# 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: 2018-11-21 11:27

**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. 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 tool. We briefly comment on the characteristics of the tool 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" [33]. 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 line organization can be found in the literature, such as FODA [28], RSEB [22] and PLUSS [29,20].

* 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).

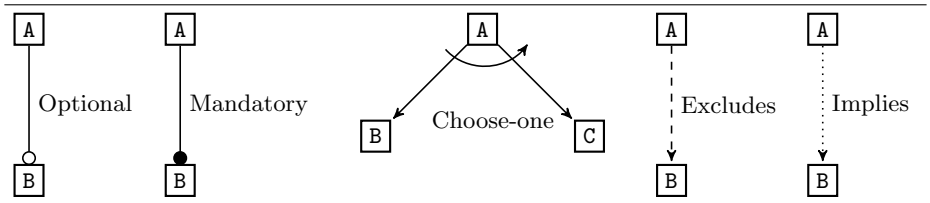


**Fig. 1.** FODA Diagram representation.

Graphical approaches are commonly used to model SPLs. Feature Oriented Domain Analysis [28] (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 [35]. In this case, the original graphical representation of FODA must be provided with a formal semantics [7]. This issue is solved by using SPLA [1], 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 [41,26,27,1], while others use either propositional or first order logic [32,13,4,3,36].

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 [**?**]. This cost may represent different aspects of a feature, such us 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. The obtained results are promising. 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 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) [30,21,11,37,31,9,18,16,23,2,40,24,17,14]. Although the addition of probabilistic information to model SPLs is relatively new, there are already several proposals in the literature [12,6,5,19]. In particular, very recent work [19] shows that statistic analysis allows users to determine relevant characteristics, like the certainty of finding valid products among complex models. Another approach [12] 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. 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 [6,5]. These models demonstrate their ability to be integrated into standard tools like QFLan, Microsoft's SMT Z3 and MultiVeStA.

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 [1,8], 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.
- 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 [39].
- A complete comparison between the overall implementation performance of the probabilistic extension and the denotational semantic implementation from SPLA [1]. 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 presents our language. Section 3 is used to prove the equivalence between the operational and denotational semantics. In section 4 we extend our language to define how sets of features can be hidden. Section 5 presents an empirical study that has been carried out by using our implementation of the denotational semantics for the probabilistic extension. Finally, section 6 presents our conclusions and some lnes of future work.

## 2 $\texttt{SPLA}^{\mathcal{P}}$: 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.

### 2.1 Syntax and operational semantics

Following our previous work [1,8], 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 SPLA, where probabilities are introduced 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 [1,8]. 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 $\cdot \vee \cdot$, $\cdot \wedge \cdot$, $\texttt{A};\cdot$, $\overline{\texttt{A}};\cdot$, $\texttt{A} \Rightarrow \texttt{B in } \cdot$, $\texttt{A} \not\Rightarrow \texttt{B in } \cdot$, that directly correspond to relationships in 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:

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

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 are going to assume left-associativity in binary operators and the following precedence in the operators (from higher to lower priority): $\texttt{A}; P$, $\overline{\texttt{A}};_p P$, $P \vee_p Q$, $P \wedge Q$, $\texttt{A} \not\Rightarrow \texttt{B in } P$, $\texttt{A} \Rightarrow \texttt{B in } P$, $\texttt{A} \Rightarrow \texttt{B in } P$, $P \backslash \texttt{A}$, and $P \Rightarrow \texttt{A}$. In [1,8] 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.

The Extended BNF in Definition 1 says that a term of SPLA is a sequence of operators and features.

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 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 $\texttt{nil}$. During the computation of an $\texttt{SPLA}^{\mathcal{P}}$ term, we have to represent the situation in which a

*valid product* of the term has been computed. This fact is represented by the ✓ symbol.

The operators $\mathtt{A}; P$ and $\overline{\mathtt{A}}_p; P$ add the feature A to any product that can be obtained from $P$. The operator $\mathtt{A}; P$ indicates that A is mandatory while $\overline{\mathtt{A}}_p; P$ indicates that 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 the *choose-one* operator while the second one represents the *conjunction* operator.

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

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

In Figure 2 we present the set of rules formally defining the operational behavior of $\mathtt{SPLA}^{\mathcal{P}}$. These rules essentially coincide with the ones corresponding to SPLA [1], but with the addition of probabilities. 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 [15]. Rule [**con3**] requires that both branches agree on the termination of a product.

We use *multisets* of transitions to consider different occurrences of the same transition. So, if a transition can be derived in several ways, then each derivation generates a different instance of this transition [25]. For example, let us consider the term $P = \mathtt{A}; ✓ \vee_{\frac{1}{2}} \mathtt{A}; ✓$. If we were not careful, then we would have the transition $P \xrightarrow{\mathtt{A}}_{\frac{1}{2}} ✓$ 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 nil.

$$
\begin{array}{ll}
[\textbf{tick}]\ \checkmark \xrightarrow{\checkmark}_1 \mathtt{nil} & [\textbf{feat}]\ \mathtt{A}; P \xrightarrow{\mathtt{A}}_1 P \\
[\textbf{ofeat1}]\ \overline{\mathtt{A}};_p P \xrightarrow{\mathtt{A}}_p P & [\textbf{ofeat2}]\ \overline{\mathtt{A}};_p P \xrightarrow{\checkmark}_{(1-p)} \mathtt{nil} \\
[\textbf{cho1}]\ \dfrac{P \xrightarrow{\mathtt{A}}_p P_1}{P \vee_q Q \xrightarrow{\mathtt{A}}_{p\cdot q} P_1} & [\textbf{cho2}]\ \dfrac{Q \xrightarrow{\mathtt{A}}_q Q_1}{P \vee_p Q \xrightarrow{\mathtt{A}}_{(1-p)\cdot q} Q_1} \\
[\textbf{con1}]\ \dfrac{P \xrightarrow{\mathtt{A}}_p P_1}{P \wedge Q \xrightarrow{\mathtt{A}}_{\frac{p}{2}} P_1 \wedge Q} & [\textbf{con2}]\ \dfrac{Q \xrightarrow{\mathtt{A}}_q Q_1}{P \wedge Q \xrightarrow{\mathtt{A}}_{\frac{q}{2}} P \wedge Q_1} \\
[\textbf{con3}]\ \dfrac{P \xrightarrow{\checkmark}_q \mathtt{nil}, Q \xrightarrow{\checkmark}_p \mathtt{nil}}{P \wedge Q \xrightarrow{\checkmark}_{p\cdot q} \mathtt{nil}} & \\
[\textbf{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} & [\textbf{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} \\
[\textbf{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} & [\textbf{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}} \\
[\textbf{req3}]\ \dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}}{\mathtt{A} \Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\checkmark}_p \mathtt{nil}} & \\
[\textbf{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} & [\textbf{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}} \\
[\textbf{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}} & [\textbf{excl4}]\ \dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}}{\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P \xrightarrow{\checkmark}_p \mathtt{nil}} \\
[\textbf{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}} & [\textbf{forb2}]\ \dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}}{P \backslash \mathtt{A} \xrightarrow{\checkmark}_p \mathtt{nil}} \\
[\textbf{mand1}]\ \dfrac{P \xrightarrow{\checkmark}_p \mathtt{nil}}{P \Rightarrow \mathtt{A} \xrightarrow{\mathtt{A}}_p \checkmark} & [\textbf{mand2}]\ \dfrac{P \xrightarrow{\mathtt{A}}_p P_1}{P \Rightarrow \mathtt{A} \xrightarrow{\mathtt{A}}_p P_1} \\
[\textbf{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}} & \\
\multicolumn{2}{c}{\mathtt{A}, \mathtt{B}, \mathtt{C} \in \mathcal{F},\ a \in \mathcal{F} \cup \{\checkmark\}}
\end{array}
$$

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

**Lemma 1.** Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$ and $p \in (0, 1]$. We have $P \xrightarrow{\checkmark}_p Q$ if and only if $Q = \mathtt{nil}$. □

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

**Definition 2.** Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$. We write $P \overset{s}{\Longrightarrow}_p Q$ 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 $p = p_1 \cdot p_2 \cdot \cdots p_n$. 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

$$\{(pr, p) \mid p > 0 \wedge p = \sum \wr q \mid P \overset{s\checkmark}{\Longrightarrow}_q Q \ \wedge\ \lfloor s \rfloor = pr \wr\}$$

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

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 and that the probabilities of products belong to $[0, 1]$.

**Lemma 2.** Let $P,\ Q \in \texttt{SPLA}^{\mathcal{P}}$.We have the following results.

1. If $P \xrightarrow{\texttt{A}}_p Q$ then $p \in (0, 1]$. If $P \stackrel{s}{\Longrightarrow}_p Q$ then $p \in (0, 1]$.
2. $\sum \wr p \mid \exists \texttt{A} \in \mathcal{F},\ Q \in \texttt{SPLA}^{\mathcal{P}} :\ P \xrightarrow{\texttt{A}}_p Q \wr \in [0, 1]$.
3. $\sum \wr p \mid \exists s \in \mathcal{F}^*,\ Q \in \texttt{SPLA}^{\mathcal{P}} :\ P \xrightarrow{s\checkmark}_p Q \wr \in [0, 1]$.
4. $\texttt{TotProb}(P) \in [0, 1]$.

□

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 [1]. In our case, we can define consistency by having $\texttt{TotProb}(P) > 0$. We will prove that a translation from our probabilistic framework into the non-probabilistic one keeps consistency in the expected way.

**Definition 3.** We define the translation function $\texttt{np} : \texttt{SPLA}^{\mathcal{P}} \mapsto \texttt{SPLA}$ as follows:

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

□

The proof of the following result is straightforward by taking into account that our operational semantics rules are the same, if we discard probabilities, as in [1]. 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 $P, Q \in \texttt{SPLA}^{\mathcal{P}}$. We have $P \stackrel{s}{\Longrightarrow}_p Q$ if and only if $\texttt{np}(P) \stackrel{s}{\Longrightarrow} \texttt{np}(Q)$. Moreover, we have $pr \in \texttt{prod}(\texttt{np}(P))$ if and only if there exists $p > 0$ such that $(pr, p) \in \texttt{prod}^{\mathcal{P}}(P)$. □

### 2.2 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 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 4.** 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 will 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 `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 `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 obtain a single product with probability 1.

**Definition 5.** 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 2.1 we define its denotational operator as follows:

- $[\![\texttt{nil}]\!]^{\mathcal{P}} = \varnothing$
- $[\![\checkmark]\!]^{\mathcal{P}} = \{(\varnothing, 1)\}$
- $[\![\texttt{A};\cdot]\!]^{\mathcal{P}}(M) = \texttt{accum}\big(\wr(\{\texttt{A}\} \cup P, p) \mid (P, p) \in M \int\big)$
- $[\![\overline{\texttt{A}};_p \cdot]\!]^{\mathcal{P}}(M) = \texttt{accum}\big(\wr(\varnothing, 1 - p) \int \uplus \wr\, (\{\texttt{A}\} \cup P, p \cdot q) \mid (P, q) \in M \int\big)$
- $[\![\cdot \vee_p \cdot]\!]^{\mathcal{P}}(M_1, M_2) = \texttt{accum}\begin{pmatrix} \wr(P, p \cdot q) \mid (P, q) \in M_1 \int \uplus \\ \wr(Q, (1 - p) \cdot q) \mid (Q, q) \in M_2 \int \end{pmatrix}$

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

□

The denotational semantics for the prefix operator $[\![\texttt{A};\cdot]\!]^{\mathcal{P}}(M)$ and the denotational semantics for the operator $[\![\texttt{A} \Rightarrow \texttt{B in } \cdot]\!]^{\mathcal{P}}(M)$ 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. So, 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 $\texttt{A}, \texttt{B} \in \mathcal{F}$ be features. We have:

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

□

## 3 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. The proof of the following result is an immediate consequence of Lemmas 3–11 (see Appendix A of the paper).

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

**Fig. 3.** Operational semantics for the hiding operator

**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:

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

□

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 $(pr, p) \in \mathtt{prod}^{\mathcal{P}}(P)$. □

## 4 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 we are going to *hide* other features. 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. So, 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 6.** 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 3. 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 7.** 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:

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

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 $\bot$ in $s$. □

**Definition 8.** Let $M \in \mathcal{M}$ and $\mathcal{A} \subseteq \mathcal{F}$. We define:

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

□

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 $\mathtt{prod}^{\mathcal{P}}(P[\mathcal{A}]) = [\![\mathtt{prod}^{\mathcal{P}}(P)[\mathcal{A}]]\!]^{\mathcal{P}}$. □

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 $\bot$. 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. The proof of the following result is easy by structural induction and by Proposition 4.

**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:

$$[\![\checkmark[\mathcal{A}]]\!]^{\mathcal{P}} = [\![\checkmark]\!]^{\mathcal{P}}$$

$$[\![\mathtt{nil}[\mathcal{A}]]\!]^{\mathcal{P}} = [\![\mathtt{nil}]\!]^{\mathcal{P}}$$

$$[\![(\mathtt{A}; P)[\mathcal{A}]]\!]^{\mathcal{P}} = \begin{cases} [\![\bot; (P[\mathcal{A}])]\!]^{\mathcal{P}} & \text{if } A \in \mathcal{A} \\ [\![\mathtt{A}; (P[\mathcal{A}])]\!]^{\mathcal{P}} & \text{if } A \notin \mathcal{A} \end{cases}$$

$$[\![(\overline{\mathtt{A}};_r P)[\mathcal{A}]]\!]^{\mathcal{P}} = \begin{cases} [\![\overline{\bot};_r (P[\mathcal{A}])]\!]^{\mathcal{P}} & \text{if } A \in \mathcal{A} \\ [\![\overline{\mathtt{A}};_r (P[\mathcal{A}])]\!]^{\mathcal{P}} & \text{if } A \notin \mathcal{A} \end{cases}$$

$$[\![(P \vee_P Q)[\mathcal{A}]]\!]^{\mathcal{P}} = [\![(P[\mathcal{A}]) \vee_P (Q[\mathcal{A}])]\!]^{\mathcal{P}}$$

$$[\![(P \wedge Q)[\mathcal{A}]]\!]^{\mathcal{P}} = [\![(P[\mathcal{A}]) \wedge (Q[\mathcal{A}])]\!]^{\mathcal{P}}$$

$$\text{If } \mathtt{A}, \mathtt{B} \notin \mathcal{A} \text{ then } [\![(\mathtt{A} \Rightarrow \mathtt{B} \text{ in } P)[\mathcal{A}]]\!]^{\mathcal{P}} = [\![\mathtt{A} \Rightarrow \mathtt{B} \text{ in } (P[\mathcal{A}])]\!]^{\mathcal{P}}$$

$$\text{If } \mathtt{A}, \mathtt{B} \notin \mathcal{A} \text{ then } [\![(\mathtt{B} \not\Rightarrow \mathtt{P} \text{ in })[\mathcal{A}]]\!]^{\mathcal{P}} = [\![\mathtt{A} \not\Rightarrow \mathtt{B} \text{ in } (P[\mathcal{A}])]\!]^{\mathcal{P}}$$

□

## 5 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 that 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.

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 developed a tool to implement the denotational semantics of the probabilistic extension presented in this paper.

The source code of this tool is available at the main project site [3]. 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.

Firstly, we have carried out an experiment to show the computing time required to calculate the probability of having each feature in a product. In order to run this experiment, a variability model consisting of 1500 features has been used. This variability model was randomly generated using BeTTy [39]. The parameters for generating the BeTTy 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* relation is 0.25.
- The probability of having a feature in a *conjunction* relation is 0.25.

Figure 4 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 a feature in a 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 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, 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.

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

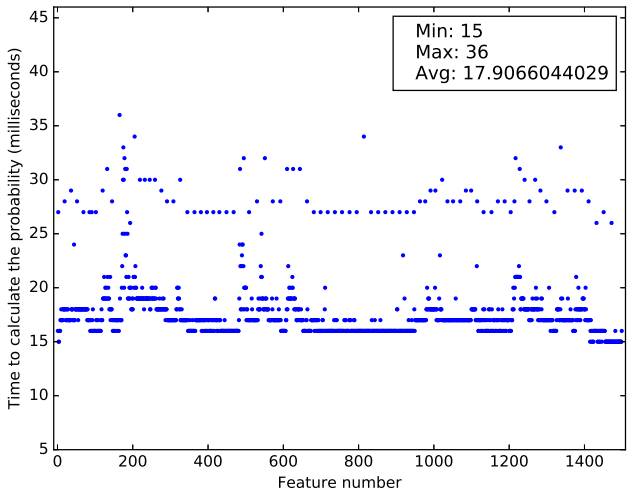


**Fig. 4.** Computing time analysis for a model consisting of 1500 features.

The model used in the experiments has been executed several times, providing similar results. That is, the major part of the features require between 15 and 20 milliseconds to be processed, while there is a small portion of them requiring a processing time between 20 and 38 milliseconds.

Figure 5 shows the probability of each feature in the analyzed model, where the $x$-axis represents the feature ID and the $y$-axis represents the probability. For the sake of clarity, we provide a chart by sorting the features using its probability as sorting criteria (see Figure 6). This chart clearly shows that there exist different groups of features having a similar probability. In this case, 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, we can ensure that at least, the 75% of the generated products, are tested.

Secondly, an experiment 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. 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.

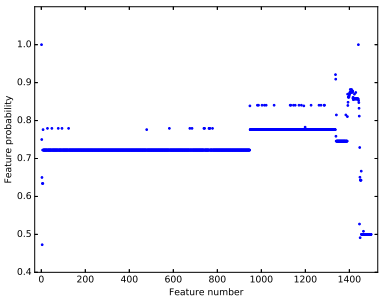


**Fig. 5.** Probabilistic analysis for a 1500 features model.

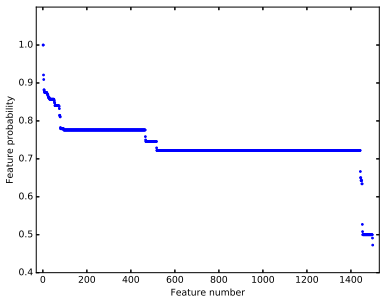


**Fig. 6.** Probabilistic analysis for a 1500 features model, decreasingly sorted.

Table 1 shows the used configurations to generate the variability models for this experiment, where *Conf. ID* represents the configuration ID, *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.

| **Conf. ID** | **Mandatory** | **Optional** | **Choose-one** | **Conjunction** |
|---|---|---|---|---|
| Config_1 | 0.58 | 0.15 | 0.15 | 0.01 |
| Config_2 | 0.7 | 0.15 | 0.15 | 0.2 |
| Config_3 | 0.2 | 0.15 | 0.15 | 0.5 |

**Table 1.** Configuration of the scalability experiments.

Figure 7 and figure 8 show the execution time and the required amount of memory, respectively, for processing the variability models generated using *Config_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. 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 *Config_1*: it requires, in the worst case scenario, 215 ms and 0.32 GB of RAM to process the model.

Figure 9 and figure 10 show the results for analyzing the models generated using *Config_2*. It is important to remark that 20% of the features in the generated

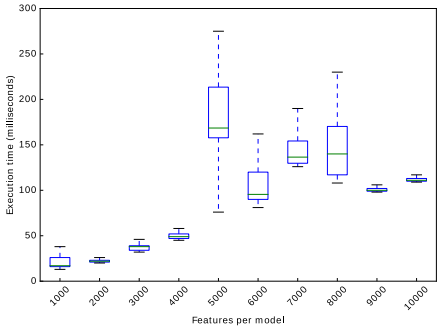


**Fig. 7.** Execution time for processing models generated with *Config_1*.

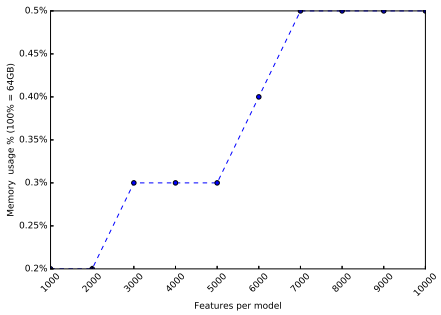


**Fig. 8.** Memory usage for processing models generated with *Config_1*.

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.

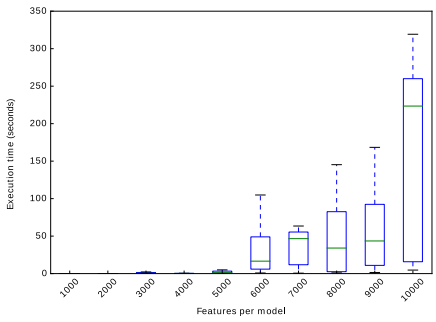


**Fig. 9.** Execution time for processing models generated with *Config_2*.

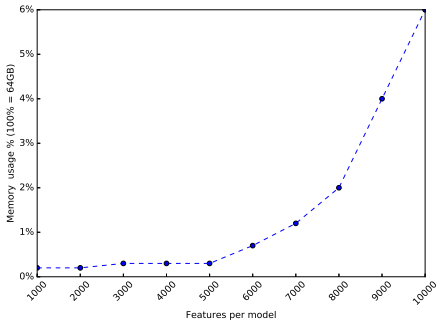


**Fig. 10.** Memory usage for processing models generated with *Config_2*.

Figure 11 and figure 12 show the results for processing the models generated using *Config_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 *Config_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.

Broadly speaking, 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 *Config_3* with 9,000 and 10,000 features cannot be processed using 64 GB of RAM.

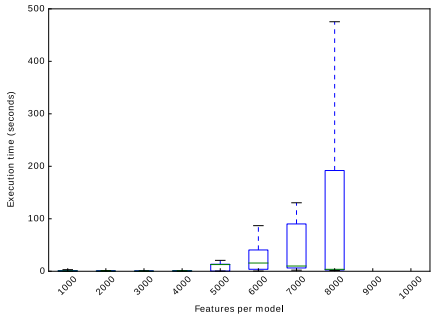


**Fig. 11.** Execution time for processing models generated with *Config_3*.

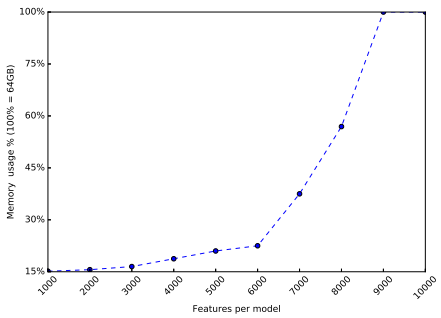


**Fig. 12.** Memory usage for processing models generated with *Config_3*.

### 5.1 Comparison between current results and previous implementations

We now compare the latest results with the ones obtained using `SPLA` [1] and the cost extension $\texttt{SPLA}^{\mathcal{C}}$ [8].

Specifically we want to compare the implementation performance between the implementation of the probabilistic extension and the implementation of the denotational semantic from `SPLA` [1].

Figure 13 shows that the experiments ran with a number of different features ranging from 50 to 300.

In this paper we managed to calculate models up to 13000 features without timeouts.

| 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. 13.** Denotational benchmark from [1].

Also based on the results published in $\texttt{SPLA}^{\mathcal{C}}$ [8] the models used for the simulations were processed in an 8-node cluster but the model had only 17 features.

Figure 14 shows the algorithm behavior which best time its approximately 300 seconds.

For those specific implementations and simulations we can highlight that the denotational semantics of the probabilistic extension implementation improves dramatically the performance in comparison with the denotational semantics implementations presented in previous works [1,**?**].

## 6 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

| | 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 |

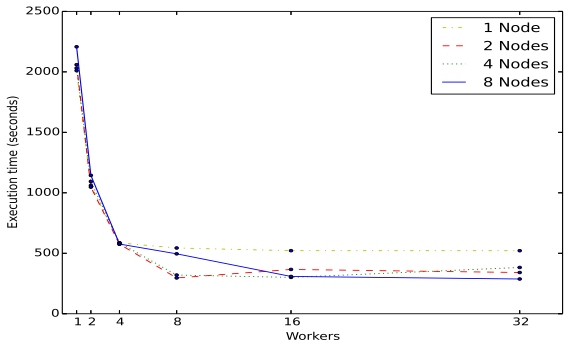


**Fig. 14.** Cluster execution times for 1, 2, 4 and 8 nodes; and 1, 2, 4, 8, 16 and 32 workers from [8].

of the probabilistic analysis. In addition, we plan to find practical use cases to show the usefulness of having a probabilistic extension for `SPLs`.

## References


1. Andrés, C., Camacho, C., Llana, L.: A formal framework for software product lines. Information & Software Technology 55(11), 1925–1947 (2013)
2. Andrés, M.E., Palamidessi, C., Rossum, P.v., Sokolova, A.: Information hiding in probabilistic concurrent systems. Theoretical Computer Science 412(28), 3072–3089 (2011)
3. Asirelli, P., ter Beek, M.H., Fantechi, A., Gnesi, S.: A logical framework to deal with variability. In: 8th Int. Conf. on Integrated Formal Methods, IFM'10. pp. 43–58. Springer(2010)
4. Asirelli, P., ter Beek, M.H., Gnesi, S., Fantechi, A.: 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)
5. ter Beek, M.H., Legay, A., Lluch-Lafuente, A., Vandin, A.: Statistical analysis of probabilistic models of software product lines with quantitative constraints. In: Proceedings of the 19th International Conference on Software Product Line. pp. 11–15. SPLC '15, ACM (2015)
6. ter Beek, M.H., Legay, A., Lluch-Lafuente, A., Vandin, A.: Quantitative analysis of probabilistic models of software product lines with statistical model checking. In: Atlee, J.M., Gnesi, S. (eds.) Proceedings 6th Workshop on Formal Methods and Analysis in SPL Engineering, London, UK, 11 April 2015. Electronic Proceedings in Theoretical Computer Science, vol. 182, pp. 56–70. Open Publishing Association (2015)

7. Bontemps, Y., Heymans, P., Schobbens, P., Trigaux, J.: 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. Camacho, C., Llana, L., Núñez, A.: Cost-related interface for software product lines. Journal of Logical and Algebraic Methods in Programming 85, 227–244 (2016)
9. Cheung, L., Stoelinga, M., Vaandrager, F.: A testing scenario for probabilistic processes. Journal of the ACM 54(6), Article 29 (2007)
10. Chrszon, P., Dubslaff, C., Klüppelholz, S., Baier, C.: Profeat: feature-oriented engineering for family-based probabilistic model checking. Formal Aspects of Computing 30(1), 45–75 (2018)
11. Cleaveland, R., Dayar, Z., Smolka, S., Yuen, S.: Testing preorders for probabilistic processes. Information and Computation 154(2), 93–148 (1999)
12. Cordy, M., Heymans, P., Schobbens, P., Sharifloo, A.M., Ghezzi, C., Legay, A.: Verification for reliable product lines. CoRR abs/1311.1343 (2013)
13. Czarnecki, K., Wasowski, A.: 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. D'Argenio, P.R., Gebler, D., Lee, M.D.: A general SOS theory for the specification of probabilistic transition systems. Information and Computation 249, 76–109 (2016)
15. D'Argenio, P., Hermanns, H., Katoen, J.P.: On generative parallel composition. In: Workshop on Probabilistic Methods in Verification, PROBMIV'98, ENTCS 22. pp. 30–54. Elsevier (1999)
16. Deng, Y., Glabbeek, R.v., Hennessy, M., Morgan, C.: Characterising testing preorders for finite probabilistic processes. Logical Methods in Computer Science 4(4) (2008)
17. Deng, Y., Glabbeek, R.v., Hennessy, M., Morgan, C.: Real-reward testing for probabilistic processes. Theoretical Computer Science 538, 16–36 (2014)
18. Deng, Y., Palamidessi, C.: Axiomatizations for probabilistic finite-state behaviors. Theoretical Computer Science 373(1-2), 92–114 (2007)
19. Devroey, X., Perrouin, G., Cordy, M., Samih, H., Legay, A., Schobbens, P.Y., Heymans, P.: Statistical prioritization for software product line testing: an experience report. Software & Systems Modeling 16(1), 153–171 (2017)
20. Eriksson, M., Borstler, J., Borg, K.: 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)
21. Glabbeek, R.v., Smolka, S., Steffen, B.: Reactive, generative and stratified models of probabilistic processes. Information and Computation 121(1), 59–80 (1995)
22. Griss, M., Favaro, J.: Integrating feature modeling with the RSEB. In: 5th International Conference on Software Reuse, ICSR'98. pp. 76–85 (1998)
23. Hierons, R.M., Merayo, M.G.: Mutation testing from probabilistic and stochastic finite state machines. Journal of Systems and Software 82(11), 1804–1818 (2009)
24. Hierons, R.M., Núñez, M.: Using schedulers to test probabilistic distributed systems. Formal Aspects of Computing 24(4-6), 679–699 (2012)
25. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, New York, NY, USA (1996)
26. Höfner, P., Khédri, R., Möller, B.: Feature algebra. In: 14th International Symposium on Formal Methods, FM'06. LNCS, vol. 4085, pp. 300–315. Springer(2006)
27. Höfner, P., Khédri, R., Möller, B.: An algebra of product families. Software and System Modeling 10(2), 161–182 (2011)

28. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-Oriented Domain Analysis (FODA) feasibility study. Tech. Rep. CMU/SEI-90-TR-21, Carnegie Mellon University (1990)
29. Kollu, K.: Evaluating The PLUSS Domain Modeling Approach by Modeling the Arcade Game Maker Product Line. Ph.D. thesis, Umea University (2005)
30. Larsen, K., Skou, A.: Bisimulation through probabilistic testing. Information and Computation 94(1), 1–28 (1991)
31. López, N., Núñez, M., Rodríguez, I.: Specification, testing and implementation relations for symbolic-probabilistic systems. Theoretical Computer Science 353(1–3), 228–248 (2006)
32. Mannion, M.: Using first-order logic for product line model validation. In: 2nd International Software Product Line Conference, SPLC'02. pp. 176–187. Springer(2002)
33. McGregor, J.: Testing a software product line. Tech. Rep. CMU/SEI-2001-TR-022, Carnegie Mellon University, Software Engineering Institute (2001), `http://www.sei.cmu.edu/reports/01tr022.pdf`
34. de Moura, L., Bjorner, N.: Z3: An efficient smt solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
35. Nakajima, S.: Semi-automated diagnosis of foda feature diagram. In: 25th ACM Symposium on Applied Computing, SAC'10. pp. 2191–2197. ACM Press(2010)
36. Nummenmaa, J., Nummenmaa, T., Zhang, Z.: On the use of ltss to analyze software product line products composed of features. In: Sun, F., Li, T., Li, H. (eds.) Knowledge Engineering and Management, Advances in Intelligent Systems and Computing, vol. 214, pp. 531–541. Springer Berlin Heidelberg (2014)
37. Núñez, M.: Algebraic theory of probabilistic processes. Journal of Logic and Algebraic Programming 56(1–2), 117–177 (2003)
38. Sebastio, S., Vandin, A.: Multivesta: Statistical model checking for discrete event simulators. In: Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools. pp. 310–315. ValueTools '13, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2013)
39. Segura, S., Hierons, R., Benavides, D., Ruiz, A.: Automated metamorphic testing on the analyses of feature models. Information & Software Technology 53(3), 245–258 (2011)
40. Sokolova, A.: Probabilistic systems coalgebraically: A survey. Theoretical Computer Science 412(38), 5095–5110 (2011)
41. Sun, J., Zhang, H., Wang, H.: 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)

## 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

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

*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

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

then

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

□

**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

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

□

**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

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

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

□

**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

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

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

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

□

**Definition 9.** Let $s, s' \in \mathcal{F}^*$ be traces, we denote with $\mathsf{int}(s, s')$ the set of traces obtained by alternating $s$ and $s'$. □

**Lemma 7.** Let $P, Q \in \mathtt{SPLA}^{\mathcal{P}}$, $p, q \in (0,1)$ and $s, s' \in \mathcal{F}^*$ such $p = \sum \wr p' \mid P \overset{s\checkmark}{\Longrightarrow}_{p'} \mathtt{nil} \int$ and $q = \sum \wr q' \mid\mid\mid Q \overset{s'}{\Longrightarrow}_{q'} \mathtt{nil} \int$ then

$$p \cdot q = \sum \wr r \mid P \wedge Q \overset{s''\checkmark}{\Longrightarrow}_r \mathtt{nil} \ \wedge \ s'' \in \mathsf{int}(s, s) \int$$

*Proof.* By induction of $|s| + |s'|$.

$|s| + |s'| = 0$ Since $P \wedge Q \xrightarrow{\checkmark}_r \mathtt{nil}$ if and only if $P \xrightarrow{\checkmark}_{r_1} \mathtt{nil}$, $Q \xrightarrow{\checkmark}_{r_2} \mathtt{nil}$, and $r = r_1 \cdot r_2$, behold the following:

$$\begin{aligned}
\sum \wr r \mid P \wedge Q \xrightarrow{\checkmark}_r \mathtt{nil} \int = \\
\sum \wr r \mid P \xrightarrow{\checkmark}_{r_1} \mathtt{nil} \ \wedge \ Q \xrightarrow{\checkmark}_{r_2} \mathtt{nil} \ \wedge \ r = r_1 \cdot r_2 \int = \\
\sum \wr r_1 \cdot r_2 \mid P \xrightarrow{\checkmark}_{r_1} \mathtt{nil} \ \wedge \ Q \xrightarrow{\checkmark}_{r_2} \mathtt{nil} \int = \\
\sum \Big\wr r_1 \cdot \Big(\sum \wr r_2 \mid Q \xrightarrow{\checkmark}_{r_2} \mathtt{nil} \int\Big) \ \Big| \ P \xrightarrow{\checkmark}_{r_1} \mathtt{nil} \Big\int = \\
\sum \wr r_1 \cdot q \mid P \xrightarrow{\checkmark}_{r_1} \mathtt{nil} \int = q \cdot \sum \wr r_1 \mid P \xrightarrow{\checkmark}_{r_1} \mathtt{nil} \int = q \cdot p
\end{aligned} \tag{1}$$

$|s| + |s'| > 0$ Suppose that $|s| > 0$ (the case $|s'| > 0$ is symmetric). Let's consider that $s = \mathtt{A} \cdot s_1$. Now consider the multiset $\mathcal{P} = \{\!|P' \mid P \xrightarrow{\mathtt{A}}_{r'} P' \stackrel{s_1\checkmark}{\Longrightarrow}_{p'} \mathtt{nil}|\!\}$ and given $r_1 = \sum \{\!|r \mid P \xrightarrow{\mathtt{A}}_{r} P' \ \wedge \ P' \in \mathcal{P}|\!\}$ and $p_1 = \sum \{\!|r \mid P' \stackrel{s_1\checkmark}{\Longrightarrow}_{r} \mathtt{nil} \ \wedge \ P' \in \mathcal{P}|\!\}$. It easy to verify that $p = r_1 \cdot p_1$:

$$\begin{aligned}
p = \sum \{\!|r \mid P \stackrel{s\checkmark}{\Longrightarrow}_{r} \mathtt{nil}|\!\} = \\
\sum \{\!|r' \cdot p' \mid P \xrightarrow{\mathtt{A}}_{r'} P' \stackrel{s_1\checkmark}{\Longrightarrow}_{p'} \mathtt{nil}|\!\} = \\
\sum \left\{\!\!\left| r' \cdot \underbrace{\left(\sum \{\!|p' \mid P' \stackrel{s_1\checkmark}{\Longrightarrow}_{p'} \mathtt{nil}|\!\}\right)}_{p_1} \right| P \xrightarrow{\mathtt{A}}_{r'} P' \ \wedge \ P' \in \mathcal{P} \right\}\!\!\right| = \\
p_1 \cdot \sum \{\!|r' \mid P \xrightarrow{\mathtt{A}}_{r'} P' \ \wedge \ P' \in \mathcal{P}|\!\} = r_1 \cdot p_1
\end{aligned} \tag{2}$$

For any $P' \in \mathcal{P}$, and then for the inductive hypothesis we obtain

$$p' \cdot q = \sum \{\!|r \mid P' \wedge Q \stackrel{s''\checkmark}{\Longrightarrow}_{r} \ \wedge s'' \in \mathsf{int}(s_1, s')|\!\}$$

So forth

$$\begin{aligned}
\sum \{\!|r \mid P' \in \mathcal{P} \ \wedge \ P' \wedge Q \stackrel{s''\checkmark}{\Longrightarrow}_{r} \ \wedge s'' \in \mathsf{int}(s_1, s')|\!\} = \\
\sum \{\!|p' \cdot q \mid P' \in \mathcal{P} \ \wedge \ P' \stackrel{s_1\checkmark}{\Longrightarrow}_{p'} \mathtt{nil} \ \wedge \ s'' \in \mathsf{int}(s_1, s')|\!\} \\
q \cdot \sum \{\!|r \mid P' \stackrel{s_1\checkmark}{\Longrightarrow}_{\mathtt{nil}} \ \wedge P' \in \mathcal{P}|\!\} = p_1 \cdot q
\end{aligned} \tag{3}$$

then for any $P' \in \mathcal{P}$ we obtain $P \wedge Q \stackrel{s''\checkmark}{\Longrightarrow}_{r} \mathtt{nil}$ where $P \xrightarrow{\mathtt{A}}_{r'} P'$, $r = \frac{r'}{2} \cdot p' \cdot q$ and $s''$ is the alternation of $s$ and $s'$. So forth

$$\begin{aligned}
\sum \{\!|r \mid P \wedge Q \xrightarrow{\mathtt{A}}_{\frac{r_1}{2}} P' \wedge Q \stackrel{s''\checkmark}{\Longrightarrow}_{r_2} \mathtt{nil} \ \wedge \ P' \xrightarrow{\mathtt{A}}_{r_1} P' \wedge \\
r = \frac{r_1}{2} \cdot r_1 \ \wedge \ s'' \in \mathsf{int}(s_1, s')|\!\} = \\
\sum \left\{\!\!\left| \frac{r'}{2} \cdot r'' \right| P \xrightarrow{\mathtt{A}}_{r'} P' \ \wedge \ P' \wedge Q \stackrel{s''\checkmark}{\Longrightarrow}_{r''} \mathtt{nil} \ \wedge \ s'' \in \mathsf{int}(s_1, s') \right\}\!\!\right| = \\
\sum \left\{\!\!\left| \frac{r'}{2} \cdot \left( \underbrace{\sum \left\{\!\!\left| r'' \right| \begin{array}{r} P' \wedge Q \stackrel{s''\checkmark}{\Longrightarrow}_{r''} \mathtt{nil} \wedge \\ s'' \in \mathsf{int}(s_1, s') \end{array} \right\}\!\!\right|}_{\text{induction hypothesis: } p_1 \cdot q} \right) \right| P \xrightarrow{\mathtt{A}}_{r'} P' \ \wedge \ P' \in \mathcal{P} \right\}\!\!\right| = \\
\sum \{\!|\frac{r'}{2} \cdot (p_1 \cdot q) \mid P \xrightarrow{\mathtt{A}}_{r'} P' \ \wedge \ P' \in \mathcal{P}|\!\} = \\
\frac{(p_1 \cdot q)}{2} \cdot \sum \{\!|r' \mid P \xrightarrow{\mathtt{A}}_{r'} P' \ \wedge \ P' \in \mathcal{P}|\!\} = \\
\frac{1}{2}(p_1 \cdot q \cdot r_1) = \frac{1}{2}(p \cdot q)
\end{aligned} \tag{4}$$

Equation 4 groups all probabilities of alternating $s$ and $s'$ where the first action of $s$ happens in the first place. Then we have to group the probabilistic information when the first action of $s'$ happens in the first place.

There are two possibilities, depending on whether these actions exist or not. Depending on $s'$ we obtain the following options.

$|s'| = 0$ In this case we have $q = \sum \wr q' \mid Q \xrightarrow{\checkmark}_{q'} \texttt{nil} \int$, If $Q \xrightarrow{\checkmark}_{q'} \texttt{nil}$, when applying the rule [**con4**] and then obtain $P \wedge Q \xrightarrow{\texttt{A}}_{\frac{r' \cdot q'}{2}} P_1$. In this case we obtain

$$\begin{aligned}
\sum \wr r \mid P \wedge Q \xrightarrow{\texttt{A}}_{\frac{q' \cdot r'}{2}} P' \xRightarrow{s_1\checkmark}_{p'} \texttt{nil} \ \wedge \ r = p' \cdot \frac{q' \cdot r'}{2} \int = \\
\sum \wr \frac{1}{2} \cdot (p' \cdot r' \cdot q' \mid P \wedge Q \xrightarrow{\texttt{A}}_{\frac{q' \cdot r'}{2}} P' \xRightarrow{s_1\checkmark}_{p'} \texttt{nil} \int = \\
\sum \wr \frac{1}{2} \cdot (p' \cdot r' \cdot (\underbrace{\sum \wr q' \mid Q \xrightarrow{\checkmark}_{q'} \texttt{nil} \int}_{q}) \mid P \xrightarrow{\texttt{A}}_{r'} P' \ \wedge \ P' \xRightarrow{s_1\checkmark}_{p'} \texttt{nil} \int = \\
\frac{q}{2} \cdot \sum \wr r' \cdot (\underbrace{\sum \wr p' \mid P' \xRightarrow{s_1\checkmark}_{p'} \texttt{nil} \int}_{p_1}) \mid P \xrightarrow{\texttt{A}}_{r'} P' \ \wedge \ P' \in \mathcal{P} \int = \\
\frac{1}{2}(q \cdot p_1) \cdot (\underbrace{\sum \wr r' \mid P \xrightarrow{\texttt{A}}_{r'} P' \ \wedge \ P' \in \mathcal{P} \int}_{r_1}) = \frac{1}{2} \cdot (q \cdot p_1 \cdot r_1) = \frac{1}{2}(q \cdot p)
\end{aligned} \tag{5}$$

The only transition of $P \wedge Q$ that allows to alternate $s$ and $s'$ has the form

1. $P \wedge Q \xrightarrow{\texttt{A}}_{\frac{r'}{2}} P' \wedge Q \xRightarrow{s''\checkmark}_{p'} \texttt{nil}$ as is indicated above , or
2. $P \wedge Q \xrightarrow{\texttt{A}}_{\frac{q' \cdot r'}{2}} P' \xRightarrow{s''\checkmark}_{p'} \texttt{nil}$.

So forth

$$\begin{aligned}
\sum \wr r \mid P \wedge Q \xRightarrow{s\checkmark} \texttt{nil} \int = \\
\sum \wr r \mid P \wedge Q \xrightarrow{\texttt{A}}_{\frac{r_1}{2}} P' \wedge Q \xRightarrow{s''\checkmark}_{r_2} \texttt{nil} \ \wedge \ P' \xrightarrow{\texttt{A}}_{r_1} P' \wedge \\
r = \frac{r_1}{2} \cdot r_1 \ \wedge \ s'' \in \mathsf{int}(s_1, s') \int + \\
\sum \wr r \mid P \wedge Q \xrightarrow{\texttt{A}}_{\frac{q' \cdot r'}{2}} P' \xRightarrow{s_1\checkmark}_{p'} \texttt{nil} \ \wedge \ r = p' \cdot \frac{q' \cdot r'}{2} \int = \\
\frac{1}{2}(p \cdot q) + \frac{1}{2}(p \cdot q) = p \cdot q
\end{aligned} \tag{6}$$

$|s'| > 0$ Let consider $s' = \texttt{B} \cdot s_2$. In this case we consider the alternations where $Q$ produces the feature $\texttt{B}$ in the first place, that is, first we apply the rule [**con2**]. Considering multiset $\mathcal{Q} = \wr Q' \mid Q \xrightarrow{\texttt{B}}_r Q' \xRightarrow{s_2\checkmark}_{q'} \texttt{nil} \int$, and given $r_2 = \sum \wr r \mid Q \xrightarrow{\texttt{B}}_r Q' \ \wedge \ Q' \in \mathcal{Q} \int$ and $q_1 = \sum \wr r \mid Q' \xRightarrow{s_2\checkmark}_r \texttt{nil} \ \wedge \ Q' \in \mathcal{Q} \int$, with a similar reasoning as for the equations (1) and (2) we obtain that $q = r_2 \cdot q_1$ and

$$\begin{aligned}
\sum \wr r \mid P \wedge Q \xrightarrow{\texttt{B}}_{\frac{r_1}{2}} P \wedge Q' \xRightarrow{s''\checkmark}_{r_2} \texttt{nil} \wedge \\
r = \frac{r_1}{2} \cdot r_1 \ \wedge \ s'' \in \mathsf{int}(s, s_2) \int = \frac{1}{2}(p \cdot q)
\end{aligned} \tag{7}$$

The only transition of $P \wedge Q$ that implies alternating $s$ and $s'$ has the form:

1. $P \wedge Q \xrightarrow{\mathtt{A}}_{p' \cdot \frac{r}{2}} P' \wedge Q \xRightarrow{s''\checkmark}_{p'} \mathtt{nil}$ where $s''$ is an alternation of $s_1$ and $s'$, or
2. $P \wedge Q \xrightarrow{\mathtt{B}}_{q' \cdot \frac{r}{2}} P \wedge Q' \xRightarrow{s''\checkmark}_{p'} \mathtt{nil}$ where $s''$ is an alternation of $s$ and $s_2$,

So forth,

$$\begin{aligned}
\sum \{\!| r \mid P \wedge Q \xRightarrow{s\checkmark} \mathtt{nil} |\!\} = \\
\sum \{\!| r \mid P \wedge Q \xrightarrow{\mathtt{A}}_{\frac{r_1}{2}} P' \wedge Q \xRightarrow{s''\checkmark}_{r_2} \mathtt{nil} \ \wedge \ P' \xrightarrow{\mathtt{A}}_{r_1} P' \wedge \\
r = \frac{r_1}{2} \cdot r_1 \ \wedge \ s'' \in \mathsf{int}(s_1, s') |\!\} + \\
\sum \{\!| r \mid P \wedge Q \xrightarrow{\mathtt{B}}_{\frac{r_1}{2}} P \wedge Q' \xRightarrow{s''\checkmark}_{r_2} \mathtt{nil} \ \wedge \ Q' \xrightarrow{\mathtt{B}}_{r_1} Q' \wedge \\
r = \frac{r_1}{2} \cdot r_1 \ \wedge \ s'' \in \mathsf{int}(s, s_2) |\!\} = \\
\frac{1}{2}(p \cdot q) + \frac{1}{2}(p \cdot q) = p \cdot q
\end{aligned} \tag{8}$$

□

**Lemma 8.** Let $P \in \mathtt{SPLA}^{\mathcal{P}}$, $\mathtt{A} \in \mathcal{F}$ and $P \xRightarrow{s\checkmark}_p \mathtt{nil}$.

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

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

□

**Lemma 9.** 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 $s$.

□

**Lemma 10.** 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 $\mathtt{A} \Rightarrow \mathtt{B}$ 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 $\mathtt{A} \Rightarrow \mathtt{B}$ in $P \xrightarrow{\mathtt{A}}_{p_1} P_1 \Rightarrow \mathtt{B}$. We obtain the result by applying the lemma 8.

$s = \mathtt{B}s_1$. In this case there exist $p_1, q \in (0,1)$ such that $P \xrightarrow{\mathtt{A}}_{p_1} P_1 \stackrel{s_1\checkmark}{\Longrightarrow}_q \mathtt{nil}$. 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 8.

$s = \mathtt{C}s_1$ **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 $P \xrightarrow{\mathtt{C}}_{p_1} P_1 \stackrel{s_1\checkmark}{\Longrightarrow}_q \mathtt{nil}$. 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 $s_1$.

□

**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 $P \stackrel{s\checkmark}{\Longrightarrow}_p \mathtt{nil}$ if and only if $\mathtt{A} \not\Rightarrow \mathtt{B}\ \mathtt{in}\ P \stackrel{s\checkmark}{\Longrightarrow}_p \mathtt{nil}$, $\mathtt{A} \notin s$ 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 $P \xrightarrow{\mathtt{A}}_{p_1} P_1 \stackrel{s_1\checkmark}{\Longrightarrow}_q \mathtt{nil}$. 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 8,

- $\mathtt{B} \in s_1$ if and only if $P_1 \Rightarrow \mathtt{B} \stackrel{s_1 \cdot \checkmark}{\Longrightarrow}_q \mathtt{nil}$.
- $\mathtt{B} \notin s_1$ if and only if $P_1 \Rightarrow \mathtt{B} \stackrel{s_1 \mathtt{B}\checkmark}{\Longrightarrow}_q \mathtt{nil}$.

$s = \mathtt{C}s_1$ **with** $\mathtt{C} \neq \mathtt{A}$**.** In this case there exist $p_1, q \in (0,1)$ such that $P \xrightarrow{\mathtt{C}}_{p_1} P_1 \stackrel{s_1\checkmark}{\Longrightarrow}_q \mathtt{nil}$. 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 $s_1$.

□

## B Proof of Proposition 4

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

*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

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

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 $r_2 = \sum \wr r2' \mid P_1 \stackrel{s_1'}{\Longrightarrow} Q,\ s_1 = s_1'[\mathcal{A}] \int$. Continuing the last equation:

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

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

□

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

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

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