Probabilistic Software product lines*

Carlos Camacho[1], Luis Llana[1], Alberto Núñez[1], and Mario Bravetti[2]

[1] Departamento Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
[2] Dipartimento di Informatica - Scienza e Ingegneria, Università di Bologna, Italy /
FOCUS team, INRIA, France
carlos.camacho@ucm.es, llana@ucm.es, alberto.nunez@pdi.ucm.es,
mario.bravetti@unibo.it
Draft: 2018-11-30 08:15

Abstract. We introduce a probabilistic extension of our previous work SPLA: a formal framework to specify and analyze software product lines. We use probabilistic information to identify those features that are more frequently used. This is done by computing the probability of having a feature in a specific software product line, from now on $\mathtt{SPLA}^{\mathcal{P}}$. We redefine the syntax of SPLA to include probabilistic operators and define new operational and denotational semantics. We prove that the expected equivalence between these two semantic frameworks holds. Our probabilistic framework is supported by a set of scripts to show the model behavior. We briefly comment on the characteristics of the scripts and discuss the advantages of using probabilities to quantify the likelihood of having features in potential software product lines.

Keywords; Software Product Lines; Probabilistic Models; Formal Methods; Feature Models

1 Introduction

During the last years, software product lines (in short, SPLs) have become a widely adopted mechanism for efficient software development. The Carnegie Mellon Software Engineering Institute defines an SPL as "a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [1]. Basically, the main goal of SPLs is to increase the productivity for creating software products, which is achieved by selecting those software systems that are better for a specific criterion (e.g. a software system is less expensive than others, it requires less time to be processed, etc.). Currently, different approaches for representing the product

* Research partially supported by the Spanish MINECO/FEDER project DArDOS (TIN2015-65845-C3-1-R) and the Comunidad de Madrid project SICOMORo-CM (S2013/ICE-3006).

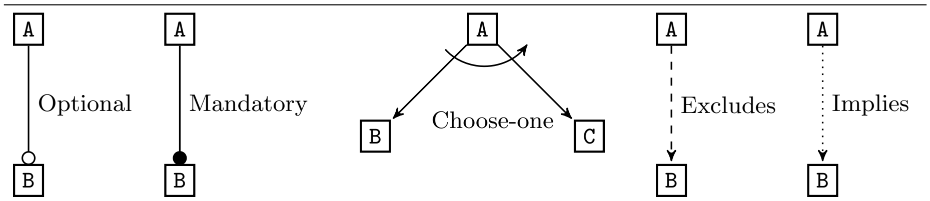

Fig. 1. `FODA` Diagram representation.

line organization can be found in the literature, such as FODA [2], RSEB [3] and PLUSS [4, 5].

Graphical approaches are commonly used to model `SPLs`. Feature Oriented Domain Analysis [2] (in short, `FODA`) is a well-known graphical approach for representing commonality and variability of systems. Figure 1 shows all `FODA` relationships and constraints. Although this kind of solutions is useful to easily model `SPLs`, a formal approach is needed for automatizing the analysis process and detecting errors in the early stages of the production process. It is therefore required that graphical representations are translated into mathematical entities [6]. In this case, the original graphical representation of `FODA` must be provided with a formal semantics [7]. This issue is solved by using `SPLA` [8], a formal framework to represent `FODA` diagrams using process algebras. `SPLA` can be applied not only to `FODA`, but also to represent other feature-related problems and variability models. Additionally, some of the existing formal approaches use algebras and semantics [8–11], while others use either propositional or first order logic [12–16].

It is worth to mention that the order in which features are processed to create a specific product is directly reflected in its final cost. In a previous work we introduced costs in our formal framework for representing the required effort to include a feature to the product under construction [17]. This cost may represent different aspects of a feature, such as lines of code of a given software component or effort, in human hours, to include a software component into a project, just to name a few, that usually depend on the target of the product line organization. Thus, efficiently processing features for building high quality products becomes a time-consuming and challenging task. Unfortunately, there are some situations where the representation of the `SPL` generates a combinatorial explosion, making unpractical to analyze all possible combinations. In order to alleviate this issue, in this paper we propose a probabilistic extension of our previous work `SPLA`. We use probabilistic information to identify those features that are more frequently used by computing the probability of having a feature in a specific `SPL`. Hence, the computation focuses on those features with a high probability to be present in the final product, reducing the total computation required for generating valid products. The proposed probabilistic extension has been fully implemented in a tool. In order to show its usefulness, we have conducted an experimental study where different models consisting of 1500 features have been analyzed. In this case, 450 features have a probability equal to 0.75 to be included in a final product. This means that, at least, the 75% of the generated products are tested by only analyzing the 30% of the features in the `SPL` (450 features).

The main contributions of this work can be summarized as:

- A model that uses probabilistic information to determine the probability of having a feature in a specific `SPL`. In contrast with our previous work [8,17], which mainly focuses in defining an algebraic language to describe Software Product Lines and using a cost model for comparing valid products, this approach is targeted to identify those features that are more frequently used to generate a product. Basically, the idea is to focus on those features with a high probability to be present in the final product and, therefore, reducing the required processing to generate valid products.
- It may be not feasible to compute all the products in a SPL. But if we are interested in a particular feature, we can compute the probability of that feature. The introduction of the notion of hiding sets of features helps us to achieve this. If we want to compute the probability of `A`, we hide the features that do not affect the processing of `A`for being part of a valid product. This analysis allows optimizing the practical application of the probabilistic extension, as it allows us to remove or hide a set of features which does not interfere with the calculus of the probability for a specific feature.
- A thorough empirical study, using different configurations to generate a wide-spectrum of variability models. The study has been carried out in order to show the applicability and scalability of our approach. These variability models have been generated using BeTTy [18].
- A complete comparison between the overall implementation performance of the probabilistic extension and the denotational semantic implementation from `SPLA` [8]. The obtained results show that the denotational semantics of the probabilistic extension implementation dramatically improves the performance in comparison with the denotational semantics implementations presented in previous work.

The rest of the paper is structured as follows. Section 2 introduces the related work on probabilistic analysis of feature models. Section 3 presents our probabilistic language $\texttt{SPLA}^{\mathcal{P}}$. Section 4 is used to prove the equivalence between the operational and denotational semantics. In section 5 we extend our language to define how sets of features can be hidden. This new hidden operator allows to improve the execution of the probabilistic extension execution, as it allows to remove those features that are not required to calculate the probability. Section 6 presents an empirical study that has been carried out by using our implementation of the denotational semantics for the probabilistic extension. Finally, section 7 presents our conclusions and some research lines for the upcoming work.

2 Related work

The study of probabilistic extensions of formal methods can be dated back to the end of the 1980s. This is already a well established area, with many extensive contributions to include probabilistic information in classical formalisms (I/O Automata, Finite State Machines, (co-)algebraic approaches, among others) [19–25]. Although the addition of probabilistic information to model `SPLs` is relatively new, different proposals can be found in the current literature [26–29]. In particular, a very recent work shows that statistic analysis allows users to determine relevant characteristics, like the certainty of finding valid products among complex models [29]. Another approach focuses on testing properties of `SPLs`, like reliability, by defining three verification techniques: a probabilistic model checker on each product, on a model range, and testing the behavior relations with other models [26]. Some of these approaches describe models to run statistical analysis over `SPLs`, where pre-defined syntactic elements are computed by applying a specific set of operational rules [27,28]. These models demonstrate their ability to be integrated into standard tools, like QFLan [28], Microsoft's SMT Z3 [30] and MultiVeStA [31].

Other works focus on describing use cases for analyzing the probability of finding features inside valid products [29]. It is true that variability models computing can create combinatory problems depending on how the models are computed and how the models are represented, which is directly correlated to the information to be generated [29]. This analysis makes the process of studying product lines a complex computational task.

An interesting aspect of $\mathtt{SPLA}^{\mathcal{P}}$is that any of the research articles in the literature manage to describe in their work the use of multisets. Also, they do not explicitly work on the translation of `FODA` to represent probabilities and they do not introduce the notion of hiding those not needed features to calculate the probability of a specific feature.

In particular, there exist mathematical verification models that allow the representation of the products - of a product line - using models, like discrete time Markov chain families, for representing the probabilistic behavior of all the products in the product line [32]. Their implementations do not provide a literal translation between the existing variability models, like FODA, and, therefore, these do not provide implementations over their practical uses. However, these approaches present analysis techniques that should have a significant impact on the effort to compute variability models.

In previous years, the studies focusing the analysis of variability models - and their practical applications - with realistic use cases have demonstrated that those uses cases do not describe such complex models [33, 34]. Thus, these can be processed in the practice without much algorithmic sophistication or complex analysis. In particular, the study of expending machines has been widely used across the whole literature to show practical and real usages of products line modeling [34]. Moreover, it is described that those models for defining products lines does not always apply to the formal definition and description of software product lines, as they are not directly related [16,35,36]. Recent implementations, like ProFeat [37], allow to help in the verification of requirements for families of probabilistic systems. These implementations, together with PRISM [38], use their own language and are based on Markov decision processes.

3 $\texttt{SPLA}^{\mathcal{P}}$: syntax and semantics

In this section we introduce our language. In addition to present its syntax, we define an operational semantics and a denotational semantics. In the next section we will show the equivalence between these two semantic frameworks.

3.1 Syntax and operational semantics

Following our previous work [8,17], we will consider a set of features. We denote this set by $\mathcal{F}$ and consider that $\texttt{A}, \texttt{B}, \texttt{C}$ range over $\mathcal{F}$. We have a special feature $\checkmark \notin \mathcal{F}$ to mark the end of a product. We consider a syntax similar to $\texttt{SPLA}$, where probabilities are introduced both in the choice operator $P \vee_p Q$ and in the optional feature operator $\overline{\texttt{A}};_p P$. We do not allow *degenerated* probabilities, that is, for all probability p we have $0 < p < 1$.

The operators syntax is defined as in [8,17]. In order to define the syntax, we need to fix the set of *features*. From now on $\mathcal{F}$ denotes a finite set of features and $\texttt{A}$, $\texttt{B}$, $\texttt{C}\ldots$ denote isolated features.

In the syntax of the language there are two sets of operators. On the one hand there are *main operators*, such as $\cdot \vee \cdot$, $\cdot \wedge \cdot$, $\texttt{A};\cdot$, $\overline{\texttt{A}};\cdot$, $\texttt{A} \Rightarrow \texttt{B} \texttt{ in } \cdot$, $\texttt{A} \not\Rightarrow \texttt{B} \texttt{ in } \cdot$, that directly correspond to relationships in $\texttt{FODA}$ diagrams. On the other hand, we have *auxiliary operators*, such as $\texttt{nil}$, $\checkmark$, $\cdot\backslash\texttt{A}$, $\cdot \Rightarrow \texttt{A}$, which we need to define the semantics of the language.

Definition 1. A *probabilitistc SPL* is a term generated by the following BNF expression:

$$\begin{aligned} P ::= {} & \checkmark \mid \texttt{nil} \mid \texttt{A};P \mid \overline{\texttt{A}};_p P \mid P \vee_p P \mid P \wedge P \mid \\ & \texttt{A} \not\Rightarrow \texttt{B} \texttt{ in } P \mid \texttt{A} \Rightarrow \texttt{B} \texttt{ in } P \mid P\backslash\texttt{A} \mid P \Rightarrow \texttt{A} \end{aligned}$$

where $\texttt{A}, \texttt{B} \in \mathcal{F}$ and $p \in (0,1)$. The set of terms of the algebra will be denoted by $\texttt{SPLA}^{\mathcal{P}}$. □

In order to avoid writing too many parentheses in the terms, we assume left-associativity in binary operators and the following precedence in the operators (from higher to lower priority): $\texttt{A};P$, $\overline{\texttt{A}};_p P$, $P \vee_p Q$, $P \wedge Q$, $\texttt{A} \not\Rightarrow \texttt{B} \texttt{ in } P$, $\texttt{A} \Rightarrow \texttt{B} \texttt{ in } P$, $\texttt{A} \Rightarrow \texttt{B} \texttt{ in } P$, $P\backslash\texttt{A}$, and $P \Rightarrow \texttt{A}$. In [8,17] we demonstrated that the binary operators are commutative and associative. As a result, the *choose-one* operator $(\cdot \vee_p \cdot)$ and the *conjunction* operator $(\cdot \wedge \cdot)$ are n-ary operators instead of just binary operators.

There are two terminal symbols in the language, $\texttt{nil}$ and $\checkmark$, we need them to define the semantics of the language. Let us note that the products of a term in $\texttt{SPLA}$ will be computed following some rules. The computation will finish when no further steps are allowed. This fact is represented by the $\texttt{nil}$ symbol.

We will introduce rules to compute a product, with this computation finishing when no further steps are required, a situation represented by `nil`. During the computation of an $\mathtt{SPLA}^{\mathcal{P}}$ term, we have to represent the situation in which a *valid product* of the term has been computed. This fact is represented by the $\checkmark$ symbol.

The operators $\mathtt{A};P$ and $\overline{\mathtt{A}}_p;P$ add the feature $\mathtt{A}$ to any product that can be obtained from P. The operator $\mathtt{A};P$ indicates that $\mathtt{A}$ is mandatory while $\overline{\mathtt{A}}_p;P$ indicates that $\mathtt{A}$ is optional and computed with probability p. There are two binary operators: $P \vee_p Q$ and $P \wedge Q$. The first one represents a probabilistic choice. It represents a point in the product line between two options. In this probabilistic framework, the choice is quantified with a probability p: the probability of choosing the left hand side is p and the probability of choosing the right hand side is $1-p$. The operator $P \wedge Q$ is the conjunction, intuitively it combines the products of both subterms P and Q by accumulating the features.

Example 1. Let us consider the term $P = \mathtt{A};\checkmark \vee_{\frac{1}{3}} \mathtt{B};\checkmark$. This term will produce two products: $\{\mathtt{A}\}$ with probability $\frac{1}{3}$ and $\{\mathtt{B}\}$ with probability $\frac{2}{3}$. Let us consider $Q = \mathtt{C};\overline{\mathtt{D}}_{\frac{1}{5}};\checkmark$. This term will produce two products: $\{\mathtt{C}\}$ with probability $\frac{4}{5}$ and $\{\mathtt{C},\mathtt{D}\}$ with probability $\frac{1}{5}$. Then $P \wedge Q$ will produce the following products: $\{\mathtt{A},\mathtt{C}\}$ with probability $\frac{4}{15}$, $\{\mathtt{A},\mathtt{C},\mathtt{D}\}$ with probability $\frac{1}{15}$, $\{\mathtt{B},\mathtt{C}\}$ with probability $\frac{8}{15}$, and $\{\mathtt{A},\mathtt{C},\mathtt{D}\}$ with probability $\frac{2}{15}$.

The constraints are easily represented in $\mathtt{SPLA}^{\mathcal{P}}$. The operator $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P$ represents the *require* constraint in FODA. The operator $\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P$ represents the *exclusion* constraint in FODA.

Example 2. The term $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ \mathtt{A};\checkmark$ has only one valid product $\{\mathtt{A},\mathtt{B}\}$ with probability 1.

Let us consider $P = \mathtt{A};(\mathtt{B};\checkmark \vee_{\frac{1}{3}} \mathtt{C};\checkmark)$. This term has two valid products: The first one $\{\mathtt{A},\mathtt{B}\}$ with probability $\frac{1}{3}$, and $\{\mathtt{A},\mathtt{C}\}$ with probability $\frac{2}{3}$.

If we add to the previous term the following constraint $\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P$, then this new term has only one $\{\mathtt{A},\mathtt{C}\}$ with with probability $\frac{2}{3}$. This term has probability $\frac{1}{3}$ of producing nothing. $\square$

The operator $P \Rightarrow \mathtt{A}$ is necessary to define the behavior of the $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P$ operator: when we compute the products of the term $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P$, we have to take into account whether product $\mathtt{A}$ has been produced or not. In the case it has been produced, we have to annotate that we need to produce $\mathtt{B}$ in the future. The operator $P \Rightarrow \mathtt{B}$ is used for this purpose. The same happens with the operator $P \backslash \mathtt{B}$. When we compute the products of $\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P$, if the feature $\mathtt{A}$ is computed at some point, we annotate that $\mathtt{B}$ must not be included. The operator $P \backslash \mathtt{B}$ indicates that product $\mathtt{B}$ is forbidden.

The rules in Figure 2 define the behavior of $\mathtt{SPLA}^{\mathcal{P}}$ terms. These rules essentially coincide with the ones corresponding to SPLA [8] (with the modification introduced in [17]). We have adapted those rules in order to incorporate probabilities.

[**tick**]	$\checkmark \xrightarrow{\checkmark}_1 \mathtt{nil}$	[**feat**]	$\mathtt{A}; P \xrightarrow{\mathtt{A}}_1 P$
[**ofeat1**]	$\overline{\mathtt{A}};_p P \xrightarrow{\mathtt{A}}_p P$	[**ofeat2**]	$\overline{\mathtt{A}};_p P \xrightarrow{\checkmark}_{(1-p)} \mathtt{nil}$
[**cho1**]	$\dfrac{P \xrightarrow{\mathtt{A}}_p P_1}{P \vee_q Q \xrightarrow{\mathtt{A}}_{p\cdot q} P_1}$	[**cho2**]	$\dfrac{Q \xrightarrow{\mathtt{A}}_q Q_1}{P \vee_p Q \xrightarrow{\mathtt{A}}_{(1-p)\cdot q} Q_1}$
[**con1**]	$\dfrac{P \xrightarrow{\mathtt{A}}_p P_1}{P \wedge Q \xrightarrow{\mathtt{A}}_{\frac{p}{2}} P_1 \wedge Q}$	[**con2**]	$\dfrac{Q \xrightarrow{\mathtt{A}}_q Q_1}{P \wedge Q \xrightarrow{\mathtt{A}}_{\frac{q}{2}} P \wedge Q_1}$
[**con3**]	$\dfrac{P \xrightarrow{\checkmark}_q \mathtt{nil}, Q \xrightarrow{\checkmark}_p \mathtt{nil}}{P \wedge Q \xrightarrow{\checkmark}_{p\cdot q} \mathtt{nil}}$		
[**con4**]	$\dfrac{P \xrightarrow{\mathtt{A}}_p P_1, Q \xrightarrow{\checkmark}_q \mathtt{nil}}{P \wedge Q \xrightarrow{\mathtt{A}}_{\frac{p\cdot q}{2}} P_1}$	[**con5**]	$\dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}, Q \xrightarrow{\mathtt{A}}_q Q_1}{P \wedge Q \xrightarrow{\mathtt{A}}_{\frac{p\cdot q}{2}} Q_1}$
[**req1**]	$\dfrac{P \xrightarrow{\mathtt{C}}_p P_1,\ \mathtt{C} \neq \mathtt{A}}{\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{C}}_p \mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P_1}$	[**req2**]	$\dfrac{P \xrightarrow{\mathtt{A}}_p P_1}{\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{A}}_p P_1 \Rightarrow \mathtt{B}}$
[**req3**]	$\dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}}{\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\checkmark}_p \mathtt{nil}}$		
[**excl1**]	$\dfrac{P \xrightarrow{\mathtt{C}}_p P_1,\ \mathtt{C} \neq \mathtt{A},\ \mathtt{C} \neq \mathtt{B}}{\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{C}}_p \mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P_1}$	[**excl2**]	$\dfrac{P \xrightarrow{\mathtt{A}}_p P_1}{\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{A}}_p P_1 \backslash \mathtt{B}}$
[**excl3**]	$\dfrac{P \xrightarrow{\mathtt{B}}_p P_1}{\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{B}}_p P_1 \backslash \mathtt{A}}$	[**excl4**]	$\dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}}{\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\checkmark}_p \mathtt{nil}}$
[**forb1**]	$\dfrac{P \xrightarrow{\mathtt{B}}_p P_1,\ \mathtt{B} \neq \mathtt{A}}{P \backslash \mathtt{A} \xrightarrow{\mathtt{B}}_p P_1 \backslash \mathtt{A}}$	[**forb2**]	$\dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}}{P \backslash \mathtt{A} \xrightarrow{\checkmark}_p \mathtt{nil}}$
[**mand1**]	$\dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}}{P \Rightarrow \mathtt{A} \xrightarrow{\mathtt{A}}_p \checkmark}$	[**mand2**]	$\dfrac{P \xrightarrow{\mathtt{A}}_p P_1}{P \Rightarrow \mathtt{A} \xrightarrow{\mathtt{A}}_p P_1}$
[**mand3**]	$\dfrac{P \xrightarrow{\mathtt{B}}_p P_1,\ \mathtt{A} \neq \mathtt{B}}{P \Rightarrow \mathtt{A} \xrightarrow{\mathtt{B}}_p P_1 \Rightarrow \mathtt{A}}$		

$\mathtt{A}, \mathtt{B}, \mathtt{C} \in \mathcal{F},\ a \in \mathcal{F} \cup \{\checkmark\}$

Fig. 2. $\mathtt{SPLA}^{\mathcal{P}}$ operational semantics.

Definition 2. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$ two terms, $\mathtt{A} \in \mathcal{F}$ and a probability $p \in (0, 1]$ we define the transition $P \xrightarrow{\mathtt{A}}_p Q$ iff can be deduced in a finite number of steps from the rules in Figure 2. □

Next we focus on the explanation of the role of probabilities. Rules [**tick**] and [**feat**] show the corresponding feature with probability 1. Rules [**ofeat1**] and [**ofeat2**] deal with the probabilistic optional feature. The feature can be chosen with probability p and can be rejected with probability $1 - p$. Let us note that both probabilities are not null. Rules [**cho1**] and [**cho2**] define the behavior of the probabilistic choice operator. The left branch is selected with probability p and the right one with probability $1 - p$. It is important to note that the rules for the conjunction operator, [**con1**], [**con2**], [**con4**] and [**con5**], equitably distribute the probability between both branches, that is, $\frac{1}{2}$. We have preferred to use a simple definition of this operator, but it is easy to replace it by a more involved version of a probabilistic conjunction operator [39]. Rule [**con3**]

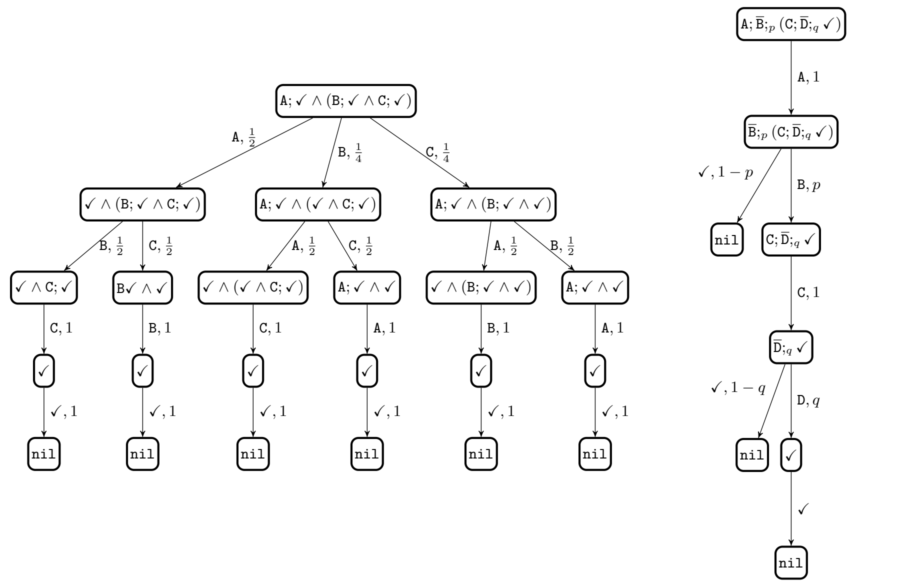

Fig. 3. Examples of the operational semantics (1/2).

requires that both branches agree on the termination of a product. Figures 3 and 4 contain some examples of the operational semantics.

We use *multisets* of transitions to consider different occurrences of the same transition. Thus, if a transition can be derived in several ways, then each derivation generates a different instance of this transition [40]. For example, let us consider the term $P = \mathtt{A};\checkmark \vee_{\frac{1}{2}} \mathtt{A};\checkmark$. If we were not careful, then we would have the transition $P \xrightarrow{\mathtt{A}}_{\frac{1}{2}} \checkmark$ only once, while we should have this transition twice. So, if a transition can be derived in several ways, then we consider that each derivation generates a different instance. In particular, we will later consider multisets of computations as well. We will use the delimiters $\wr$ and $\int$ to denote multisets and $\uplus$ to denote the union of multisets.

The following result, whose proof is immediate, shows that successful termination leads to $\mathtt{nil}$.

Lemma 1. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$ and $p \in \mathbb{R}$. We have $P \xrightarrow{\checkmark}_p Q$ if and only if $Q = \mathtt{nil}$. $\square$

Next we present some notions associated with the composition of consecutive transitions.

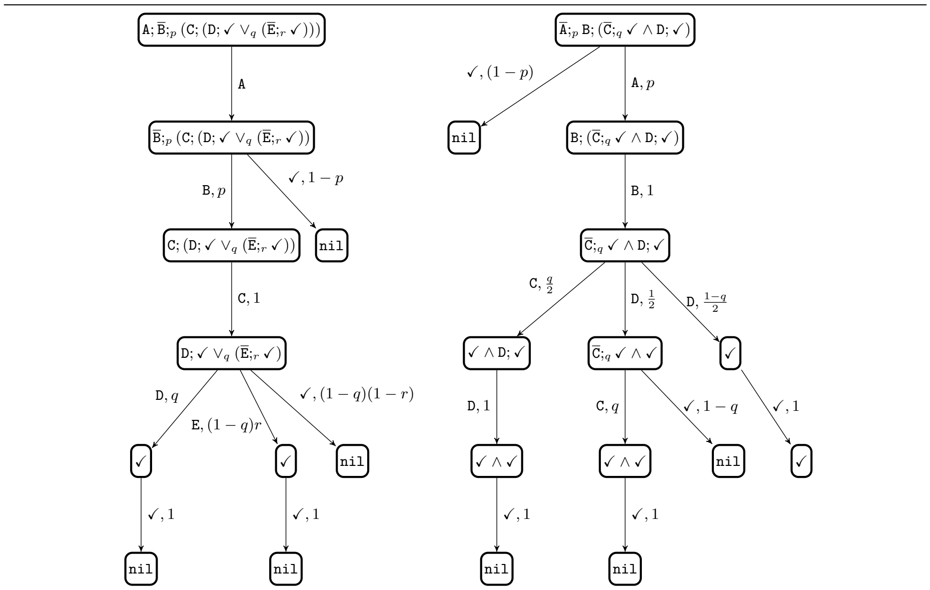

Fig. 4. Examples of the operational semantics (2/2).

Definition 3. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$. We write $P \stackrel{s}{\Longrightarrow}_p Q$ if there exists a sequence of consecutive transitions

$$P = P_0 \xrightarrow{a_1}_{p_1} P_1 \xrightarrow{a_2}_{p_2} P_2 \cdots P_{n-1} \xrightarrow{a_n}_{p_n} P_n = Q$$

where $n \geq 0$, $s = a_1 a_2 \cdots a_n$ and $p = p_1 \cdot p_2 \cdot \cdots \cdot p_n$. We say that s is a trace of P.

Let $s \in \mathcal{F}^*$ be a trace of P. We define the product $\lfloor s \rfloor \subseteq \mathcal{F}$ as the set consisting of all features belonging to s.

Let $P \in \mathtt{SPLA}^{\mathcal{P}}$. We define the set of probabilistic products of P, denoted by $\mathtt{prod}^{\mathcal{P}}(P)$, as the set

$$\mathtt{prod}^{\mathcal{P}}(P) = \{(pr, p) \mid p > 0 \wedge p = \sum \wr q \mid P \stackrel{s\checkmark}{\Longrightarrow}_q Q \;\wedge\; \lfloor s \rfloor = pr \int\}$$

We define the total probability of P, denoted by $\mathtt{TotProb}(P)$, as the value $\sum \wr p \mid \exists pr : (pr, p) \in \mathtt{prod}^{\mathcal{P}}(P) \int$. In addition, we define $\mathtt{waste}(P) = 1 - \mathtt{TotProb}(P)$. $\square$

The following result shows some properties, concerning probabilities, of the operational semantics. In particular, we have that the probability of (sequences of) transitions is greater than zero.

Lemma 2. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$, we have the following results.

1. If $P \xrightarrow{\mathtt{A}}_p Q$ then $p \in (0, 1]$. If $P \stackrel{s}{\Longrightarrow}_p Q$ then $p \in (0, 1]$.

2. $\sum \lbag p \mid \exists \mathtt{A} \in \mathcal{F},\ Q \in \mathtt{SPLA}^{\mathcal{P}} :\ P \xrightarrow{\mathtt{A}}_p Q \rbag \in [0,1]$.
3. $\sum \lbag p \mid \exists s \in \mathcal{F}^*,\ Q \in \mathtt{SPLA}^{\mathcal{P}} :\ P \overset{s\checkmark}{\Longrightarrow}_p Q \rbag \in [0,1]$.
4. $\mathtt{TotProb}(P) \in [0,1]$.

$\square$

Next we prove an important property of our language: its consistency. We say that a non-probabilistic SPL model is *consistent* if it has products [8]. In our case, we can define consistency by having $\mathtt{TotProb}(P) > 0$. We will prove that a translation from our probabilistic framework into the non-probabilistic one keeps consistency in the expected way.

Definition 4. We define the translation function $\mathtt{np} : \mathtt{SPLA}^{\mathcal{P}} \mapsto \mathtt{SPLA}$ as follows:

$$\mathtt{np}(P) = \begin{cases} \checkmark & \text{if } P = \checkmark \\ \mathtt{nil} & \text{if } P = \mathtt{nil} \\ \mathtt{A}; \mathtt{np}(P) & \text{if } P = \mathtt{A}; P \\ \overline{\mathtt{A}}; \mathtt{np}(P) & \text{if } P = \overline{\mathtt{A}};_p P \\ \mathtt{np}(P) \vee \mathtt{np}(Q) & \text{if } P \vee_p Q \\ \mathtt{np}(P) \wedge \mathtt{np}(Q) & \text{if } P \wedge Q \\ \mathtt{A} \Rightarrow \mathtt{B} \ \mathtt{in}\ \mathtt{np}(P) & \text{if } \mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \\ \mathtt{A} \not\Rightarrow \mathtt{B} \ \mathtt{in}\ \mathtt{np}(P) & \text{if } \mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P \\ \mathtt{np}(P) \Rightarrow \mathtt{A} & \text{if } P \Rightarrow \mathtt{A} \\ \mathtt{np}(P) \backslash \mathtt{A} & \text{if } P \backslash \mathtt{A} \end{cases}$$

$\square$

The proof of the following result is straightforward by taking into account that, if we discard probabilities, our operational semantics rules are the same as in [8]. Therefore, any sequence of transitions derived in the probabilistic model can be also derived in the non probabilistic one. In addition, by Lemma 2 we know that any derived trace in the probabilistic model has a non null probability.

Theorem 1. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$. We have $P \overset{s}{\Longrightarrow}_p Q$ if and only if $\mathtt{np}(P) \overset{s}{\Longrightarrow} \mathtt{np}(Q)$. Moreover, we have $pr \in \mathtt{prod}(\mathtt{np}(P))$ if and only if there exists $p > 0$ such that $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(P)$. $\square$

3.2 Denotational Semantics

Next we define a denotational semantics for the terms of our language. The main characteristic of the semantic domain is that we consider products (set of features) with a probability such that the sum of all the probabilities associated with products belongs to the interval $(0, 1]$. First, we precisely define the members of the semantic domain.

Definition 5. We define the semantic domain $\mathcal{M}$ as the largest set $\mathcal{M} \subseteq \mathcal{P}(\mathcal{P}(\mathcal{F}) \times (0, 1]))$ such that if $A \in \mathcal{M}$ then the following conditions hold:

- If $(P,q) \in A$ and $(P,r) \in A$ then $q = r$.
- $0 \leq \sum \wr q \mid \exists P : (P,q) \in A \int \leq 1$.

Let M be a multiset with elements in the set $\mathcal{P}(\mathcal{F}) \times [0,1]$. We define the operator `accum` as follows:

$$\texttt{accum}(M) = \left\{ (P,p) \;\middle|\; p = \sum_{(P,q)\in M} q \wedge p > 0 \right\}$$

□

Even though the elements of the semantic domain are sets of pairs (product, probability), with at most one occurrence of a given product, we use multisets as auxiliary elements in our semantic functions. Then, the function $\texttt{accum}(M)$ will *flatten* them to become sets. The following result is immediate.

Proposition 1. Let M be a multiset with elements in the set $\mathcal{P}(\mathcal{F}) \times [0,1]$. If $1 \geq \sum \wr q \mid (P,q) \in M \int$ then $\texttt{accum}(M) \in \mathcal{M}$. □

Next we define the operators of the denotational semantics (called denotational operators). As we have said before, multisets meeting the conditions of the previous result appear when defining these operators. For instance, the prefix operator $[\![\texttt{A};]\!](M)$ should add feature $\texttt{A}$ to any product in M. Let us suppose that $M = \{(\{\texttt{B},\texttt{A}\}, \frac{1}{2}), (\{\texttt{B}\}, \frac{1}{2})\}$. If we add $\texttt{A}$ to the products of M then we obtain the product $\{\texttt{A},\texttt{B}\}$ twice, having probability $\frac{1}{2}$ associated with each occurrence. So we need to apply the function `accum` to accumulate both probabilities and to obtain a single product with probability 1.

Definition 6. Let $M, M_1, M_2 \in \mathcal{M}$, $\texttt{A},\texttt{B} \in \mathcal{F}$ and $p \in (0,1]$. For any operator appearing in Definition 1 we define its denotational operator as follows:

- $[\![\texttt{nil}]\!]^{\mathcal{P}} = \varnothing$
- $[\![\checkmark]\!]^{\mathcal{P}} = \{(\varnothing, 1)\}$
- $[\![\texttt{A};\cdot]\!]^{\mathcal{P}}(M) = \texttt{accum}\big(\wr(\{\texttt{A}\} \cup P, p) \mid (P,p) \in M \int\big)$
- $[\![\overline{\texttt{A}};_p \cdot]\!]^{\mathcal{P}}(M) = \texttt{accum}\big(\wr(\varnothing, 1-p) \int \uplus \wr (\{\texttt{A}\} \cup P, p \cdot q) \mid (P,q) \in M \int\big)$
- $[\![\cdot \vee_p \cdot]\!]^{\mathcal{P}}(M_1, M_2) = \texttt{accum}\begin{pmatrix} \wr(P, p\cdot q) \mid (P,q) \in M_1 \int \uplus \\ \wr(Q, (1-p)\cdot q) \mid (Q,q) \in M_2 \int \end{pmatrix}$
- $[\![\cdot \wedge \cdot]\!]^{\mathcal{P}}(M_1, M_2) = \texttt{accum}\Big(\wr(P \cup Q, p \cdot q) \mid (P,p) \in M_1,\ (Q,q) \in M_2 \int\Big)$
- $[\![\texttt{A} \Rightarrow \texttt{B in} \cdot]\!]^{\mathcal{P}}(M) = \texttt{accum}\begin{pmatrix} \wr\big(P,p\big) \mid (P,p) \in M, \texttt{A} \notin P \int \ \uplus \\ \wr\big(\{\texttt{B}\} \cup P, p\big) \mid (P,p) \in M, \texttt{A} \in P \int \end{pmatrix}$
- $[\![\texttt{A} \not\Rightarrow \texttt{B in} \cdot]\!]^{\mathcal{P}}(M) = \{(P,p) \mid (P,p) \in M, \texttt{A} \notin P\} \cup$
  $\{(P,p) \mid (P,p) \in M, \texttt{B} \notin P\}$
- $[\![\cdot \Rightarrow \texttt{A}]\!]^{\mathcal{P}}(M) = [\![\texttt{A};\cdot]\!]^{\mathcal{P}}(M)$
- $[\![\cdot \backslash \texttt{A}]\!]^{\mathcal{P}}(M) = \{(P,p) \mid (P,p) \in M, \texttt{A} \notin P\}$

□

The denotational semantics for the prefix operator $[\![\mathtt{A};\cdot]\!]^{\mathcal{P}}(M)$ and the denotational semantics for the operator $[\![\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ \cdot]\!]^{\mathcal{P}}(M)$ behave in the same way if the feature is added to the products. In the first case the feature A is mandatory so it will be added, and in the second case the feature B is required if the feature A is already included in the product.

It is easy to check that all the multisets appearing in the previous definition meet the conditions of Proposition 1. Thus, the operators are actually well defined. This is formalized in the following result.

Proposition 2. Let $M, M_1, M_2 \in \mathcal{M}$, $p \in (0,1]$ be a probability, and $\mathtt{A}, \mathtt{B} \in \mathcal{F}$ be features. We have:

- $[\![\mathtt{A};\cdot]\!]^{\mathcal{P}}(M) \in \mathcal{M}$
- $[\![\overline{\mathtt{A}};_p \cdot]\!]^{\mathcal{P}}(M) \in \mathcal{M}$
- $[\![\cdot \vee_p \cdot]\!]^{\mathcal{P}}(M_1, M_2) \in \mathcal{M}$
- $[\![\cdot \wedge \cdot]\!]^{\mathcal{P}}(M_1, M_2) \in \mathcal{M}$
- $[\![\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ \cdot]\!]^{\mathcal{P}}(M) \in \mathcal{M}$
- $[\![\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ \cdot]\!]^{\mathcal{P}}(M) \in \mathcal{M}$
- $[\![\cdot \Rightarrow \mathtt{A}]\!]^{\mathcal{P}}(M) \in \mathcal{M}$
- $[\![\cdot \backslash \mathtt{A}]\!]^{\mathcal{P}}(M) \in \mathcal{M}$

□

4 Equivalence between the operational and denotational semantics

We have defined two different semantics for our language: the products derived from the operational semantics and the products obtained from the denotational semantics. It is important that both semantics are consistent, so that we can chose the approach that suits better in any moment.

Proposition 3. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$ be terms, $\mathtt{A}, \mathtt{B} \in \mathcal{F}$ be features and $q \in (0,1)$, be a probability. We have the following results:

$$
\begin{aligned}
\mathtt{prod}^{\mathcal{P}}(\mathtt{A};P) &= [\![\mathtt{A};\cdot]\!]^{\mathcal{P}}(\mathtt{prod}^{\mathcal{P}}(P)) && (1)\\
\mathtt{prod}^{\mathcal{P}}(\overline{\mathtt{A}};_q P) &= [\![\overline{\mathtt{A}};_q \cdot]\!]^{\mathcal{P}}(\mathtt{prod}^{\mathcal{P}}(P)) && (2)\\
\mathtt{prod}^{\mathcal{P}}(P \vee_q Q) &= [\![\cdot \vee_q \cdot]\!]^{\mathcal{P}}\big(\mathtt{prod}^{\mathcal{P}}(P), \mathtt{prod}^{\mathcal{P}}(Q)\big) && (3)\\
\mathtt{prod}^{\mathcal{P}}(P \wedge Q) &= [\![\cdot \wedge \cdot]\!]^{\mathcal{P}}\big(\mathtt{prod}^{\mathcal{P}}(P), \mathtt{prod}^{\mathcal{P}}(Q)\big) && (4)\\
\mathtt{prod}^{\mathcal{P}}(P \Rightarrow \mathtt{A}) &= [\![\cdot \Rightarrow \mathtt{A}]\!]^{\mathcal{P}}(\mathtt{prod}^{\mathcal{P}}(P)) && (5)\\
\mathtt{prod}^{\mathcal{P}}(P \backslash \mathtt{A}) &= [\![\cdot \backslash \mathtt{A}]\!]^{\mathcal{P}}(\mathtt{prod}^{\mathcal{P}}(P)) && (6)\\
\mathtt{prod}^{\mathcal{P}}(\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P) &= [\![\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ \cdot]\!]^{\mathcal{P}}(\mathtt{prod}^{\mathcal{P}}(P)) && (7)\\
\mathtt{prod}^{\mathcal{P}}(\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P) &= [\![\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ \cdot]\!]^{\mathcal{P}}(\mathtt{prod}^{\mathcal{P}}(P)) && (8)
\end{aligned}
$$

Proof. The full proof of this Proposition is in Appendix A. Each equality above is proved in a different Lemma: (1) is consequence of Lemma 3, (2) is consequence of Lemma 4, (3) is consequence of Lemma 6, (4) is consequence of Lemma 8, (5) is consequence of Lemma 9, (6) is consequence of Lemma 12, (7) is consequence of Lemma 11, and (8) is consequence of Lemma 12.

□

$$[\mathbf{hid1}]\ \frac{P \xrightarrow{\mathtt{A}}_p P', \mathtt{A} \in \mathcal{A}}{P[\mathcal{A}] \xrightarrow{\perp}_p P'[\mathcal{A}]} \qquad\qquad [\mathbf{hid2}]\ \frac{P \xrightarrow{\mathtt{A}}_p P', \mathtt{A} \notin \mathcal{A}}{P[\mathcal{A}] \xrightarrow{\mathtt{A}}_p P'[\mathcal{A}]}$$

Fig. 5. Operational semantics for the hiding operator

The definition of the operator $[\![\cdot \wedge \cdot]\!]^{\mathcal{P}}$ is clearly associative and commutative. Then, as consequence of the previous proposition, the semantics of the conjunction operator $\wedge$ is associative and commutative.

Finally, we have the previously announced result. The proof, by structural induction on P, is easy from Proposition 3.

Theorem 2. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$ be a term, $pr \subseteq \mathcal{F}$ be a product, and $p \in (0, 1]$ be a probability. We have that $(pr, p) \in [\![P]\!]^{\mathcal{P}}$ if and only if $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(P)$. □

5 Hiding sets of features

The probability of a single feature in a software product line is a measure of the occurrences of this feature in the set of products. For instance, in case of testing, it is interesting to know the most frequent components to focus our analysis on these components. In order to compute the probability of a set of features, other features from the software product line are *hidden*. We hide features because it is usually not feasible to compute all the products of the software product line. However, we expect to achieve our goal if we restrict ourselves to a subset of features. Thus, non interesting features are transformed into a new feature, denoted by $\perp \notin \mathcal{F}$, and we consider the set $\mathcal{F}_\perp = \mathcal{F} \cup \{\perp\}$.

We extend the set of operators with a new one: hiding a set of features in a term.

Definition 7. Let $\mathcal{A} \subseteq \mathcal{F}$ be a subset of features and $P \in \mathtt{SPLA}^{\mathcal{P}}$ be a term. We have that $P[\mathcal{A}]$ denotes the hiding of the features in $\mathcal{A}$ for the term P. □

We need to define the semantics of the new operator. The operational semantics is given by the rules appearing in Figure 5. In order to define the denotational semantics of the new operator, first we need an auxiliary function that hides some features of a given product.

Definition 8. Let $pr \subseteq \mathcal{F}$ be a product and $\mathcal{A} \subseteq \mathcal{F}$ be a set of features. The *hiding of the set $\mathcal{A}$ in pr*, denoted by $pr[\mathcal{A}]$, is defined as follows:

$$pr[\mathcal{A}] = \{\mathtt{A} \mid \mathtt{A} \in pr \wedge \mathtt{A} \notin \mathcal{A}\} \cup \begin{cases} \{\perp\} & \text{if } pr \cap \mathcal{A} \neq \varnothing \\ \varnothing & \text{if } pr \cap \mathcal{A} = \varnothing \end{cases}$$

Analogously, for any sequence $s \in \mathcal{F}^*$ we consider that $s[\mathcal{A}]$ denotes the trace produced from s after replacing all the occurrences of features belonging to $\mathcal{A}$ by the symbol $\perp$ in s. $\square$

Definition 9. Let $M \in \mathcal{M}$ and $\mathcal{A} \subseteq \mathcal{F}$. We define:

$$[\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}(M) = \mathtt{accum}\Big(\wr (pr[\mathcal{A}], p) \mid (pr, p) \in M \int\Big)$$

$\square$

Finally, we have to prove that the operational semantics and the denotational semantics are consistent. The proof of the following result is an immediate consequence of Proposition 6 (see Appendix B).

Proposition 4. Let $\mathcal{A} \subseteq \mathcal{F}$ be a subset of features and $P \in \mathtt{SPLA}^{\mathcal{P}}$ be a term. We have $\mathtt{prod}^{\mathcal{P}}(P[\mathcal{A}]) = [\![\mathtt{prod}^{\mathcal{P}}(P)[\mathcal{A}]]\!]^{\mathcal{P}}$. $\square$

As usual in process algebras, it would be desirable that the hiding operator is *derived*, that is, given a syntactic term, there exists a semantically equivalent term without occurrences of the hiding operator. The idea is to substitute any occurrence of the hidden actions by the symbol $\perp$. However, it is necessary to take into account that we cannot hide actions that appear in the restriction operators and, therefore, these cases are not contemplated.

Proposition 5. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$ be terms, $r \in (0, 1]$ be a probability, and $\mathcal{A} \subseteq \mathcal{F}$ be a set of hidden actions. We have the following results:

$$\begin{array}{r}
[\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}([\![\checkmark]\!]^{\mathcal{P}}) = [\![\checkmark]\!]^{\mathcal{P}} \\
[\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}([\![\mathtt{nil}]\!]^{\mathcal{P}}) = [\![\mathtt{nil}]\!]^{\mathcal{P}} \\
[\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}([\![\mathtt{A}; P]\!]^{\mathcal{P}}) = \begin{cases} [\![\perp; (P[\mathcal{A}])]\!]^{\mathcal{P}} & \text{if } A \in \mathcal{A} \\ [\![\mathtt{A}; (P[\mathcal{A}])]\!]^{\mathcal{P}} & \text{if } A \notin \mathcal{A} \end{cases} \\
[\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}([\![\overline{\mathtt{A}};_r P]\!]^{\mathcal{P}}) = \begin{cases} [\![\overline{\perp};_r (P[\mathcal{A}])]\!]^{\mathcal{P}} & \text{if } A \in \mathcal{A} \\ [\![\overline{\mathtt{A}};_r (P[\mathcal{A}])]\!]^{\mathcal{P}} & \text{if } A \notin \mathcal{A} \end{cases} \\
[\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}([\![P \vee_P Q]\!]^{\mathcal{P}}) = [\![(P[\mathcal{A}]) \vee_P (Q[\mathcal{A}])]\!]^{\mathcal{P}} \\
[\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}([\![P \wedge Q]\!]^{\mathcal{P}}) = [\![(P[\mathcal{A}]) \wedge (Q[\mathcal{A}])]\!]^{\mathcal{P}} \\
\text{If } \mathtt{A}, \mathtt{B} \notin \mathcal{A} \text{ then } [\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}([\![\mathtt{A} \Rightarrow \mathtt{B} \ \mathtt{in}\ P]\!]^{\mathcal{P}}) = [\![\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ (P[\mathcal{A}])]\!]^{\mathcal{P}} \\
\text{If } \mathtt{A}, \mathtt{B} \notin \mathcal{A} \text{ then } [\![\cdot[\mathcal{A}]]\!]^{\mathcal{P}}([\![\mathtt{B} \not\Rightarrow \mathtt{P}\ \mathtt{in}\ ]\!]^{\mathcal{P}}) = [\![\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ (P[\mathcal{A}])]\!]^{\mathcal{P}}
\end{array}$$

Proof. The proof is immediate applying the definitions and Proposition 4.

$\square$

6 Empirical study

In the field of SPLs analysis, the use of probabilistic methods carries two practical applications. The first one consists in calculating the probability of having a feature in a specific product. This allows us to efficiently assign resources by prioritizing those features with a high probability of being included into the SPL. The second application consists in estimating the testing coverage in the product line, which allows us to calculate those products that can be generated in the testing process.

The idea to compute the probability of each feature is to hide all the other features and then compute the resulting SPL. This approach is based on Proposition 4. The problem with that Proposition is that we cannot remove the features involved in restrictions (requirement or exclusion) associated with the feature in which we are interested. So, we need to add to the non-hidden features those that appear in a restriction associated with the original one.

Example 3. Let us assume that we want to compute the probability of A in the term

$$\mathtt{B} \not\Rightarrow \mathtt{C}\ \mathtt{in}\ \mathtt{C} \Rightarrow \mathtt{A}\ \mathtt{in}\ P$$

Where P is a term without restrictions. Then we compute the probability of A in the term

$$\mathtt{B} \not\Rightarrow \mathtt{C}\ \mathtt{in}\ \mathtt{C} \Rightarrow \mathtt{A}\ \mathtt{in}\ (Q[\{\mathtt{A}, \mathtt{B}, \mathtt{C}\}])$$

This section presents the results obtained from an experimental study to show the applicability and scalability of our approach. In order to carry out this study, we have implemented a set of scripts to demonstrate the applicability of the probabilistic extension - of the denotational semantics - presented in this paper. The source code of the scripts used in this section is available at the main project site [3]. In essence, we perform two experiments. The former focuses on measuring the performance of our proposed implementation for processing a feature model. This means, given a feature model, calculating the time to compute the probability of having each feature in the valid products set. The second experiment consists on analyzing the scalability of our proposed implementation. The idea is to study if there is a correlation between the number of features of each type and the processing time. The experiments have been executed in a computer with the following features: Intel(R) Xeon(R) Quad-Core CPU E5-2670 @ 2.60GHz, 64 GB of RAM memory and Centos 7 Operating System.

The study described in this section seeks to answer the following questions:

- **RQ1**: Is it possible to translate current feature models to support probabilistic information?
- **RQ2**: Is it possible to describe a formal framework that translates the current feature models to probabilistic methods?
- **RQ3**: What is the impact of applying probabilistic analysis methods to current feature models like FODA?

[3] `http://ccamacho.github.io/phd/resources/03_splap.tar`

6.1 Model analysis

Firstly, we have carried out an experiment to show the computing time required to calculate the probability of having each feature in the set of valid products. In order to run this experiment, a variability model consisting of 1500 features has been used. This variability model has been generated using BeTTy [18]. Figure 6 depicts the parameters used in the feature models generator.

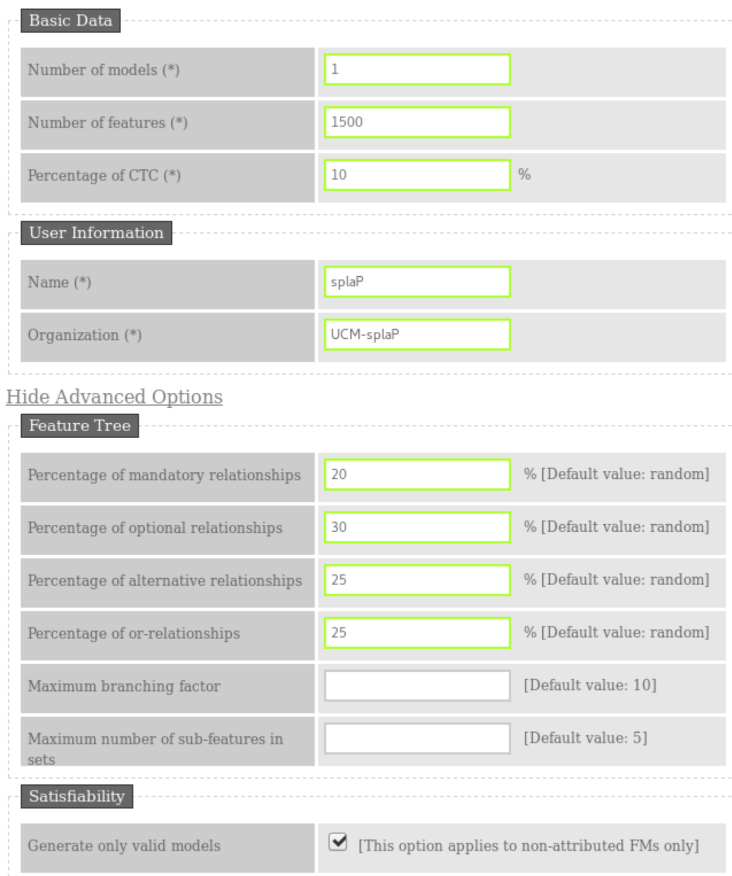

Fig. 6. BeTTy parameters.

BeTTy generates feature models based on a set of pre-defined parameters. The meaning of these parameters focuses on how BeTTy randomly generates these models. In this case BeTTy requires 4 parameters, where the sum of the probabilities for these parameters must be 1, that is:

– The probability of having a mandatory feature.

- The probability of having an optional feature.
- The probability of having a feature in a *choose-one* relationship.
- The probability of having a feature in a *conjunction* relationship.

The values used for these parameters to generate the feature model are the following:

- The probability of having a mandatory feature is 0.2.
- The probability of having an optional feature is 0.3.
- The probability of having a feature in a *choose-one* relationship is 0.25.
- The probability of having a feature in a *conjunction* relationship is 0.25.

The idea of using this configuration is to have the same probability for the different relationships in the model, that is, we use a probability of 0.25 for both the *choose-one* and *conjunction* relationships. It is known that the optional features induce a combinatorial explosion for creating all the possible valid products from a feature model. Since we are interested on investigating the performance of our approach, we use a probability of 0.3 for having optional features in the model and a probability of 0.2 for having mandatory features. The sum of all probabilities must be 1. If no weight is configured, all features and relationships have a random weight, it being not possible to correlate the obtained results with our model analysis. Additionally, the percentage of cross-tree constraints is set to 10%, which is not related to the sum of the probabilities of the previous parameters.

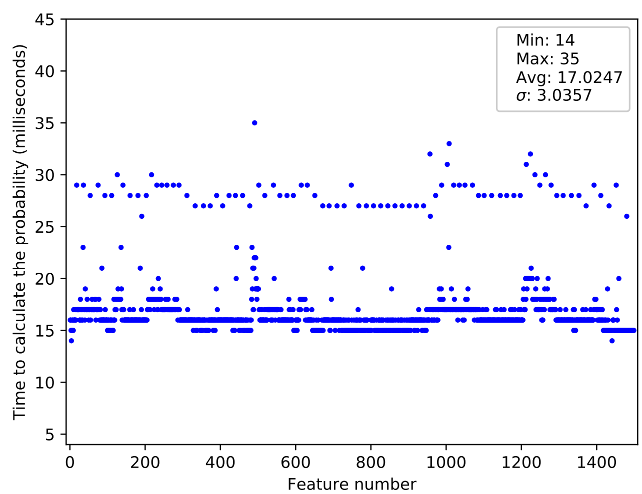

Fig. 7. Computing time analysis for a model consisting of 1500 features.

Figure 7 shows the obtained results from this experiment, where the x-axis depicts the ID of each generated feature and the y-axis represents the time required to calculate the probability of having the feature in a final product. It is important to remark that there is a small variation in the processing time for calculating the probability of each single feature. We think that this variation is mainly caused by both the stochastic nature of the generated model and the inherent noise of the node where the experiment is launched (e.g. disk latencies, operating system overhead, memory paging, etc.) and, therefore, it is not related to the algorithm itself.

Since the processing time for each feature is relatively low, it being around milliseconds, a single delay in the process scheduling may have a direct impact in the overall algorithm performance. Hence, this overhead can be considered insignificant since, in general, the time for processing each feature in the model ranges from 15 to 28 milliseconds.

The model used in the experiments has been generated and processed 10 times, providing similar results (see Table 1). That is, the major part of the features require between 14 and 20 milliseconds to be processed, while there is a small portion of them requiring a processing time between 20 and 36 milliseconds. However, there are unusual situations where a feature require 60 milliseconds to be processed.

Execution	**Minimum**	**Maximum**	**Average**	**Standard deviation**
1	14	35	17.0247	3.0357
2	14	41	17.2325	3.3911
3	14	36	16.3983	3.0823
4	14	34	16.5417	3.1394
5	14	34	16.7105	3.1343
6	14	43	16.7865	3.3871
7	14	47	17.3589	3.9768
8	14	63	17.6684	4.2557
9	15	51	18.5851	4.3072
10	14	55	18.0947	4.4526

Table 1. Computing time analysis table.

Figure 8 shows the probability of each feature - in the analyzed model - to be part of a final product, where the x-axis represents the feature ID and the y-axis represents the probability. Figure 9 represents a histogram of the calculated probabilities for a better readability of the results. This chart clearly shows that there exist different groups of features having a similar probability. In this case, the probability of the major part of the features ranges between 0.7 and 0.8. Thus, there are 450 features with, at least, a probability equal to 0.75 of being in a final product. As a conclusion, this analysis might allow us to establish that by testing only the 30% of the software product line components (450 features) we can ensure that, at least, the 75% of the generated products, are tested.

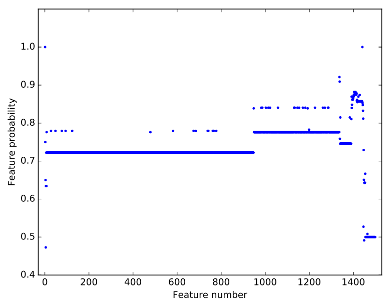

Fig. 8. Probabilistic analysis for a 1500 features model.

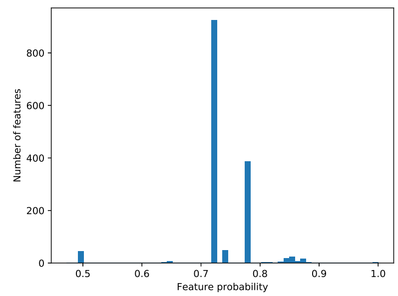

Fig. 9. Probabilistic histogram for processing a 1500 features model.

It is important to differentiate the probabilities defined in BeTTy, which are used to generate a feature model, and the probability - calculated from the model - to have a feature in a final product.

For instance, if we configure BeTTy to generate a feature model using a probability of 0.2 for having a mandatory feature, that means that 20% of the generated features are mandatory. However, that does not imply that these features be part of the 20% of the generated products, because the probability of having a feature in a final product depends on where this feature is placed in the model. If a given mandatory feature is placed in a choose-one relationship, it is possible that the other branch is used to generate the final product, discarding the mandatory feature. Hence, we can not assume that these 20% of the features will have a probability of 1 for being installed in the products.

6.2 Performance analysis

Secondly, an evaluation to analyze the scalability of our approach have been carried out. We are interested in investigating both the execution time and the amount of memory required for processing a model when the number of features increases. Hence, we use different configurations for creating a wide spectrum of variability models, which are randomly generated, using a different number of features that ranges from 1.000 to 10.000 (in increments of one thousand per experiment).

Specifically for each case, that is, given a configuration and a number of features, a model is randomly generated 30 times. Additionally, for each model, 100 features are randomly selected and, for each one, both the processing time and memory required to calculate its probability are analyzed.

Table 2 shows the configurations used to generate the variability models for this part of the empirical study, where each configuration represents the set of probabilities chosen for each operator across the three experiments, that is, *Mandatory* represents the probability of having a mandatory feature, *Optional* represents the probability of having an optional feature, *Choose-one* represents the probability of having a feature in a *choose-one* relation and *Conjunction* represents the probability of having a feature in a *conjunction* relation.

Configuration	Mandatory	Optional	Choose-one	Conjunction
1	0.69	0.15	0.15	0.01
2	0.5	0.15	0.15	0.2
3	0.2	0.15	0.15	0.5

Table 2. Configuration of the scalability experiments.

In this experiment, we have set the same values for the probabilities of the *Optional* and *Choose-one* features. Hence, these will remain the same across all the experiments and, thus, they should not interfere in the obtained results. We start with a low probability of having a *Conjunction* relationship in the model. In this case, for the first experiment, we use a probability of 0.01, which is increased in the next configurations to 0.2 and 0.5, respectively. This idea is to show the impact of the *Conjunction* relationship in the time and memory required for processing the models.

For each configuration, we have generated 30 models per number of features, that is, we generate 30 different models containing 1000 features, 30 different models containing 2000 features, and so on until 10.000 features.

Figure 10 and figure 11 show the execution time and the required amount of memory, respectively, for processing the variability models generated using *Configuration 1*. In these models, only 1% of the features have a conjunction relation. In general, the processing time when the number of features increases is linear. Only in few cases, where the number of features ranges from 5000 to 8000, the results provide anomalous values. This is mainly caused by the random nature of the generated models (30 for each case). On the contrary, the memory usage depicts that there are several groups where the memory usage remains constant, one group of models containing between 3,000 and 5,000 features and other group of models containing between 7,000 and 10,000 features. In summary, our implementation shows good scalability results for processing the models generated using *Configuration 1*: it requires, in the worst case scenario, 215 ms and 0.32 GB of RAM to process the model.

Figure 12 and figure 13 show the results for analyzing the models generated using *Configuration 2*. It is important to remark that 20% of the features in the generated models have a conjunction relation. In this case, both the execution time and memory usage for processing a model when the number of features increases are exponential. These charts clearly show a turning point when the model reaches 6,000 features and, therefore, the required processing time and memory are significantly lower for those models that do not reach 6,000 features. However, the requirements to process the model in the worst case scenario, that is, using a model containing 10,000 features, are 300 sec. and 3.84 GB of RAM memory, which are acceptable.

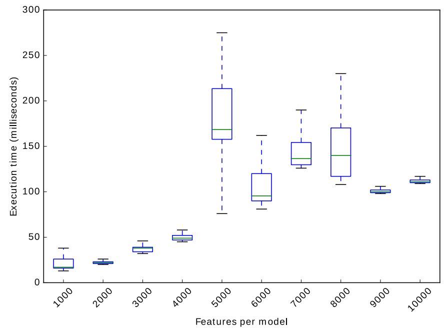

Fig. 10. Execution time for processing models generated using *Configuration 1*.

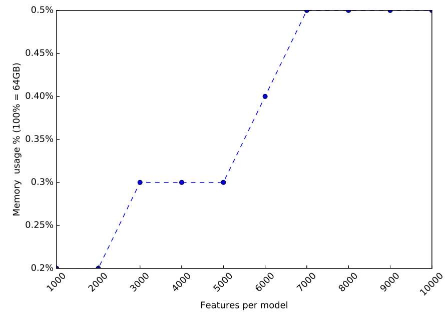

Fig. 11. Memory usage for processing models generated using *Configuration 1*.

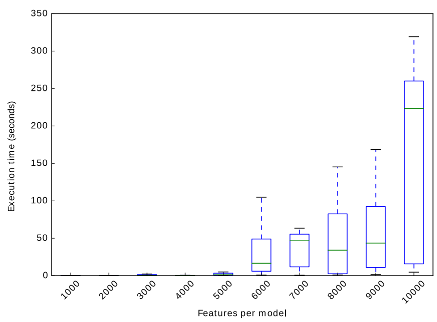

Fig. 12. Execution time for processing models generated using *Configuration 2*.

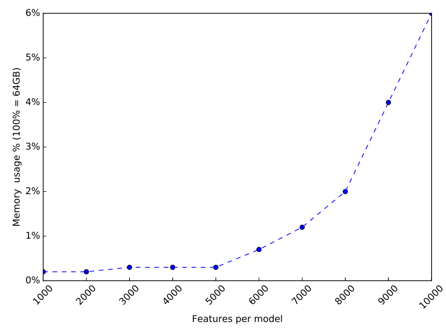

Fig. 13. Memory usage for processing models generated using *Configuration 2*.

Figure 14 and figure 15 show the results for processing the models generated using *Configuration 3*. In this case, half of the features in the model have a conjunction relation. Similarly to the previous experiment, these charts show that both the execution time and the memory usage for processing a model when the number of features increases are exponential. In the obtained results we can observe the same turning point detected in the previous models generated using *Configuration 2*, that is, when the model reaches 6,000 features. Model processing requirements, that is, execution time and memory usage, grow much faster for these models than for those based on previous configurations. Also, it is important to notice that the models containing 9,000 and 10,000 features cannot be processed due to memory limitations.

6.3 Discussion of the results

In this section we discuss the obtained results from the empirical study and provide a comparison with the ones obtained using `SPLA` [8] and the cost extension $\mathtt{SPLA}^{\mathcal{C}}$ [17]. Specifically, we are interested in comparing the performance

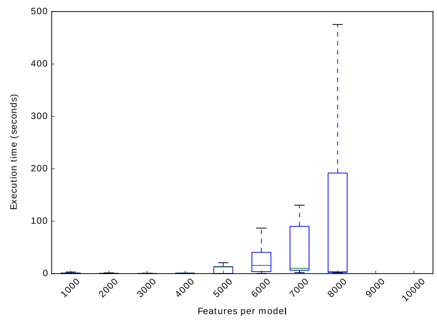

Fig. 14. Execution time for processing models generated using *configuration 3*.

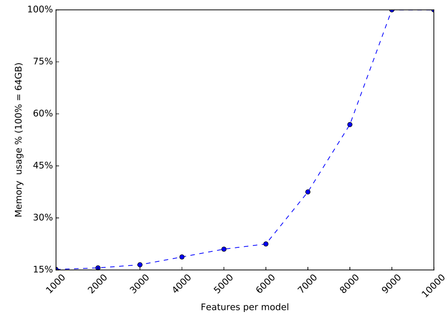

Fig. 15. Memory usage for processing models generated using *configuration 3*.

between the implementation of the probabilistic extension and the implementation of the denotational semantic from `SPLA` [8]. Also, we provide the answers for the research questions.

The experiments carried out in Section 6.2 uses feature models containing a maximum of 10000 features. In general, these results show that increasing the number of features having a conjunction relation has a direct impact on the overall performance. In fact, increasing the number of features having a conjunction relation generates a combinatorial explosion that hampers the processing of models. First, the execution time to completely process a model significantly grows. Second, large amounts of memory are required to store those combinations. In some cases, using large models with a high percentage of features having a conjunction relation may cause a bottleneck in the memory system. In fact, models generated using *Configuration 3* with 9,000 and 10,000 features cannot be processed using 64 GB of RAM. In this case, the worst case scenario, which generates a model where the 50% of the features are placed in a conjunction relationship, requires approximately 500000 ms.

Figure 16 shows the results of an experiment using models containing different number of features, which ranges from 50 to 300. In this case, we use the implementation of the denotational semantic from `SPLA` [8] to process the models, which do not use probabilistic information. If we use only the executions that successfully process the model, the worst case scenario requires 786126 ms to process a model containing 150 features.

Also, based on the results published in $\texttt{SPLA}^{\mathcal{C}}$ [17], the models used for the simulations were processed in an 8-node cluster. However, these models only contains 17 features. Figure 17 shows the algorithm behavior which best time its approximately 300 seconds.

For those specific implementations and simulations we can conclude that the denotational semantics of the probabilistic extension implementation improves dramatically the performance in comparison with the denotational semantics implementations presented in previous works [8, 17].

Following, we provide the answers to the research questions.

Features	Time (ms.)	Products
50	6	48
60	25	108
70	15	104
80	8	31
90	38	0
100	55	1404
110	13	12
120	2139	1
130	511	24802
140	7	6
150	786126	1312848
160	136	5670
170	42	398

Features	Time (ms.)	Products
180	744	6384
190	1390	7232
200	960000	-
210	97770	800544
220	263	51
230	47	8
240	65	29
250	191	5920
260	205	7296
270	250	4301
280	960000	-
290	65	3
300	960000	-

Fig. 16. Denotational benchmark from [8].

RQ1: Is it possible to translate current feature models to support probabilistic information?

In order to ask this question we have implemented the denotational semantic of the probabilistic extension. Since our framework is based on `FODA`, we can state that the answer is yes, it is possible to translate current feature models, like `FODA`, to represent and support probabilistic information.

RQ2: Is it possible to describe a formal framework that translates the current feature models to probabilistic methods?

General use models have been proposed to model variability in software product lines [27, 28] and, specifically, for feature-oriented systems [25, 37]. Thus, all previous work focuses on generic representations. However, this work is based on including probabilistic information to the well-known feature model `FODA`. Based on our previous results [8, 17], together with the results presented in this work, we can state that state it is possible to describe a formal framework that translates the current feature models to probabilistic methods.

RQ3: What is the impact of applying probabilistic analysis methods to current feature models like `FODA`?

In order to ask this question we carried out some experiments using our implementation of the denotational semantic `SPLA` [8] and the cost extension $\texttt{SPLA}^{\mathcal{C}}$ [17]. The results obtained have been compared with the implementation of the probabilistic extension. We observed that the latter is able to process larger models faster, which provides a greater scalability when the size of the model to be processed grows. Since the probabilistic extension focuses on hiding those features that do not affect the processing of the probability of given feature for being part of a valid product, the required tome for processing the model is considerably reduced.

	1 Worker	2 Workers	4 Workers	8 Workers	16 Workers	32 Workers
1 Node	2010.44763303	1060.80047798	586.014445066	543.712262154	521.616870165	521.292215109
2 Nodes	2059.06947899	1046.87119007	575.115453959	296.285589933	366.384442091	341.007520199
4 Nodes	2031.25184584	1093.52087283	586.344302893	320.010899067	300.745616913	382.748430014
8 Nodes	2207.35427213	1143.951792	576.370896101	495.507214785	308.374300957	287.340030909

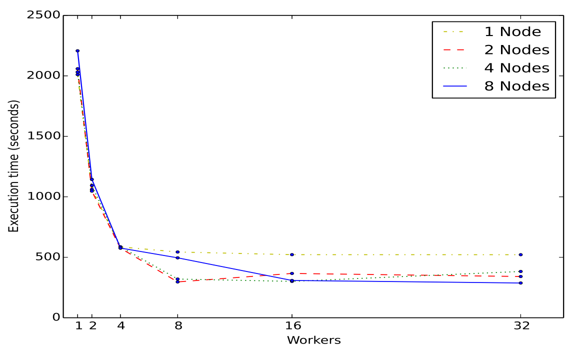

Fig. 17. Cluster execution times for 1, 2, 4 and 8 nodes; and 1, 2, 4, 8, 16 and 32 workers from [17].

7 Conclusions

We have presented a probabilistic extension of our formal framework to specify and analyze SPLs. The main goal of this proposal is to alleviate the combinatorial explosion issue, where a vast number of combinations are generated by some of the algebra operators, that making unpractical to process the entire SPL. By including probabilistic information in our process algebra, we are able to generate significant information for determining the probability of a given feature to be present in a valid product. We have provided two semantic frameworks for our language and have proved that they identify the same processes. In order to show the applicability of our approach, a tool containing the implementation of the denotational semantics for our probabilistic extension has been developed. This tool has been used to conduct an experimental study. The results of this study show that, using our approach, it is possible to compute the probability of each feature in the SPL to be present in a valid product. Thus, the testing process can focus on those features having a high probability of being included in a product.

We have two main lines for future work. First, it is important to develop mechanisms allowing us to simplify and/or optimize terms based on the results of the probabilistic analysis. In addition, we plan to find practical use cases to show the usefulness of having a probabilistic extension for SPLs.

Also it is interesting the future integration's of our formal framework to existing tooling frameworks like ProFeat [37] and probabilistic model checkers like PRISM [38].

References

1. J. McGregor, "Testing a software product line," Tech. Rep. CMU/SEI-2001-TR-022, Carnegie Mellon University, Software Engineering Institute, 2001.
2. K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) feasibility study," Tech. Rep. CMU/SEI-90-TR-21, Carnegie Mellon University, 1990.
3. M. Griss and J. Favaro, "Integrating feature modeling with the RSEB," in *5th International Conference on Software Reuse, ICSR'98*, pp. 76–85, 1998.
4. K. Kollu, *Evaluating The PLUSS Domain Modeling Approach by Modeling the Arcade Game Maker Product Line*. PhD thesis, Umea University, 2005.
5. M. Eriksson, J. Borstler, and K. Borg, "The pluss approach - domain modeling with features, use cases and use case realizations," in *9th International Conference on Software Product Lines, SPLC'06*, pp. 33–44, Springer-Verlag, 2006.
6. S. Nakajima, "Semi-automated diagnosis of foda feature diagram," in *25th ACM Symposium on Applied Computing, SAC'10*, pp. 2191–2197, ACM Press, 2010.
7. Y. Bontemps, P. Heymans, P. Schobbens, and J. Trigaux, "Semantics of FODA feature diagrams," in *1st Workshop on Software Variability Management for Product Derivation – Towards Tool Support, SPLCW'04*, pp. 48–58, Springer, 2004.
8. C. Andrés, C. Camacho, and L. Llana, "A formal framework for software product lines," *Information & Software Technology*, vol. 55, no. 11, pp. 1925–1947, 2013.
9. J. Sun, H. Zhang, and H. Wang, "Formal semantics and verification for feature modeling," in *10th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS'05*, pp. 303–312, IEEE Computer Society Press, 2005.
10. P. Höfner, R. Khédri, and B. Möller, "Feature algebra," in *14th International Symposium on Formal Methods, FM'06*, vol. 4085 of *LNCS*, pp. 300–315, Springer, 2006.
11. P. Höfner, R. Khédri, and B. Möller, "An algebra of product families," *Software and System Modeling*, vol. 10, no. 2, pp. 161–182, 2011.
12. M. Mannion, "Using first-order logic for product line model validation," in *2nd International Software Product Line Conference, SPLC'02*, pp. 176–187, Springer, 2002.
13. K. Czarnecki and A. Wasowski, "Feature diagrams and logics: There and back again," in *11th International Software Product Line Conference, SPLC'07*, pp. 23–34, IEEE Computer Society Press, 2007.
14. P. Asirelli, M. H. ter Beek, S. Gnesi, and A. Fantechi, "A deontic logical framework for modelling product families," in *4th International Workshop on Variability Modelling of Software-Intensive Systems, VaMoS'10*, pp. 37–44, 2010.
15. P. Asirelli, M. H. ter Beek, A. Fantechi, and S. Gnesi, "A logical framework to deal with variability," in *8th Int. Conf. on Integrated Formal Methods, IFM'10*, pp. 43–58, Springer, 2010.
16. J. Nummenmaa, T. Nummenmaa, and Z. Zhang, "On the use of ltss to analyze software product line products composed of features," in *Knowledge Engineering and Management* (F. Sun, T. Li, and H. Li, eds.), vol. 214 of *Advances in Intelligent Systems and Computing*, pp. 531–541, Springer Berlin Heidelberg, 2014.
17. C. Camacho, L. Llana, and A. Núñez, "Cost-related interface for software product lines," *Journal of Logical and Algebraic Methods in Programming*, vol. 85, pp. 227–244, 2016.
18. S. Segura, R. Hierons, D. Benavides, and A. Ruiz, "Automated metamorphic testing on the analyses of feature models," *Information & Software Technology*, vol. 53, no. 3, pp. 245–258, 2011.

19. R. M. Hierons and M. G. Merayo, “Mutation testing from probabilistic and stochastic finite state machines,” *Journal of Systems and Software*, vol. 82, no. 11, pp. 1804–1818, 2009.
20. M. E. Andrés, C. Palamidessi, P. v. Rossum, and A. Sokolova, “Information hiding in probabilistic concurrent systems,” *Theoretical Computer Science*, vol. 412, no. 28, pp. 3072–3089, 2011.
21. A. Sokolova, “Probabilistic systems coalgebraically: A survey,” *Theoretical Computer Science*, vol. 412, no. 38, pp. 5095–5110, 2011.
22. R. M. Hierons and M. Núñez, “Using schedulers to test probabilistic distributed systems,” *Formal Aspects of Computing*, vol. 24, no. 4-6, pp. 679–699, 2012.
23. Y. Deng, R. v. Glabbeek, M. Hennessy, and C. Morgan, “Real-reward testing for probabilistic processes,” *Theoretical Computer Science*, vol. 538, pp. 16–36, 2014.
24. P. R. D’Argenio, D. Gebler, and M. D. Lee, “A general SOS theory for the specification of probabilistic transition systems,” *Information and Computation*, vol. 249, pp. 76–109, 2016.
25. C. Dubslaff, C. Baier, and S. Klüppelholz, *Probabilistic Model Checking for Feature-Oriented Systems*, pp. 180–220. Springer Berlin Heidelberg, 2015.
26. M. Cordy, P. Heymans, P. Schobbens, A. M. Sharifloo, C. Ghezzi, and A. Legay, “Verification for reliable product lines,” *CoRR*, vol. abs/1311.1343, 2013.
27. M. H. ter Beek, A. Legay, A. Lluch-Lafuente, and A. Vandin, “Quantitative analysis of probabilistic models of software product lines with statistical model checking,” in Proceedings 6th Workshop on *Formal Methods and Analysis in SPL Engineering,* London, UK, 11 April 2015 (J. M. Atlee and S. Gnesi, eds.), vol. 182 of *Electronic Proceedings in Theoretical Computer Science*, pp. 56–70, Open Publishing Association, 2015.
28. M. H. ter Beek, A. Legay, A. Lluch-Lafuente, and A. Vandin, “Statistical analysis of probabilistic models of software product lines with quantitative constraints,” in *Proceedings of the 19th International Conference on Software Product Line*, SPLC ’15, pp. 11–15, ACM, 2015.
29. X. Devroey, G. Perrouin, M. Cordy, H. Samih, A. Legay, P.-Y. Schobbens, and P. Heymans, “Statistical prioritization for software product line testing: an experience report,” *Software & Systems Modeling*, vol. 16, no. 1, pp. 153–171, 2017.
30. L. de Moura and N. Bjorner, “Z3: An efficient smt solver,” in *Tools and Algorithms for the Construction and Analysis of Systems* (C. R. Ramakrishnan and J. Rehof, eds.), (Berlin, Heidelberg), pp. 337–340, Springer Berlin Heidelberg, 2008.
31. S. Sebastio and A. Vandin, “Multivesta: Statistical model checking for discrete event simulators,” in *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, ValueTools ’13, pp. 310–315, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013.
32. M. Varshosaz and R. Khosravi, “Discrete time markov chain families: Modeling and verification of probabilistic software product lines,” pp. 34–41, 08 2013.
33. M. F. Johansen, O. Haugen, and F. Fleurey, “Properties of realistic feature models make combinatorial testing of product lines feasible,” in *Proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems*, MODELS’11, pp. 638–652, 2011.
34. M. F. Johansen, O. Haugen, F. Fleurey, A. G. Eldegard, and T. Syversen, “Generating better partial covering arrays by modeling weights on sub-product lines,” in *Proceedings of the 15th International Conference on Model Driven Engineering Languages and Systems*, MODELS’12, pp. 269–284, 2012.

35. M. B. Cohen, M. B. Dwyer, and J. Shi, "Coverage and adequacy in software product line testing," in *Workshop on Role of software architecture for testing and analysis, ROSATEA'06*, pp. 53–63, ACM Press, 2006.
36. D. Fischbein, S. Uchitel, and V. Braberman, "A foundation for behavioural conformance in software product line architectures," in *Workshop on Role of software architecture for testing and analysis, ROSATEA'06*, pp. 39–48, ACM Press, 2006.
37. P. Chrszon, C. Dubslaff, S. Klüppelholz, and C. Baier, "Profeat: feature-oriented engineering for family-based probabilistic model checking," *Formal Aspects of Computing*, vol. 30, no. 1, pp. 45–75, 2018.
38. M. Kwiatkowsa, G. Norman, and D. Parker, "The prism benchmark suite," in *2012 Ninth International Conference on Quantitative Evaluation of Systems*, pp. 203–204, Sept 2012.
39. P. D'Argenio, H. Hermanns, and J.-P. Katoen, "On generative parallel composition," in *Workshop on Probabilistic Methods in Verification, PROBMIV'98, ENTCS 22*, pp. 30–54, Elsevier, 1999.
40. J. Hillston, *A Compositional Approach to Performance Modelling*. New York, NY, USA: Cambridge University Press, 1996.

A Results for the proof of Proposition 3

Lemma 3. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$ and $\mathtt{A} \in \mathcal{F}$, then $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(\mathtt{A}; P)$ if and only if

$$p = \sum \{\!| r \mid (pr', r) \in \mathtt{prod}^{\mathcal{P}}(P) \ \wedge \ pr' \cup \{\mathtt{A}\} = pr |\!\}$$

Proof. The other transition of $\mathtt{A}; P$ is $\mathtt{A}; P \xrightarrow{\mathtt{A}}_1 Q$. Then $\mathtt{A}; P \overset{s}{\Longrightarrow}_p P$ if and only if

$$\mathtt{A}; P \xrightarrow{\mathtt{A}}_1 P \overset{s}{\Longrightarrow}_p Q \qquad \wedge \qquad s = \mathtt{A} \cdot s'$$

then

$$\begin{aligned} p = \sum \{\!| r \mid \mathtt{A}; P \overset{s\checkmark}{\Longrightarrow}_p \mathtt{nil} \ \wedge \ \lfloor s \rfloor = pr \ |\!\} = \\ \sum \{\!| r \mid \mathtt{A}; P \xrightarrow{\mathtt{A}}_1 P \overset{s'\checkmark}{\Longrightarrow}_r \mathtt{nil} \ \wedge \ \lfloor \mathtt{A} \cdot s' \rfloor = pr |\!\} \\ \sum \{\!| r \mid P \overset{s'\checkmark}{\Longrightarrow}_r \mathtt{nil} \ \wedge \ \{\mathtt{A}\} \cup \lfloor s' \rfloor = pr |\!\} \\ \sum \{\!| r \mid (pr', r) \in \mathtt{prod}^{\mathcal{P}}(P) \ \wedge \ \{\mathtt{A}\} \cup pr' = pr |\!\} \end{aligned}$$

□

Lemma 4. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$, $\mathtt{A} \in \mathcal{F}$ and $q \in (0, 1)$, then $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(\overline{\mathtt{A}};_q P)$ if and only if $(pr, p) = (\varnothing, 1 - q)$ or

$$p = q \cdot \sum \{r \mid (pr', r) \in \mathtt{prod}^{\mathcal{P}}(P) \ \wedge \ pr' \cup \{\mathtt{A}\} = pr\}$$

Proof. There exist two transitions to $\overline{\mathtt{A}};_q P$: $\overline{\mathtt{A}};_q P \xrightarrow{\mathtt{A}}_q P$ and $\overline{\mathtt{A}};_q P \xrightarrow{\checkmark}_{1-q} \mathtt{nil}$. So forth if $\overline{\mathtt{A}};_q P \overset{s}{\Longrightarrow}_r Q$ then

- $s = \checkmark$ and $r = 1 - q$, or
- $s = \mathtt{A} \cdot s'$, $P \overset{s}{\Longrightarrow}_{r'} Q$, and $r = q \cdot r'$.

So, if $pr = \lfloor \mathtt{A} \cdot s' \rfloor$ then $pr \neq \varnothing$. So then $(\varnothing, 1 - q) \in \mathtt{prod}^{\mathcal{P}}(\overline{\mathtt{A}};_q P)$. Now suppose $pr \neq \varnothing$, then $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(\overline{\mathtt{A}};_q P)$ if and only if

$$\begin{aligned} d = \sum \{\!| r \mid \overline{\mathtt{A}};_q P \overset{s\checkmark}{\Longrightarrow} \mathtt{nil} \ \wedge \ \lfloor s \rfloor = pr |\!\} = \\ \sum \{\!| r \mid \overline{\mathtt{A}};_q P \xrightarrow{\mathtt{A}}_q P \overset{s'\checkmark}{\Longrightarrow}_{r'} \mathtt{nil} \ \wedge \ \lfloor \mathtt{A} \cdot s' \rfloor = pr \ \wedge \ r = q \cdot r' |\!\} = \\ \sum \{\!| r \mid P \overset{s'\checkmark}{\Longrightarrow}_{r'} \mathtt{nil} \ \wedge \ \{\mathtt{A}\} \cup \lfloor s' \rfloor = pr \ \wedge \ r = q \cdot r' |\!\} = \\ \sum \{\!| r \mid (pr', r') \in \mathtt{prod}^{\mathcal{P}}(P) \ \wedge \ \{\mathtt{A}\} \cup pr' = pr \ \wedge \ r = q \cdot r' |\!\} = \\ q \cdot \sum \{\!| r' \mid (pr', r') \in \mathtt{prod}^{\mathcal{P}}(P) \ \wedge \ \{\mathtt{A}\} \cup pr' = pr |\!\} \end{aligned}$$

□

Lemma 5. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$ and $q \in (0, 1)$, then $P \vee_q Q \overset{s}{\Longrightarrow}_r R$ if and only if

- $P \xRightarrow{s}_{r'} R$ y $r = q \cdot r'$, o
- $Q \xRightarrow{s}_{r'} R$ y $r = (1-q) \cdot r'$

Proof. This lemma is a consequence of rules [**cho1**] and [**cho2**] from the operational semantics.

$\square$

Lemma 6. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$ and $q \in (0,1)$, then $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(P \vee_q Q)$ if and only if

$$p = \left(q \cdot \sum\{r \mid (pr, r) \in \mathtt{prod}^{\mathcal{P}}(P)\}\right) + \left((1-q) \cdot \sum\{r \mid (pr, r) \in \mathtt{prod}^{\mathcal{P}}(Q)\}\right)$$

Proof. $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(P \vee_q Q)$ if and only if

$$\begin{aligned}
p = \sum \wr r \mid P \vee_q Q \xRightarrow{s\checkmark}_{r} \mathtt{nil} \int = \\
\sum \wr r \mid (P \xRightarrow{s\checkmark}_{r'} \mathtt{nil} \ \wedge \ r = q \cdot r') \ \vee \ (Q \xRightarrow{s\checkmark}_{r'} \mathtt{nil} \ \wedge \ r = (1-q) \cdot r') \int = \\
\sum \wr r \mid P \xRightarrow{s\checkmark}_{r'} \mathtt{nil} \ \wedge \ r = q \cdot r' \int + \sum \wr r \mid Q \xRightarrow{s\checkmark}_{r'} \mathtt{nil} \ \wedge \ r = (1-q) \cdot r' \int = \\
q \cdot \sum \wr r \mid P \xRightarrow{s\checkmark}_{r} \mathtt{nil} \int + (1-q) \cdot \sum \wr r \mid Q \xRightarrow{s\checkmark}_{r} \mathtt{nil} \int = \\
q \cdot \sum \wr r \mid (pr, r) \in \mathtt{prod}^{\mathcal{P}}(P) \int + (1-q) \cdot \sum \wr r \mid (pr, r) \in \mathtt{prod}^{\mathcal{P}}(Q) \int
\end{aligned}$$

$\square$

Definition 10. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$. We define the height of the syntactic tree of P, written $\mathsf{h}(P)$ as follows:

$$\begin{array}{lcl}
\mathsf{h}(\mathtt{nil}) & = & 0 \\
\mathsf{h}(\checkmark) & = & 1 \\
\mathsf{h}(\mathtt{A}; P),\ \mathsf{h}(\overline{\mathtt{A}};_p P), & & \\
\mathsf{h}(\mathtt{A} \not\Rightarrow \mathtt{B} \ \mathtt{in} \ P),\ \mathsf{h}(P \backslash \mathtt{A}), & = & 1 + \mathsf{h}(P) \\
\mathsf{h}(\mathtt{A} \Rightarrow \mathtt{B} \ \mathtt{in} \ P),\ \mathsf{h}(P \Rightarrow \mathtt{A}) & & \\
\mathsf{h}(P \vee_p Q),\ \mathsf{h}(P \wedge Q) & = & 1 = \max(\mathsf{h}(P), \mathsf{h}(Q))
\end{array}$$

$\square$

Lemma 7. Let $P, P' \in \mathtt{SPLA}^{\mathcal{P}}$, $\mathtt{A} \in \mathcal{F}$, and $p \in (0,1]$. If $P \xrightarrow{\mathtt{A}}_p P'$ then $\mathsf{h}(P') < \mathsf{h}(P)$.

Proof. The proof is done easily by structural induction.

$\square$

Lemma 8. Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$, $pr \subseteq \mathcal{F}$ be a product, and $p \in (0,1)$, then $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(P \wedge Q)$ iff

$$p = \sum \wr r \mid \exists (pr_1, p_1) \in \mathtt{prod}^{\mathcal{P}}(Q), (pr_2, p_2) \in \mathtt{prod}^{\mathcal{P}}(Q) : \ pr = pr_1 \cup pr_2, r = p_1 \cdot p_2 \int$$

Proof. The proof is made by induction on $\mathsf{h}(P)+\mathsf{h}(Q)$. First let us consider the base case $\mathsf{h}(P)+\mathsf{h}(Q)=0$, that is $P,Q\in\{\mathtt{nil},\checkmark\}$. If $P=\mathtt{nil}$ (respectively $Q=\mathtt{nil}$) then $P\wedge Q$ has no products. If $P=Q\checkmark$ then

$$\mathtt{prod}^{\mathcal{P}}(P)=\mathtt{prod}^{\mathcal{P}}(Q)=\mathtt{prod}^{\mathcal{P}}(P\wedge Q)=\{(0,1)\}$$

from with the result are immediate from the definitions.

So let assume the inductive case where $|pr|\geq 1$. In this case we obtain $(pr,p)\in\mathtt{prod}^{\mathcal{P}}(P\wedge Q)$ (by definition) iff

$$p=\sum \wr r \mid P\wedge Q \overset{s\checkmark}{\Longrightarrow}_r \mathtt{nil},\ pr=\lfloor s\rfloor \int \tag{1}$$

If $pr=\varnothing$, the only possible transition for $P\wedge Q$ is the one derived from [**con3**]. Then we obtain easily the result:

$$\begin{aligned}p=\sum \wr r \mid P\wedge Q \overset{\checkmark}{\Longrightarrow}_r \mathtt{nil} \int =\\ \sum \wr r_1\cdot r_2 \mid P \overset{\checkmark}{\Longrightarrow}_{r_1} \mathtt{nil},\ Q \overset{\checkmark}{\Longrightarrow}_{r_2} \mathtt{nil} \int =\\ \sum \wr r_1\cdot r_2 \mid (\varnothing,r_1)\in\mathtt{prod}^{\mathcal{P}}(P),\ (\varnothing,r_2)\in\mathtt{prod}^{\mathcal{P}}(Q) \int\end{aligned}$$

If $pr\neq\varnothing$, we can split the previous sum according the first transition of $P\wedge Q$ according to rules [**con1**], [**con2**], [**con4**], and [**con5**]. Since rules [**con1**] and [**con4**] are symmetric to [**con2**] and [**con5**], we only show the corresponding transitions to the first two rules:

$$\begin{aligned}(1)=\sum \wr \frac{1}{2}\cdot r_1\cdot r_2 \mid P \overset{\mathtt{A}}{\longrightarrow}_{r_1} P',\\ P'\wedge Q \overset{s'\checkmark}{\Longrightarrow}_{r_2} \mathtt{nil},\ pr=\{\mathtt{A}\}\cup\lfloor s'\rfloor \int +\\ \sum \wr \frac{1}{2}\cdot r_1\cdot r_2\cdot r_3 \mid P \overset{\mathtt{A}}{\longrightarrow}_{r_1} P',\ Q \overset{\checkmark}{\longrightarrow}_{r_2} \mathtt{nil},\\ P' \overset{s'\checkmark}{\Longrightarrow}_{r_3} \mathtt{nil},\ pr=\{\mathtt{A}\}\cup\lfloor s'\rfloor \int +\\ \textit{term corresponding rule } [\mathbf{con2}] + \textit{term corresponding rule } [\mathbf{con5}]\end{aligned} \tag{2}$$

Applying the definitions and grouping traces giving the same product, the previous term can be transformed as follows

$$\begin{aligned}(2)=\sum \wr \frac{1}{2}\cdot r_1\cdot r_2 \mid P \overset{\mathtt{A}}{\longrightarrow}_{r_1} P',\\ (pr',r_2)\in\mathtt{prod}^{\mathcal{P}}(P'\wedge Q),\ pr=\{\mathtt{A}\}\cup pr' \int +\\ \sum \wr \frac{1}{2}\cdot r_1\cdot r_2\cdot r_3 \mid P \overset{\mathtt{A}}{\longrightarrow}_{r_1} P',\ (pr',r_3)\in\mathtt{prod}^{\mathcal{P}}(P'),\\ (\varnothing,r_2)\in\mathtt{prod}^{\mathcal{P}}(Q),\ pr=\{\mathtt{A}\}\cup pr' \int +\\ \text{term corresponding rule } [\mathbf{con2}] + \text{term corresponding rule } [\mathbf{con5}]\end{aligned} \tag{3}$$

Now we can apply induction hypothesis to the first term of the previous sum (and the third that is symmetric).

$$\begin{aligned} (3) = \sum \wr \frac{1}{2} \cdot r_1 \cdot r_2 \cdot r_3 \mid P \xrightarrow{\mathtt{A}}_{r_1} P',\ (pr', r_2) \in \mathtt{prod}^{\mathcal{P}}(P'), \\ (pr'', r_3) \in \mathtt{prod}^{\mathcal{P}}(Q),\ pr = \{\mathtt{A}\} \cup pr' \cup pr'' \int + \\ \sum \wr \frac{1}{2} \cdot r_1 \cdot r_2 \cdot r_3 \mid P \xrightarrow{\mathtt{A}}_{r_1} P',\ (\varnothing, r_2) \in \mathtt{prod}^{\mathcal{P}}(Q), \\ (pr', r_3) \in \mathtt{prod}^{\mathcal{P}}(P'),\ pr = \{\mathtt{A}\} \cup pr' \int + \\ \text{term corresponding rule } [\textbf{con2}] + \text{term corresponding rule } [\textbf{con5}] \end{aligned} \tag{4}$$

Now let us consider the following set

$$\begin{aligned} \mathcal{Q} = \{(pr', r) \mid (pr', r) \in \mathtt{prod}^{\mathcal{P}}(Q),\ \exists P' \in \mathtt{SPLA}^{\mathcal{P}}, \mathtt{A} \in \mathcal{F}, pr' \subseteq \mathcal{F}, r_1, r_2 \in (0,1] : \\ P \xrightarrow{\mathtt{A}}_{r_1} P', (pr'',\ r_2) \in \mathtt{prod}(P'),\ pr = \{\mathtt{A}\} \cup pr' \cup pr''\} \end{aligned}$$

All pairs in $\mathcal{Q}$ appear in the first term of Equation (4). So we can apply the distributive property to reorganize that term obtaining

$$\begin{aligned} (4) = \frac{1}{2} \sum_{(pr', r) \in \mathcal{Q}} r \cdot \sum \wr r_1 \cdot r_2 \mid P \xrightarrow{\mathtt{A}}_{r_1} P',\ (pr', r_2) \in \mathtt{prod}^{\mathcal{P}}(P'), \\ pr = \{\mathtt{A}\} \cup pr' \cup pr'' \int + \\ \frac{1}{2} \cdot \sum \wr r_1 \cdot r_2 \cdot r_3 \mid P \xrightarrow{\mathtt{A}}_{r_1} P',\ (\varnothing, r_2) \in \mathtt{prod}^{\mathcal{P}}(Q), \\ (pr', r_3) \in \mathtt{prod}^{\mathcal{P}}(P'),\ pr = \{\mathtt{A}\} \cup pr' \int + \\ \text{term corresponding rule } [\textbf{con2}] + \text{term corresponding rule } [\textbf{con5}] \end{aligned} \tag{5}$$

If the empty product is not a product of Q the second term of the previous sum may disappear. Otherwise there exists $r \in (0,1]$ such that $(\varnothing, r) \in \mathtt{prod}^{\mathcal{P}}(Q)$. By definition, $(\varnothing, r) \in \mathcal{Q}$, so we remove the empty set from the first term and we obtain:

$$\begin{aligned} (5) = \frac{1}{2} \sum_{(pr', r) \in \mathcal{Q}, pr' \neq \varnothing} r \cdot \sum \wr r_1 \cdot r_2 \mid P \xrightarrow{\mathtt{A}}_{r_1} P', \\ (pr'', r_2) \in \mathtt{prod}^{\mathcal{P}}(P'), pr = \{\mathtt{A}\} \cup pr' \cup pr'' \int + \\ \frac{1}{2} \cdot \sum \wr r_1 \cdot r_2 \cdot r_3 \mid P \xrightarrow{\mathtt{A}}_{r_1} P',\ (\varnothing, r_2) \in \mathtt{prod}^{\mathcal{P}}(Q), \\ (pr', r_3) \in \mathtt{prod}^{\mathcal{P}}(P'),\ pr = \{\mathtt{A}\} \cup pr' \int + \\ \frac{1}{2} \cdot \sum \wr r_1 \cdot r_2 \cdot r_3 \mid P \xrightarrow{\mathtt{A}}_{r_1} P',\ (\varnothing, r_2) \in \mathtt{prod}^{\mathcal{P}}(Q), \\ (pr', r_3) \in \mathtt{prod}^{\mathcal{P}}(P'),\ pr = \{\mathtt{A}\} \cup pr' \int + \\ \text{term corresponding rule } [\textbf{con2}] + \text{term corresponding rule } [\textbf{con5}] \end{aligned} \tag{6}$$

Since the two last terms are identical can be added. Then, grouping the elements with the same product in the first, we obtain definition we obtain

$$\begin{aligned}(6) = \frac{1}{2} \sum_{(pr,r)\in\mathcal{Q},pr'\neq\varnothing} r \cdot \sum \wr r' \mid (pr', r') \in \mathtt{prod}^{\mathcal{P}}(P), pr' \neq \varnothing,\\ pr = pr \cup pr' \int + \\ \sum \wr r_1 \cdot r_2 \mid (pr, r_1) \in \mathtt{prod}(P),\ pr \neq \varnothing,\ (\varnothing, r_2) \in \mathtt{prod}^{\mathcal{P}}(Q) \int + \\ \text{term corresponding rule } [\mathbf{con2}] + \text{term corresponding rule } [\mathbf{con5}]\end{aligned} \tag{7}$$

Rewriting, taking into account the definition of $\mathcal{Q}$ the previous equation we obtain

$$\begin{aligned}(7) = \frac{1}{2} \sum \wr r_1 \cdot r_2 \mid (pr_1, r_1) \in \mathtt{prod}^{\mathcal{P}}(Q), pr_1 \neq \varnothing,\\ (pr_2, p_2) \in \mathtt{prod}^{\mathcal{P}}(P),\ pr_2 \neq \varnothing, pr = pr_1 \cup pr_2 \int + \\ \sum \wr r_1 \cdot r_2 \mid (pr, r_1) \in \mathtt{prod}^{\mathcal{P}}(P),\ pr \neq \varnothing,\ (\varnothing, r_2) \in \mathtt{prod}^{\mathcal{P}}(Q) \int + \\ \text{term corresponding rule } [\mathbf{con2}] + \text{term corresponding rule } [\mathbf{con5}]\end{aligned} \tag{8}$$

Then adding the symmetrical terms, and having into account that $pr \neq$, we obtain

$$\begin{aligned}(8) = \sum \wr r_1 \cdot r_2 \mid (pr_1, r_1) \in \mathtt{prod}^{\mathcal{P}}(Q), pr_1 \neq \varnothing,\\ (pr_2, p_2) \in \mathtt{prod}^{\mathcal{P}}(P), pr_2 \neq \varnothing, pr = pr_1 \cup pr_2 \int + \\ \sum \wr r_1 \cdot r_2 \mid (pr, r_1) \in \mathtt{prod}^{\mathcal{P}}(P),\ (\varnothing, r_2) \in \mathtt{prod}^{\mathcal{P}}(Q) \int + \\ \sum \wr r_1 \cdot r_2 \mid (pr, r_1) \in \mathtt{prod}^{\mathcal{P}}(Q),\ (\varnothing, r_2) \in \mathtt{prod}^{\mathcal{P}}(P) \int +\end{aligned} \tag{9}$$

Finally we can include the two last terms into the first one having into account that $pr \neq \varnothing$.

$$\begin{aligned}(9) = \sum \wr r_1 \cdot r_2 \mid (pr_1, r_1) \in \mathtt{prod}^{\mathcal{P}}(Q),\\ (pr_2, p_2) \in \mathtt{prod}^{\mathcal{P}}(P),\ pr = pr_1 \cup pr_2 \int\end{aligned} \tag{10}$$

□

Since the two last terms are identical can be add Since the two last terms are identical can be added and by definition we obtain ed and by definition we obtain

Lemma 9. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$, $\mathtt{A} \in \mathcal{F}$ and $P \overset{s\checkmark}{\Longrightarrow}_p \mathtt{nil}$.

1. $\mathtt{A} \in s$ if and only if $P \Rightarrow \mathtt{A} \overset{s\checkmark}{\Longrightarrow}_p \mathtt{nil}$.
2. $\mathtt{A} \notin s$ if and only if $P \Rightarrow \mathtt{A} \overset{s\mathtt{A}\checkmark}{\Longrightarrow}_p \mathtt{nil}$.

Proof. In both cases the proof is made by induction of the length of s.

$\square$

Lemma 10. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$, $\mathtt{A} \in \mathcal{F}$, $s \in \mathcal{F}^*$ and $p \in (0,1)$. $P \xRightarrow{s\checkmark}_p \mathtt{nil}$, if and only if $A \backslash \mathtt{P} \xRightarrow{s\checkmark}_p \mathtt{nil}$ and $\mathtt{A} \notin s$.

Proof. The proof is simply by induction on the length of s.

$\square$

Lemma 11. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$, $\mathtt{A}, \mathtt{B} \in \mathcal{F}$, $s \in \mathcal{F}^*$ and $p \in (0,1)$. Then $P \xRightarrow{s\checkmark}_p \mathtt{nil}$ if and only if $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xRightarrow{s'\checkmark}_p \mathtt{nil}$ and s' is in the form: $\mathtt{A} \notin s$ and $s' = s$, $\mathtt{B} \in s$ and $s' = s$, or $\mathtt{A} \in s$, $\mathtt{B} \notin s$ and $s' = s \cdot \mathtt{B}$.

Proof. By induction of the length of s.

$|s| = 0$ In this case $P \xrightarrow{\checkmark}_p \mathtt{nil}$. We obtain the result applying the rule [**req3**].

$|s| > 0$ Now we can distinguish three cases depending on the first feature of s:

- $s = \mathtt{A}s_1$**.** In this case there exist $p_1, q \in (0,1)$ such that $P \xrightarrow{\mathtt{A}}_{p_1} P_1 \xRightarrow{s_1\checkmark}_q \mathtt{nil}$. When applying the rule [**req2**] we obtain $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{A}}_{p_1} P_1 \Rightarrow \mathtt{B}$. We obtain the result by applying the lemma 9.
- $s = \mathtt{B}s_1$**.** In this case there exist $p_1, q \in (0,1)$ such that $P \xrightarrow{\mathtt{A}}_{p_1} P_1 \xRightarrow{s_1\checkmark}_q \mathtt{nil}$. When applying the rule [**req2**] we obtain $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{B}}_{p_1} P_1 \Rightarrow \mathtt{A}$. We obtain the result by applying the lemma 9.
- $s = \mathtt{C}s_1$ **with** $\mathtt{C} \neq \mathtt{A}$ **and** $\mathtt{C} \neq \mathtt{A}$**.** In this case there exist $p_1, q \in (0,1)$ such that $P \xrightarrow{\mathtt{C}}_{p_1} P_1 \xRightarrow{s_1\checkmark}_q \mathtt{nil}$. When applying the rule [**req1**], we obtain $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{C}}_{p_1} \mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P_1$, and then the result by applying the inductive hypothesis over s_1.

$\square$

Lemma 12. Let $P \in \mathtt{SPLA}^{\mathcal{P}}$, $\mathtt{A}, \mathtt{B} \in \mathcal{F}$, $s \in \mathcal{F}^*$ and $p \in (0,1)$. Then $P \xRightarrow{s\checkmark}_p \mathtt{nil}$ if and only if $\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P \xRightarrow{s\checkmark}_p \mathtt{nil}$, $\mathtt{A} \notin s$ and $\mathtt{B} \notin s$.

Proof. By the induction on the length of s.

$|s| = 0$ In this case $P \xrightarrow{\checkmark}_p \mathtt{nil}$. We obtain the result by applying the rule [**excl4**].

$|s| > 0$ Now it is possible to distinguish three cases depending on the first feature of s:

- $s = \mathtt{A}s_1$**.** In this case there exist $p_1, q \in (0,1)$ such that $P \xrightarrow{\mathtt{A}}_{p_1} P_1 \xRightarrow{s_1\checkmark}_q \mathtt{nil}$. When applying rule [**req2**] we obtain $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{A}}_{p_1} P_1 \backslash \mathtt{B}$. Now based on Lemma 9,
  - $\mathtt{B} \in s_1$ if and only if $P_1 \Rightarrow \mathtt{B} \xRightarrow{s_1 \cdot \checkmark}_q \mathtt{nil}$.
  - $\mathtt{B} \notin s_1$ if and only if $P_1 \Rightarrow \mathtt{B} \xRightarrow{s_1\mathtt{B}\checkmark}_q \mathtt{nil}$.
- $s = \mathtt{C}s_1$ **with** $\mathtt{C} \neq \mathtt{A}$**.** In this case there exist $p_1, q \in (0,1)$ such that $P \xrightarrow{\mathtt{C}}_{p_1} P_1 \xRightarrow{s_1\checkmark}_q \mathtt{nil}$. When applying rule [**req1**], we obtain $\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\mathtt{C}}_{p_1} \mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P_1$, and then the result is obtained by applying the inductive hypothesis over s_1.

$\square$

B Proof of Proposition 4

Proposition 6. $P[\mathcal{A}] \stackrel{s}{\Longrightarrow}_r Q[\mathcal{A}]$ if and only if $r = \sum \lbag p \mid P \stackrel{s'}{\Longrightarrow}_p Q,\ s = s'[\mathcal{A}] \rbag$

Proof. The proof is achieved by induction over the length of the trace s. If the length is zero the result is trivial. Then we suppose that $s = \mathtt{A} \cdot s_1$. If $\mathtt{A} = \bot$ then any transition $P[\mathcal{A}] \stackrel{s}{\Longrightarrow}_p Q[\mathcal{A}]$ can be divided in transitions, possibly more than one, for example.

$$P[\mathcal{A}] \xrightarrow{\bot}_{r_1} P_1[\mathcal{A}] \stackrel{s_1}{\Longrightarrow}_{r2} Q$$

then we have

$$r = \sum \lbag p \mid P[\mathcal{A}] \stackrel{s}{\Longrightarrow}_p Q \rbag = \sum \lbag r_1 \cdot r2 \mid P[\mathcal{A}] \xrightarrow{\bot}_{r_1} P_1[\mathcal{A}] \stackrel{s_1}{\Longrightarrow}_{r2} Q \rbag =$$
$$\sum \lbag r_1' \cdot r2 \mid P[\mathcal{A}] \xrightarrow{\mathtt{B}}_{r_1'} P_1'[\mathcal{A}] \stackrel{s_1}{\Longrightarrow}_{r2} Q,\ \mathtt{B} \in \mathcal{A} \rbag$$

Now for each r_1', we can apply the induction hypothesis to each of the transitions $P_1'[\mathcal{A}] \stackrel{s_1}{\Longrightarrow}_{r2} Q$ to obtain $r_2 = \sum \lbag r2' \mid P_1 \stackrel{s_1'}{\Longrightarrow} Q,\ s_1 = s_1'[\mathcal{A}] \rbag$. Continuing the last equation:

$$\sum \lbag r_1' \cdot r2 \mid P[\mathcal{A}] \xrightarrow{\mathtt{B}}_{r_1'} P_1'[\mathcal{A}] \stackrel{s_1}{\Longrightarrow}_{r2} Q,\ \mathtt{B} \in \mathcal{A} \rbag =$$
$$\sum \lbag r_1' \cdot r2' \mid P[\mathcal{A}] \xrightarrow{\mathtt{B}}_{r_1'} P_1' \stackrel{s_1'}{\Longrightarrow}_{r2'} Q,\ \mathtt{B} \in \mathcal{A}.\ s_1 = s_1'[\mathcal{A}] \rbag =$$
$$\sum \lbag r_1 \cdot r2' \mid P[\mathcal{A}] \xrightarrow{\bot}_{r_1} P_1 \stackrel{s_1'}{\Longrightarrow}_{r2'} Q,\ \mathtt{B} \in \mathcal{A}.\ s_1 = s_1'[\mathcal{A}] \rbag =$$
$$\sum \lbag r \mid P \stackrel{s'}{\Longrightarrow}_r Q,\ s = s'[\mathcal{A}] \rbag$$

The case $\mathtt{A} \notin \mathcal{A}$ is similar to the last one: we just skip the step from $\mathtt{B}$ to $\bot$.

$\square$

Proof of Proposition 4 $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(P[\mathcal{A}])$ if and only if

$$p = \sum \lbag r \mid P[\mathcal{A}] \stackrel{s\checkmark}{\Longrightarrow}_r P'[\mathcal{A}].\ pr = \lfloor s \rfloor \rbag =$$
$$\sum \lbag r \mid P \stackrel{s'\checkmark}{\Longrightarrow}_r P',\ s = s'[\mathcal{A}],\ pr = \lfloor s \rfloor \rbag =$$
$$\sum \lbag r \mid P \stackrel{s'\checkmark}{\Longrightarrow}_r P',\ s = pr[\mathcal{A}] \rbag =$$
$$\sum \lbag r \mid (pr', r) \in \mathtt{prod}^{\mathcal{P}}(P),\ pr' = pr[\mathcal{A}] \rbag =$$

So, $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(P[\mathcal{A}])$ if and only if $(pr, p) \in [\![(\mathtt{prod}^{\mathcal{P}}(P))[\mathcal{A}]]\!]^{\mathcal{P}}$