}$ (union of the sub-configurations of the mandatory and the optional children plus the root node).

References

[Baranov et al., 2020] Baranov, E., Legay, A., and Meel, K. S. (2020). Baital: an adaptive weighted sampling approach for improved t-wise coverage. In Devanbu, P., Cohen, M. B., and Zimmermann, T., editors, *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pages 1114–1126. ACM.

[Fernández-Amorós et al., 2014] Fernández-Amorós, D., Heradio, R., Cerrada, J. A., and Cerrada, C. (2014). A scalable approach to exact model and commonality counting for extended feature models. *IEEE Trans. Software Eng.*, 40(9):895–910.

[Oh et al., 2019a] Oh, J., Gazzillo, P., Batory, D., Heule, M., and Myers, M. (2019a). Uniform sampling from kconfig feature models. *The University of Texas at Austin, Department of Computer Science, Tech. Rep. TR-19*, 2.

[Oh et al., 2019b] Oh, J., Gazzillo, P., and Batory, D. S. (2019b). t-wise coverage by uniform sampling. In Berger, T., Collet, P., Duchien, L., Fogdal, T., Heymans, P., Kehrer, T., Martinez, J., Mazo, R., Montalvillo, L., Salinesi, C., Tërnava, X., Thüm, T., and Ziadi, T., editors, *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume A, Paris, France, September 9-13, 2019*, pages 15:1–15:4. ACM.

[von der Maßen and Lichter, 2005] von der Maßen, T. and Lichter, H. (2005). Determining the variation degree of feature models. In Obbink, J. H. and Pohl, K., editors, *Software Product Lines, 9th International Conference, SPLC 2005, Rennes, France, September 26-29, 2005, Proceedings*, volume 3714 of *Lecture Notes in Computer Science*, pages 82–88. Springer.

A Proofs

A.1 Expansion Operator

To ease the proofs, we first introduce an operator called the expansion.

Definition 6 (Expansion Operator)**.** Let E_1 and E_2 be two sets of configurations, we define the expansion operator as

$$E_1 \diamond E_2 = \bigcup_{\substack{C_1 \in E_1 \\ C_2 \in E_2}} \{C_1 \cup C_2\}$$

Given $E = \{E_1, \ldots, E_n\}$ n sets of configurations, we extend the expansion operator to

$$\bigdiamond_{E_i \in E} E_i = E_1 \diamond \ldots \diamond E_n$$

Remark. *As a consequence of the definition, an expansion on the empty set is:*

$$\bigdiamond_{E_i \in \emptyset} E_i = \{\emptyset\}$$

This definition is similar to the cartesian product but for merging sets of configurations. Informally, if there is a set of configurations E_1 on features $\mathcal{F}_1$, and E_2 on features $\mathcal{F}_2$ then $E_1 \diamond E_2$ is the allowed configurations on $\mathcal{F}_1 \cup \mathcal{F}_2$ (assuming there are no constraints between the features in $\mathcal{F}_1$ and $\mathcal{F}_2$).

This operator allows us to easily recursively compute the set of allowed configurations of a feature diagram.

Proposition 3 (Set of Configurations)**.** *Using the expansion operator, the set of allowed configurations of a feature diagram D can be computed recursively as:*

- *If $D.children = \emptyset$, then $Sols\,(D) = \{\{D\}\}$*
- *If $D.mand \cup D.opt \neq \emptyset$,*

$$Sols(D) = \{\{D.feature\}\} \diamond \bigdiamond_{D' \in D.mand} Sols(D') \diamond \bigdiamond_{D' \in D.opt} Sols(D') \cup \{\emptyset\}$$

- *If* $D.xor \neq \emptyset$,
$$Sols(D) = \{\{D.feature\}\} \diamond \bigcup_{D' \in D.xor} Sols(D')$$
- *If* $D.or \neq \emptyset$,
$$Sols(D) = \{\{D.feature\}\} \diamond \left(\left(\bigdiamond_{D' \in D.or} Sols(D') \cup \{\emptyset\} \right) \setminus \{\emptyset\} \right)$$

Proof. We recall that the expansion operator is the operator for merging sets of configurations. The formula boils down to 5 items:

- $\{\{D.feature\}\} \diamond \ldots$ is the part where the current feature is added to the set of configurations
- $\bigdiamond_{D' \in D.mand} Sols(D')$ is the part where all the configurations of all mandatory children are merged
- $\bigdiamond_{D' \in D.opt} Sols(D') \cup \{\emptyset\}$ is the part for optional children. The singleton containing the empty set is a neutral element for the expansion operator. Adding the empty set to the set of configurations is a way to allow to either take a configuration of the children, or not, which is exactly the definition of optional children.
- $\bigcup_{D' \in D.xor} Sols(D')$ just makes the union of the configuration of children, without the expansion operator because a single configuration is chosen from the *xor* children
- $\left(\bigdiamond_{D' \in D.or} Sols(D') \cup \{\emptyset\} \right) \setminus \{\emptyset\}$ is almost the same as the optional children, except that at least one child has to be chosen, so the empty set is removed.

□

A.2 Variation Degree

Before proving the formula of the variation degree, we show a lemma to show that it is easy to count with the expansion operator.

Lemma 1. *Let* E_1 *and* E_2 *two sets of configurations on different sets of features. Then*
$$|E_1 \diamond E_2| = |E_1| \cdot |E_2|$$

Proof. If the sets of features of E_1 and E_2 are disjoint, then the union in the definition of the expansion operator is a disjoint union. Then
$$\begin{aligned} |E_1 \diamond E_2| &= \left| \bigcup_{\substack{C_1 \in E_1 \\ C_2 \in E_2}} \{C_1 \cup C_2\} \right| \\ &= \sum_{\substack{C_1 \in E_1 \\ C_2 \in E_2}} |\{C_1 \cup C_2\}| \\ &= \sum_{\substack{C_1 \in E_1 \\ C_2 \in E_2}} 1 \\ &= |E_1| \cdot |E_2| \end{aligned}$$

□

Theorem 1. *[Variation Degree of Feature Diagrams [von der Maßen and Lichter, 2005]] Let D be a feature diagram. Then*

- *If $D.children = \emptyset$, then $|Sols(D)| = 1$*
- *If $D.mand \cup D.opt \neq \emptyset$,*

$$|Sols(D)| = \prod_{D' \in D.mand} |Sols(D')| \times \prod_{D' \in D.opt} |Sols(D')| + 1$$

- *If $D.xor \neq \emptyset$,*

$$|Sols(D)| = \sum_{D' \in D.xor} |Sols(D')|$$

- *If $D.or \neq \emptyset$,*

$$|Sols(D)| = \left(\prod_{D' \in D.or} |Sols(D')| + 1 \right) - 1$$

Proof. The proof follows from Lemma 1 and Property 3. All the sub-feature diagrams use disjoint sets of features. □

A.3 Commonalities

Theorem 2. *[Commonalities on Feature Diagrams [Fernández-Amorós et al., 2014]] Let f be a feature and D be a Feature Diagram. We note $\phi_f(D) = |\{C \in Sols(D) \,|\, f \in C\}|$ the number of occurrences of a feature in the set of allowed configurations. Then*

$$\phi_f(D) = \begin{cases} |Sols(D)| & \text{if } D.feature = f \\ \frac{|Sols(D)|}{|Sols(D')|} \cdot \phi_f(D') & \text{if } f \in D' \text{ and } D' \in D.mand \\ \frac{|Sols(D)|}{|Sols(D')|+1} \cdot \phi_f(D') & \text{if } f \in D' \text{ and } D' \in D.opt \text{ or } D' \in D.or \\ \phi_f(D') & \text{if } f \in D' \text{ and } D' \in D.xor \end{cases}$$

The commonality of f in D can then be computed with $\varphi_f(D) = \frac{\phi_f(D)}{|Sols(D)|}$.

Proof. The idea of the proof is the same as Theorem 1, but instead of directly computing the variation degree, we first restrict to the set of allowed configurations that contain the selected feature.

Let D be a feature diagram and f a feature in it. We note $\Phi_f(D)$ the set of allowed configurations of D including f (i.e. $\Phi_f(D) = \{C \in Sols(D) \,|\, f \in C\}$). Then $\phi_f(D) = |\Phi_f(D)|$.

If $f = D.feature$ then all the allowed combinations of D contain f, hence $\phi_f(D) = |Sols(D)|$. Now suppose that f is in some $D' \in D.children$. There are different cases depending on whether D' is in $D.mand, D.opt, D.or$ or $D.xor$:

- If $D' \in D.mand$, we split the formula of Property 3 between D' and the other children of D

$$\begin{aligned}
Sols\left(D\right) =& \{\{D.feature\}\} \diamond Sols\left(D'\right) \diamond \bigdiamond_{D'' \in D.mand \setminus \{D'\}} Sols\left(D''\right) \\
\Phi_f\left(D\right) =& \{\{D.feature\}\} \diamond \Phi_f\left(D'\right) \diamond \bigdiamond_{D'' \in D.mand \setminus \{D'\}} Sols\left(D''\right) \\
\phi_f\left(D\right) =& \phi_f\left(D'\right) \times \prod_{D'' \in D.mand \setminus \{D'\}} \left|Sols\left(D''\right)\right| \\
\phi_f\left(D\right) =& \phi_f\left(D'\right) \cdot \frac{\left|Sols\left(D\right)\right|}{\left|Sols\left(D'\right)\right|}
\end{aligned}$$

- If $D' \in D.opt$ the same reasoning works, just by remarking that $\prod_{D'' \in D.mand \setminus \{D'\}} \left|Sols\left(D''\right)\right| = \frac{|Sols(D)|}{|Sols(D')|+1}$.
- If $D' \in D.or$, we need to remark that removing the empty set does not matter because we are interested in the solutions that *contain* the feature f. The formula of Property 3 for the $D.or$ children becomes the same as the one for $D.opt$ children.
- If $D' \in D.xor$,

$$\begin{aligned}
Sols\left(D\right) =& \{\{D.feature\}\} \diamond \left(Sols\left(D'\right) \cup \bigcup_{D'' \in D.xor \setminus \{D'\}} Sols\left(D''\right) \right) \\
\Phi_f\left(D\right) =& \{\{D.feature\}\} \diamond \Phi_f\left(D'\right) \\
\phi_f\left(D\right) =& \phi_f\left(D'\right)
\end{aligned}$$

□

A.4 Uniform Sampling

To prove the uniformity of $\mathcal{U}^{FD}$, we first need to introduce lemmas to link the expansion operator with sampling.

Lemma 2. *Let E_1 and E_2 two sets of configurations on different sets of features, and $\mathcal{U}^1$ (resp. $\mathcal{U}^2$) a uniform sampler on E_1 (resp. E_2). Then the sampler defined as*

$$\mathcal{U}(E_1 \diamond E_2) = \mathcal{U}^1(E_1) \cup \mathcal{U}^2(E_2)$$

is a uniform sampler.

Proof. Let $C \in E_1 \diamond E_2$, we want to show that

$$\mathbb{P}\left[\mathcal{U}(E_1 \diamond E_2) = C\right] = \frac{1}{|E_1 \diamond E_2|}$$

As E_1 and E_2 have different sets of features, a configuration sampled can be uniquely divided in

two sub-configurations $C = C_1 \cup C_2$ such that $C_1 \in E_1$ and $C_2 \in E_2$. Then

$$\begin{aligned}
\mathbb{P}\left[\mathcal{U}(E_1 \diamond E_2) = C\right] &= \mathbb{P}\left[\mathcal{U}^1(E_1) \cup \mathcal{U}^2(E_2) = C_1 \cup C_2\right] \\
&= \mathbb{P}\left[\mathcal{U}^1(E_1) = C_1 \wedge \mathcal{U}^2(E_2) = C_2\right] \\
&= \mathbb{P}\left[\mathcal{U}^1(E_1) = C_1\right] \cdot \mathbb{P}\left[\mathcal{U}^2(E_2) = C_2\right] && \text{independency} \\
&= \frac{1}{|E_1|} \cdot \frac{1}{|E_2|} \\
&= \frac{1}{|E_1 \diamond E_2|} && \text{by Lemma 1}
\end{aligned}$$

□

Lemma 3. *Let S be a set of n elements, and c an element not in S. If $\mathcal{U}$ is a uniform sampler on S, then $\mathcal{U}'$ defined as*

$$\mathcal{U}'(S \cup \{c\}) = \begin{cases} c & \text{with probability } \frac{1}{n+1} \\ \mathcal{U}(S) & \text{otherwise} \end{cases}$$

Proof. By definition,

$$\mathbb{P}\left[\mathcal{U}'(S \cup \{c\}) = c\right] = \frac{1}{n+1}$$

and

$$\begin{aligned}
\forall s \in S, \mathbb{P}\left[\mathcal{U}'(S \cup \{c\}) = s\right] &= \mathbb{P}\left[\mathcal{U}'(S \cup \{c\}) \neq c\right] \cdot \mathbb{P}\left[\mathcal{U}(S) = s\right] \\
&= \frac{n}{n+1} \cdot \frac{1}{n} \\
&= \frac{1}{n+1}
\end{aligned}$$

□

Lemma 4. *Let S be a set of n elements, and $s \in S$. If $\mathcal{U}$ is a uniform sampler on S. To define $\mathcal{U}'$ on $S \backslash \{c\}$ we first sample s' from S, and define $\mathcal{U}'$ as*

$$\mathcal{U}'(S \backslash \{s\}) = \begin{cases} s' & \text{if } s' \neq s \\ \mathcal{U}'(S \backslash \{s\}) & \text{otherwise} \end{cases}$$

Then, $\mathcal{U}$ is a uniform sampler in the set $S \backslash \{s\}$.

Proof. At each step, there is a probability of $\frac{1}{n}$ of sampling s from S, which we do not want. Otherwise there is $\frac{1}{n}$ chances to pick every other element.

$$\begin{aligned}
\forall s' \in S \backslash \{s\}, \mathbb{P}\left[\mathcal{U}'(S \backslash \{s\}) = s')\right] &= \frac{1}{n} + \frac{1}{n} \cdot \frac{1}{n} + \frac{1}{n} \left(\frac{1}{n}\right)^2 + \dots \\
&= \frac{1}{n} \cdot \sum_{i=0}^{\infty} \left(\frac{1}{n}\right)^i \\
&= \frac{1}{n} \cdot \frac{1}{1 - \frac{1}{n}} \\
&= \frac{1}{n-1}
\end{aligned}$$

□

From these three lemmas we can then prove that the sampler we proposed for feature diagrams is uniform.

Proposition 2. *[Uniform Sampler of Feature Diagrams] Given a feature diagram D, the following algorithm $\mathcal{U}^{FD}$ recursively defined is a uniform sampler.*

- *If $D.children = \emptyset$, then $\mathcal{U}^{FD}(D) = \{D.feature\}$*
- *If $D.mand \cup D.opt \neq \emptyset$,*

$$\mathcal{U}^{FD}(D) = \{D.feature\} \cup \bigcup_{D' \in D.mand} \mathcal{U}^{FD}(D') \cup \bigcup_{D' \in D.opt} \begin{cases} \emptyset & \text{with probability } \frac{1}{|Sols(D')|+1} \\ \mathcal{U}^{FD}(D') & \text{otherwise} \end{cases}$$

- *If $D'.xor \neq \emptyset$, choose $D' \in D.xor$ with probability $\frac{|Sols(D')|}{|Sols(D)|}$, then*

$$\mathcal{U}^{FD}(D) = \{D.feature\} \cup \mathcal{U}^{FD}(D')$$

- *If $D.or \neq \emptyset$, we define*

$$C = \bigcup_{D' \in D.or} \begin{cases} \emptyset & \text{with probability } \frac{1}{|Sols(D')|+1} \\ \mathcal{U}^{FD}(D') & \text{otherwise} \end{cases}$$

and

$$\mathcal{U}^{FD}(D) = \begin{cases} \{D.feature\} \cup C & \text{if } C \neq \emptyset \\ \mathcal{U}^{FD}(D) & \text{otherwise} \end{cases}$$

Proof. We use Property 3 and the previous lemmas.

- If $D.children = \emptyset$, $Sols(D) = \{\{D.feature\}\}$, so there is only one solution to sample.
- If $D.mand \cup D.opt \neq \emptyset$, then we use Lemma 2, and for the $D.opt$ children we also use Lemma 3
- If $D.xor \neq \emptyset$, first a child D' is chosen with probability $\frac{|Sols(D')|}{|Sols(D)|}$, and then a solution is chosen uniformly in D'. The probability to choose any solution is then

$$\begin{aligned} \forall s \in Sols(D), \mathbb{P}\left[\mathcal{U}^{FD} = s\right] &= \mathbb{P}\left[s \in Sols(D') \wedge \mathcal{U}(D') = s\right] \\ &= \mathbb{P}\left[s \in Sols(D')\right] \cdot \mathbb{P}\left[\mathcal{U}(D') = s\right] \\ &= \frac{|Sols(D')|}{|Sols(D)|} \cdot \frac{1}{|Sols(D')|} \\ &= \frac{1}{|Sols(D)|} \end{aligned}$$

- If $D.or \neq \emptyset$, we apply Property 3, lemmas 2 and 4.

□

OUR WORLDWIDE PARTNERS UNIVERSITIES - DOUBLE DEGREE AGREEMENTS

3 CAMPUS, 1 SITE

IMT Atlantique Bretagne–Pays de la Loire – **http://www.imt-atlantique.fr/**

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
T +33 (0)2 29 00 11 11
F +33 (0)2 29 00 10 00

Campus de Nantes
4, rue Alfred Kastler
CS 20722
44307 Nantes Cedex 3
France
T +33 (0)2 51 85 81 00
F +33 (0)2 99 12 70 08

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
T +33 (0)2 99 12 70 00
F +33 (0)2 51 85 81 99

Site de Toulouse
10, avenue Édouard Belin
BP 44004
31028 Toulouse Cedex 04
France
T +33 (0)5 61 33 83 65

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom