On Two Characterizations of Feature Models

Ferruccio Damiani, Michael Lienhardt, Luca Paolini

On Two Characterizations of Feature Models

Ferruccio Damiani[1(✉)], Michael Lienhardt[2(✉)], and Luca Paolini[1]

[1] University of Turin, Turin, Italy
{ferruccio.damiani,luca.paolini}@unito.it
[2] ONERA, Palaiseau, France
michael.lienhardt@onera.fr

Abstract. Software-intensive systems can have thousands of interdependent configuration options across different subsystems. Feature models allow designers to organize the configuration space by describing configuration options using interdependent features: a feature is a name representing some functionality and each software variant is identified by a set of features. Different representations of feature models have been proposed in the literature. In this paper we focus on the propositional representation (which works well in practice) and the extensional representation (which has been recently shown well suited for theoretical investigations). We provide an algebraic and a propositional characterization of feature model operations and relations, and we formalize the connection between the two characterizations as monomorphisms from lattices of propositional feature models to lattices of extensional features models. This formalization sheds new light on the correspondence between the extensional and the propositional representations of feature models. It aims to foster the development of a formal framework for supporting practical exploitation of future theoretical developments on feature models and software product lines.

1 Introduction

Software-intensive systems can have thousands of interdependent configuration options across different subsystems. In the resulting configuration space, different software variants can be obtained by selecting among these configuration options and accordingly assembling the underlying subsystems. The interdependencies between options are dictated by corresponding interdependencies between the underlying subsysems [7].

Feature models [8] allow developers to organize the configuration space and facilitate the construction of software variants by describing configuration options using interdependent *features* [23]: a feature is a name representing some functionality, a set of features is called a *configuration*, and each configuration that fulfills the interdependencies expressed by the feature model, called a *product*, identifies a software variant. Software-intensive systems can comprise thousands of features and several subsystems [11,12,24,33]. The design, development

V. K. I. Pun et al. (Eds.): ICTAC 2020, LNCS 12545, pp. 103–122, 2020.
https://doi.org/10.1007/978-3-030-64276-1_6

and maintenance of feature models with thousands of features can be simplified by representing large feature models as sets of smaller interdependent feature models [11,29] that we call *fragments*. To this aim, several representations of feature models have been proposed in the literature (see, e.g., Batory [8] and Sect. 2.2 of Apel *et al.* [7]) and many approaches for composing feature models from fragments have been investigated [3,6,13,28,32].

In this paper we focus on the propositional representation (which works well in practice [10,27,34]) and the extensional representation (which has been recently shown well suited for theoretical investigations [25,31]). We investigate the correspondence between the formulation for these two representations of feature model operators and relations. The starting point of this investigation is a novel partial order between feature models, that we call the feature model *fragment relation*. It is induced by a notion of feature model composition that has been used to model industrial-size configuration spaces [25,31], such as the configuration space of the Gentoo source-based Linux distribution [20], that consists of many configurable packages (the March 1st 2019 version of the Gentoo distribution comprises 671617 features spread across 36197 feature models). We exploit this partial order to provide an algebraic characterization of feature model operations and relations. Then, we provide a propositional characterizations of them and formalize the connection between the two characterizations as monomorphisms from lattices of propositional feature models to lattices of extensional features models.

The remainder of this paper is organized as follows. In Sect. 2 we recollect the necessary background and introduce the feature model fragment relation. In Sect. 3 we present the algebraic characterization of feature model operations and relations, and in Sect. 4 we present the propositional characterization of the operations and relations together with a formal account of the connection between the two characterizations. We discuss related work in Sect. 5, and conclude the paper in Sect. 6 by outlining planned future work.

2 Background and Concept

We first recall the propositional and the extensional representations of feature models (in Sect. 2.1) together with the feature model composition operation (in Sect. 2.2), then we formalize the notion of feature model fragment in terms of a novel partial order relation on feature models (in Sect. 2.3).

2.1 Feature Model Representations

In this paper, we focus on the propositional and on the extensional representations of feature models (see, e.g., Batory [8] and Sect. 2.3 of Apel *et al.* [7] for a discussion about other representations).

Definition 1 (Feature model, propositional representation). *A* propositional feature model Φ *is a pair* $(\mathcal{F}, \phi)$ *where* $\mathcal{F}$ *is a set of features and* ϕ *is a propositional formula whose variables* x *are elements of* $\mathcal{F}$*:*

$$\phi = x \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \neg\phi \mid \mathsf{false} \mid \mathsf{true}$$

Its products are the set of features $p \subseteq \mathcal{F}$ such that ϕ is satisfied by assigning value true *to the variables x in p and* false *to the variables in $\mathcal{F} \setminus p$.*

Example 1 (A propositional representation of the glibc feature model). Gentoo packages can be configured by selecting features (called *use flags* in Gentoo), which may trigger dependencies or conflicts between packages. Version 2.29 of the *glibc* library, that contains the core functionalities of most Linux systems, is provided by the package sys-libs/glibc-2.29-r2 (abbreviated to glibc in the sequel). This package has many dependencies, including (as expressed in Gentoo's notation):

```
doc? ( sys−apps/texinfo )
    vanilla?( !sys−libs/timezone−data )
```

This dependency expresses that glibc requires the texinfo documentation generator (provided by any version of the sys-apps/texinfo package) whenever the feature doc is selected and if the feature vanilla is selected, then glibc conflicts with any version of the time zone database (as stated with the !sys-libs/timezone-data constraint). These *dependencies* can be expressed by a feature model $(\mathcal{F}_{\mathsf{glibc}}, \phi_{\mathsf{glibc}})$ where

$$\mathcal{F}_{\mathsf{glibc}} = \{\mathsf{glibc}, \mathsf{texinfo}, \mathsf{tzdata}, \mathsf{glibc{:}doc}, \mathsf{glibc{:}v}\}$$
$$\phi_{\mathsf{glibc}} = \mathsf{glibc} \wedge (\mathsf{glibc{:}doc} \rightarrow \mathsf{texinfo}) \wedge (\mathsf{glibc{:}v} \rightarrow (\neg\mathsf{tzdata}))$$

Here, the feature glibc represents the glibc package; texinfo represents any sys-apps/texinfo package; tzdata represents any version of the sys-libs/timezone-data package; and glibc:doc (resp. glibc:v) represents the glibc's doc (resp. vanilla) use flag.

The propositional representation of feature models works well in practice [10,27,34]. Recently, Schröter *et al.* [31] pointed out that using an extensional representation of feature models simplifies the presentation of feature model concepts.

Definition 2 (Feature model, extensional representation). *An* extensional feature model $\mathcal{M}$ *is a pair $(\mathcal{F}, \mathcal{P})$ where $\mathcal{F}$ is a set of features and $\mathcal{P} \subseteq 2^{\mathcal{F}}$ a set of products.*

Example 2 (An extensional representation of the glibc feature model). Let 2^S denote the powerset of S. The feature model of Example 1 can be given an extensional representation $\mathcal{M}_{\mathsf{glibc}} = (\mathcal{F}_{\mathsf{glibc}}, \mathcal{P}_{\mathsf{glibc}})$ where $\mathcal{F}_{\mathsf{glibc}}$ is the same as in Example 1 and

$$\begin{aligned}\mathcal{P}_{\mathsf{glibc}} = {} & \{\{\mathsf{glibc}\}, \{\mathsf{glibc}, \mathsf{texinfo}\}, \{\mathsf{glibc}, \mathsf{tzdata}\}, \{\mathsf{glibc}, \mathsf{texinfo}, \mathsf{tzdata}\}\} \cup \\ & \{\{\mathsf{glibc}, \mathsf{glibc{:}doc}, \mathsf{texinfo}\}, \{\mathsf{glibc}, \mathsf{glibc{:}doc}, \mathsf{texinfo}, \mathsf{tz\text{-}data}\}\} \cup \\ & \{\{\mathsf{glibc}, \mathsf{glibc{:}v}\}, \{\mathsf{glibc}, \mathsf{glibc{:}v}, \mathsf{texinfo}\}\} \cup \\ & \{\{\mathsf{glibc}, \mathsf{glibc{:}doc}, \mathsf{glibc{:}v}, \mathsf{texinfo}\}\}\end{aligned}$$

In the description of $\mathcal{P}_{\mathsf{glibc}}$, the first line contains products with glibc but none of its use flags are selected, so texinfo and tz-data can be freely installed; the second line contains products with the use flag doc selected in glibc, so a package of sys-apps/texinfo is always required; the third line contains products with the use flag vanilla selected in glibc, so no package of sys-libs/timezone-data is allowed; finally, the fourth line contains products with both glibc's use flags selected, so sys-apps/texinfo is mandatory and sys-libs/timezone-data forbidden.

The following definition introduces the extensional representation of the empty feature model and of the void feature models.

Definition 3 (Empty feature model and void feature models). *The* empty *feature model, denoted* $\mathcal{M}_\emptyset = (\emptyset, \{\emptyset\})$*, has no features and has just the empty product* $\emptyset$*. A* void *feature model is a feature model that has no products, i.e., of the form* $(\mathcal{F}, \emptyset)$ *for some set of features* $\mathcal{F}$*.*

2.2 Feature Model Composition

Complex software systems, like the Gentoo source-based Linux distribution [20], often consist of many interdependent configurable packages [24–26]. The configuration options of each package can be represented by a feature model. Therefore, configuring two packages in such a way that they can be installed together corresponds to finding a product in the composition of their associated feature models. As pointed out by Lienhardt *et al.* [25], in the propositional representation of feature models this composition corresponds to logical conjunction: the composition of two feature models $(\mathcal{F}_1, \phi_1)$ and $(\mathcal{F}_2, \phi_2)$ is the feature model $(\mathcal{F}_1 \cup \mathcal{F}_2, \phi_1 \wedge \phi_2)$. Lienhardt *et al.* [25] also claimed that in the extensional representation of feature models this composition corresponds to the binary operator $\bullet$ of Schröter *et al.* [31], which combines the products sets in way similar to the join operator from relational algebra [14].

Definition 4 (Feature model composition). *The composition of two feature models* $\mathcal{M}_1 = (\mathcal{F}_1, \mathcal{P}_1)$ *and* $\mathcal{M}_2 = (\mathcal{F}_2, \mathcal{P}_2)$*, denoted* $\mathcal{M}_1 \bullet \mathcal{M}_2$*, is the feature model* $(\mathcal{F}_1 \cup \mathcal{F}_2, \{p \cup q \mid p \in \mathcal{P}_1, q \in \mathcal{P}_2, p \cap \mathcal{F}_2 = q \cap \mathcal{F}_1\})$.

As proved in [16,17], the composition operator $\bullet$ is associative and commutative, with $\mathcal{M}_\emptyset$ as identity element (i.e., $\mathcal{M} \bullet \mathcal{M}_\emptyset = \mathcal{M}$). Note that $(\mathcal{F}_1, \mathcal{P}_1) \bullet (\mathcal{F}_2, \emptyset) = (\mathcal{F}_1 \cup \mathcal{F}_2, \emptyset)$.

Example 3 (Composing glibc and gnome-shell feature models). Let us consider another important package of the Gentoo distribution: *gnome-shell*, a core component of the Gnome Desktop environment. Version 3.30.2 of *gnome-shell* is provided by the package gnome-base/gnome-shell-3.30.2-r2 (abbreviated to g-shell in the sequel), and its dependencies include the following statement:

```
networkmanager?( sys−libs/timezone−data )
```

This dependency expresses that g-shell requires any version of the time zone database when the feature networkmanager (abbreviated to g-shell:nm in the sequel) is selected.

The *propositional representation* of this dependency can be captured by the feature model $(\mathcal{F}_{\text{g-shell}}, \phi_{\text{g-shell}})$, where

$$\mathcal{F}_{\text{g-shell}} = \{\text{g-shell, tzdata, g-shell:nm}\} \quad \phi_{\text{g-shell}} = \text{g-shell} \wedge (\text{g-shell:nm} \rightarrow \text{tzdata})$$

The corresponding *extensional representation* of this feature model is $\mathcal{M}_{\text{g-shell}} = (\mathcal{F}_{\text{g-shell}}, \mathcal{P}_{\text{g-shell}})$, where:

$$\mathcal{P}_{\text{g-shell}} = \{\{\text{g-shell}\}, \{\text{g-shell, tzdata}\}\} \cup \{\{\text{g-shell, tzdata, g-shell:nm}\}\}$$

Here, the first line contains products with g-shell but none of its use flags are selected: tzdata can be freely selected; and the second line is the product where g-shell:nm is also selected and tzdata becomes mandatory.

The *propositional representation* of the composition is the feature model $(\mathcal{F}_{\text{full}}, \phi_{\text{full}})$, where

$$\begin{aligned} \mathcal{F}_{\text{full}} &= \mathcal{F}_{\text{glibc}} \cup \mathcal{F}_{\text{g-shell}} = \{\text{glibc, texinfo, tzdata, g-shell, glibc:doc, glibc:v, g-shell:nm}\} \\ \phi_{\text{full}} &= \phi_{\text{glibc}} \wedge \phi_{\text{g-shell}} = (\text{glibc} \wedge ((\text{glibc:doc} \rightarrow \text{texinfo}) \wedge (\text{glibc:v} \rightarrow (\neg\text{tz-data}))) \\ &\qquad \wedge (\text{g-shell} \wedge (\text{g-shell:nm} \rightarrow \text{tzdata})) \end{aligned}$$

The *extensional representation* of the composition is the feature model $\mathcal{M}_{\text{full}} = \mathcal{M}_{\text{glibc}} \bullet \mathcal{M}_{\text{g-shell}} = (\mathcal{F}_{\text{full}}, \mathcal{P}_{\text{full}})$ where

$$\begin{aligned} \mathcal{P}_{\text{full}} = {} & \{\{\text{glibc, g-shell}\} \cup p \mid p \in 2^{\{\text{texinfo, tzdata}\}}\} \cup \\ & \{\{\text{glibc, glibc:doc, texinfo, g-shell}\} \cup p \mid p \in 2^{\{\text{tzdata}\}}\} \cup \\ & \{\{\text{glibc, glibc:v, g-shell}\} \cup p \mid p \in 2^{\{\text{texinfo}\}}\} \cup \\ & \{\{\text{glibc, g-shell, g-shell:nm, tzdata}\} \cup p \mid p \in 2^{\{\text{texinfo}\}}\} \cup \\ & \{\{\text{glibc, glibc:doc, glibc:v, texinfo, g-shell}\}\} \cup \\ & \{\{\text{glibc, glibc:doc, texinfo, g-shell, g-shell:nm, tzdata}\}\} \end{aligned}$$

Here, the first line contains the products where both glibc and g-shell are installed, but without use flags selected, so all optional package can be freely selected; the second line contains the products with the glibc's use flag doc selected, so sys-apps/texinfo becomes mandatory; the third line contains the products with the glibc's use flag vanilla selected, so sys-libs/timezone-data is forbidden; the fourth line contains the products with the g-shell's use flag networkmanager, so sys-libs/timezone-data is mandatory; the fifth line contains the product with glibc's both use flags selected and the sixth line contains the product with glibc's use flag doc and g-shell's use flag networkmanager are selected.

2.3 The Feature Model Fragment Relation

The notion of feature model composition induces the definition of the notion of feature model fragment as a binary relation between feature models.

Definition 5 (Feature model fragment relation). *A feature model $\mathcal{M}_0$ is a* fragment *of a feature model $\mathcal{M}$, denoted as $\mathcal{M}_0 \leq \mathcal{M}_1$, whenever there exists a feature model $\mathcal{M}'$ such that $\mathcal{M}_0 \bullet \mathcal{M}' = \mathcal{M}_1$.*

For instance, we have (by definition) that $\mathcal{M}_{\text{g-shell}} \leq (\mathcal{M}_{\text{g-shell}} \bullet \mathcal{M}_{\text{glibc}})$. It is worth observing that, as illustrated by the following example, some combination of features that are allowed in the members of the composition might be no longer available in the result of the composition.

Example 4 (Composing glibc and libical feature models). Consider for instance the version 3.0.8 of the library libical in Gentoo. Its feature model contains the following constraint (as expressed in Gentoo notation):

berkdb? (sys−libs/db) sys−libs/timezone−data

This dependency expresses that libical requires the db library whenever the feature berkdb is selected and requires the package sys-libs/timezone-data to be installed. These *dependencies* can be extensionally expressed by a feature model $\mathcal{M}_{\text{libical}} = (\mathcal{F}_{\text{libical}}, \mathcal{P}_{\text{libical}})$ where

$$\mathcal{F}_{\text{libical}} = \{\text{libical, berkdb, sys-libs/db, tzdata}\}$$
$$\mathcal{P}_{\text{libical}} = \{\{\text{libical, tzdata}\}, \{\text{libical, tzdata, berkdb, sys-libs/db}\}\}$$

Composing the feature model of glibc and libical gives the feature model $\mathcal{M}_c = (\mathcal{F}_c, \mathcal{P}_c)$ where $\mathcal{F}_c = \mathcal{F}_{\text{glibc}} \cup \mathcal{F}_{\text{libical}}$ and:

$$\begin{aligned}\mathcal{P}_c = \{&\{\text{glibc, libical, tzdata}\} \cup p \mid p \in 2^{\{\text{texinfo, sys-libs/db}\}}\} \cup \\ &\{\{\text{glibc, glibc:doc, texinfo, libical, tzdata}\} \cup p \mid p \in 2^{\{\text{sys-libs/db}\}}\} \cup \\ &\{\{\text{glibc, libical, berkdb, sys-libs/db, tzdata}\} \cup p \mid p \in 2^{\{\text{texinfo}\}}\} \cup \\ &\{\{\text{glibc, glibc:doc, texinfo, libical, berkdb, sys-libs/db, tzdata}\}\}\end{aligned}$$

Here, the first line contains the products where both glibc and libical are installed, but without use flags selected, so only the annex package timezone-data is mandatory; the second line contains the products with the glibc's use flag doc selected, so sys-apps/texinfo becomes mandatory; the third line contains the products with the libical's use flag berkdb, so sys-libs/db becomes mandatory; finally, the fourth line contains the product with all optional features of both glibc and libical selected.

It is easy to see from the constraint, and also from the extensional representation, that combining glibc and libical makes the feature glibc:v dead (i.e., not selectable): when composed, the feature models interact and not all combinations of products are available. Feature incompatibilities such as this are a normal occurrence in many product lines (such as the linux kernel) but have two negative properties: first, it means that some features that are stated to be optional (i.e., can be freely selected or not by the user) actually are not optional in some cases, depending on some other packages being selected or not; second, it means that some packages cannot be installed at the same time because of their dependencies: consider for instance a package that requires the feature glibc:v being selected, that package is not compatible with libical.

3 Algebraic Characterization of Feature Models

In Sect. 3.1, we recall some relevant algebraic notions. In Sect. 3.2, we show that the feature model fragment relation induces a lattice of feature models where the join operation is feature model composition. Then, in Sect. 3.3, we show that the feature model fragment relation generalizes the feature model interface relation [31] and we provide some more algebraic properties.

3.1 A Recollection of Algebraic Notions

In this section we briefly recall the notions of lattice, bounded lattice and Boolean algebra (see, e.g., Davey and Priestley [18] for a detailed presentation). An *ordered lattice* is a partially ordered set $(P, \sqsubseteq)$ such that, for every $x, y \in P$, both the least upper bound (lub) of $\{x, y\}$, denoted $\sup\{x, y\} = \min\{a \mid x, y \leq a\}$, and the greatest lower bound (glb) of $\{x, y\}$, denoted $\inf\{x, y\} = \max\{a \mid a \leq x, y\}$, are always defined.

An *algebraic lattice* is an algebraic structure $(L, \sqcup, \sqcap)$ where L is non-empty set equipped with two binary operations $\sqcup$ (called *join*) and $\sqcap$ (called *meet*) which satisfy the following.

- Associative laws: $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$, $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$.
- Commutative laws: $x \sqcup y = y \sqcup x$, $x \sqcap y = y \sqcap x$.
- Absorption laws: $x \sqcup (x \sqcap y) = x$, $x \sqcap (x \sqcup y) = x$.
- Idempotency laws: $x \sqcup x = x$, $x \sqcup x = x$.

As known, the two notions of lattice are equivalent (Theorem 2.9 and 2.10 of [18]). In particular, given an ordered lattice $(P, \sqsubseteq)$ with the operations $x \sqcup y = \sup\{x, y\}$ and $x \sqcap y = \inf\{x, y\}$, the following three statements are equivalent (Theorem 2.8 of [18]):

$$x \sqsubseteq y, \qquad x \sqcup y = y, \qquad x \sqcap y = x.$$

A *bounded lattice* is a lattice that contains two elements $\bot$ (the lattice's *bottom*) and $\top$ (the lattice's *top*) which satisfy the following law: $\bot \sqsubseteq x \sqsubseteq \top$. Let L be a bounded lattice, $y \in L$ is a *complement* of $x \in L$ if $x \sqcap y = \bot$ and $x \sqcup y = \top$. If x has a unique complement, we denote this complement by $\bar{x}$.

A *distributive lattice* is a lattice which satisfies the following distributive law: $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$. In a bounded distributive lattice the complement (whenever it exists) is unique (see [18, Section 4.13]).

A *Boolean lattice* (a.k.a. *Boolean algebra*) L is a bounded distributive lattice such that each $x \in L$ has a (necessarily unique) complement $\bar{x} \in L$.

3.2 Lattices of Feature Models

Although (to the best of our knowledge) only finite feature models are relevant in practice, in our theoretical development (in order to enable a better understanding of the relation between the extensional and the propositional representations)

we consider also feature models with infinitely many features and products. The following definition introduces a notation for three different sets of extensional feature models (see Definition 2) over a given set of features.

Definition 6 (Sets of extensional feature models over a set of features). *Let X be a set of features. We denote:*

- $\mathfrak{E}(X)$ *the set of the extensional feature models* $(\mathcal{F}, \mathcal{P})$ *such that* $\mathcal{F} \subseteq X$*;*
- $\mathfrak{E}_{\text{fin}}(X)$ *the subset of the finite elements of* $\mathfrak{E}(X)$*, i.e.,* $(\mathcal{F}, \mathcal{P})$ *such that* $\mathcal{F} \subseteq_{\text{fin}} X$*; and*
- $\mathfrak{E}_{\text{eql}}(X)$ *the subset of elements of* $\mathfrak{E}(X)$ *that have exactly the features* X*, i.e.,* $(\mathcal{F}, \mathcal{P})$ *such that* $\mathcal{F} = X$*.*

Note that, if X has infinitely many elements then $\mathfrak{E}_{\text{fin}}(X)$ has infinitely many elements too. Instead, if X is finite then $\mathfrak{E}(X)$ and $\mathfrak{E}_{\text{fin}}(X)$ coincide and have a finite number of elements.

In order to simplify the presentation, $\mathcal{P}|_Y$ is used to denote $\{p \cap Y \mid p \in \mathcal{P}\}$ where $\mathcal{P}$ is a set of products and Y is a set of features.

Lemma 1 (Two criteria for the feature model fragment relation). *Given a set X, for all $\mathcal{M}_1 = (\mathcal{F}_1, \mathcal{P}_1)$ and $\mathcal{M}_2 = (\mathcal{F}_2, \mathcal{P}_2)$ in $\mathfrak{E}(X)$, the following statements are equivalent:*

i) $\mathcal{M}_1 \leq \mathcal{M}_2$
ii) $\mathcal{M}_1 \bullet \mathcal{M}_2 = \mathcal{M}_2$
iii) $\mathcal{F}_1 \subseteq \mathcal{F}_2$ *and* $\mathcal{P}_1 \supseteq \mathcal{P}_2|_{\mathcal{F}_1}$

Proof *i)* $\Rightarrow$ *ii)*. It is straightforward to check that $\mathcal{M} \bullet \mathcal{M} = \mathcal{M}$, for all $\mathcal{M}$. Then, by definition of $\leq$ (Definition 5) there is $\mathcal{M}' \in \mathfrak{E}(X)$ such that $\mathcal{M}_2 = \mathcal{M}_1 \bullet \mathcal{M}'$. Thus,

$$\mathcal{M}_1 \bullet \mathcal{M}_2 = \mathcal{M}_1 \bullet (\mathcal{M}_1 \bullet \mathcal{M}') = (\mathcal{M}_1 \bullet \mathcal{M}_1) \bullet \mathcal{M}' = \mathcal{M}_1 \bullet \mathcal{M}' = \mathcal{M}_2.$$

ii) $\Rightarrow$ *iii)*. By definition of $\bullet$ (Definition 4), it is clear from the hypothesis that $\mathcal{F}_1 \subseteq \mathcal{F}_2$. Moreover, $\mathcal{P}_2 = \{p \cup q \mid p \in \mathcal{P}_1, q \in \mathcal{P}_2, p \cap \mathcal{F}_2 = q \cap \mathcal{F}_1\}$ immediately implies that $\mathcal{P}_2 = \{q \mid p \in \mathcal{P}_1, q \in \mathcal{P}_2, p = q \cap \mathcal{F}_1\}$, which in turn implies $\mathcal{P}_2|_{\mathcal{F}_1} \subseteq \mathcal{P}_1$.
iii) $\Rightarrow$ *i)*. By using the hypothesis, we have $(\mathcal{F}_1 \cup \mathcal{F}_2, \{p \cup q \mid p \in \mathcal{P}_1, q \in \mathcal{P}_2, p \cap \mathcal{F}_2 = q \cap \mathcal{F}_1\}) = (\mathcal{F}_2, \mathcal{P}_2)$, i.e. $\mathcal{M}_1 \bullet \mathcal{M}_2 = \mathcal{M}_2$. This implies, by definition of $\leq$, that $\mathcal{M}_1 \leq \mathcal{M}_2$. □

Lemma 2 (The operator $\bullet$ on $\mathfrak{E}_{\text{eql}}(X)$). *Given two feature models $\mathcal{M}_1 = (X, \mathcal{P}_1)$ and $\mathcal{M}_2 = (X, \mathcal{P}_2)$ in $\mathfrak{E}_{\text{eql}}(X)$, we have that: $\mathcal{M}_1 \bullet \mathcal{M}_2 = (X, \mathcal{P}_1 \cap \mathcal{P}_2)$.*

Proof According to the definition of $\bullet$ we have:
$\mathcal{M}_1 \bullet \mathcal{M}_2 = (X, \{p_1 \cup p_2 \mid p_1 \in \mathcal{P}_1, p_2 \in \mathcal{P}_2, p_1 = p_2\}) = (X, \mathcal{P}_1 \cap \mathcal{P}_2)$. □

Theorem 1 (Lattices of feature models over a set of features). *Given a set X and two feature models $\mathcal{M}_1 = (\mathcal{F}_1, \mathcal{P}_1), \mathcal{M}_2 = (\mathcal{F}_2, \mathcal{P}_2) \in \mathfrak{E}(X)$, we define: $\mathcal{M}_1 \star \mathcal{M}_2 = (\mathcal{F}_1 \cap \mathcal{F}_2,\ \mathcal{P}_1|_{\mathcal{F}_2} \cup \mathcal{P}_2|_{\mathcal{F}_1})$, and $\overline{\mathcal{M}_1} = (\mathcal{F}_1, 2^{\mathcal{F}_1} \setminus \mathcal{P}_1)$. Then:*

1. $(\mathfrak{E}(X), \leq)$ *is a bounded lattice with join* $\bullet$*, meet* $\star$*, bottom* $\mathcal{M}_\emptyset = (\emptyset, \{\emptyset\})$ *and top* $(X, \emptyset)$.
2. *If* X *is an infinite set then* $\mathfrak{E}_{\text{fin}}(X)$ *is a sublattice of* $\mathfrak{E}(X)$ *with the same bottom and no top.*
3. $\mathfrak{E}_{\text{eql}}(X)$ *is a sublattice of* $\mathfrak{E}(X)$ *and it is a Boolean lattice with bottom* $(X, 2^X)$*, same top of* $\mathfrak{E}(X)$*, and complement* $^-$.

Proof. Let first prove that $\leq$ is a partial order. Let $\mathcal{M}_1 \leq \mathcal{M}_2 \leq \mathcal{M}_3$.

- Reflexivity. In $\mathfrak{E}(X)$ and $\mathfrak{E}_{\text{fin}}(X)$ holds, because $\mathcal{M}_1 \bullet \mathcal{M}_\emptyset = \mathcal{M}_1$. In $\mathfrak{E}_{\text{eql}}(X)$ holds, because $\mathcal{M}_1 \bullet (X, 2^X) = \mathcal{M}_1$. Clearly, $\mathcal{M}_\emptyset$ belongs to $\mathfrak{E}_{\text{eql}}(X)$ only when $X = \emptyset$ (in this case, $(X, 2^X)$ is $\mathcal{M}_\emptyset$).
- Antisymmetry. Suppose additionally $\mathcal{M}_2 \leq \mathcal{M}_1$, so by hypothesis there are $\mathcal{M}, \mathcal{M}'$ such that $\mathcal{M}_2 = \mathcal{M}_1 \bullet \mathcal{M}'$ and $\mathcal{M}_1 = \mathcal{M}_2 \bullet \mathcal{M}$. Clearly,

$$\begin{aligned}\mathcal{M}_1 = \mathcal{M}_2 \bullet \mathcal{M} &= \mathcal{M}_1 \bullet \mathcal{M}' \bullet \mathcal{M}' \bullet \mathcal{M} = \mathcal{M}_1 \bullet \mathcal{M}' \bullet \mathcal{M}' \bullet \mathcal{M} \\ &= \mathcal{M}_2 \bullet \mathcal{M}' \bullet \mathcal{M} = \mathcal{M}_2 \bullet \mathcal{M} \bullet \mathcal{M}' = \mathcal{M}_1 \bullet \mathcal{M}' = \mathcal{M}_2 \,.\end{aligned}$$

  The proof is the same for $\mathfrak{E}(X)$, $\mathfrak{E}_{\text{fin}}(X)$, $\mathfrak{E}_{\text{eql}}(X)$.
- Transitivity. Let $\mathcal{M}, \mathcal{M}'$ such that $\mathcal{M}_3 = \mathcal{M}_2 \bullet \mathcal{M}$ and $\mathcal{M}_2 = \mathcal{M}_1 \bullet \mathcal{M}'$. Clearly, $\mathcal{M}_3 = \mathcal{M}_2 \bullet \mathcal{M} = (\mathcal{M}_1 \bullet \mathcal{M}') \bullet \mathcal{M} = \mathcal{M}_1 \bullet (\mathcal{M}' \bullet \mathcal{M})$ which ensures that $\mathcal{M}_1 \leq \mathcal{M}_3$. The proof is the same for $\mathfrak{E}(X)$, $\mathfrak{E}_{\text{fin}}(X)$, $\mathfrak{E}_{\text{eql}}(X)$.

Part 1: $(\mathfrak{E}(X), \leq)$ **is a lattice with** $\mathcal{M}_\emptyset$ **as bottom and** $(X, \emptyset)$ **as top.** Let $\uparrow \mathcal{M}$ be the set of upper bounds of $\mathcal{M}$ w.r.t. $\leq$, viz. $\{\mathcal{M}' \mid \mathcal{M} \leq \mathcal{M}'\}$; and, let $\downarrow \mathcal{M}$ be the set of lower bounds of $\mathcal{M}$ w.r.t. $\leq$, viz. $\{\mathcal{M}' \mid \mathcal{M}' \leq \mathcal{M}\}$.

- If $i = 1, 2$ then $\mathcal{M}_i \leq \mathcal{M}_1 \bullet \mathcal{M}_2$ by definition of $\leq$, thus $\mathcal{M}_1 \bullet \mathcal{M}_2 \in (\uparrow \mathcal{M}_1) \cap (\uparrow \mathcal{M}_2)$. Moreover, for all common upper bounds $\mathcal{M} \in (\uparrow \mathcal{M}_1) \cap (\uparrow \mathcal{M}_2)$, we have (cf. Lemma 1)

$$\mathcal{M} = \mathcal{M}_1 \bullet \mathcal{M} = \mathcal{M}_1 \bullet (\mathcal{M}_2 \bullet \mathcal{M}) = (\mathcal{M}_1 \bullet \mathcal{M}_2) \bullet \mathcal{M}$$

  And so we have that $\mathcal{M}_1 \bullet \mathcal{M}_2$ is the join $\mathcal{M}_1 \sqcup \mathcal{M}_2$.
- Let $\mathcal{M} = (\mathcal{F}, \mathcal{P}) = (\mathcal{F}_1 \cap \mathcal{F}_2, \mathcal{P}_1|_{\mathcal{F}_2} \cup \mathcal{P}_2|_{\mathcal{F}_1})$. We have $\{p \cap \mathcal{F} \mid p \in \mathcal{P}_i\} \subseteq \mathcal{P}$ and $\mathcal{F} \subseteq \mathcal{F}_i$ for $i \in \{1, 2\}$: we thus have $\mathcal{M} \in (\downarrow \mathcal{M}_1) \cap (\downarrow \mathcal{M}_2)$. Moreover, for all $(\mathcal{F}', \mathcal{P}') \in (\downarrow \mathcal{M}_1) \cap (\downarrow \mathcal{M}_2)$, it is easy to see that $\mathcal{F}' \subseteq \mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{F}$ and $\mathcal{P}|_{\mathcal{F}'} \subseteq \mathcal{P}'$ by Lemma 1. And so, again by Lemma 1, $\mathcal{M}$ is the meet.
- For all $\mathcal{M} \in \mathfrak{E}(X)$, we have $\mathcal{M} \bullet \mathcal{M}_\emptyset = \mathcal{M}$ which implies by definition that $\mathcal{M}_\emptyset \leq \mathcal{M}$. Similarily, it is easy to see that for all $\mathcal{M} \in \mathfrak{E}(X)$, we have $\mathcal{M} \bullet (X, \emptyset) = (X, \emptyset)$ which implies by definition that $\mathcal{M} \leq (X, \emptyset)$.

Part 2: $\mathfrak{E}_{\text{fin}}(X)$**, is a sublattice of** $\mathfrak{E}(X)$ **with the same bottom and no top.** It is clear that for every $\mathcal{M}_1 \in \mathfrak{E}_{\text{fin}}(X)$ and $\mathcal{M}_2 \in \mathfrak{E}(X)$ such that $\mathcal{M}_2 \leq \mathcal{M}_1$, we have that $\mathcal{M}_2 \in \mathfrak{E}_{\text{fin}}(X)$. It follows that $\mathfrak{E}_{\text{fin}}(X)$ is a sublattice of $\mathfrak{E}(X)$ with $\mathcal{M}_\emptyset$ as bottom. Moreover, if follows from the definition of $\bullet$ that if $\mathfrak{E}_{\text{fin}}(X)$ with X infinite would have a top $(\mathcal{F}, \mathcal{P})$, we would have $S \subseteq \mathcal{F}$ for all $S \subseteq_{\text{fin}} X$. This means that $\mathcal{F}$ should be equal to X, which is not possible.

Part 3: $\mathfrak{E}_{\text{eql}}(X)$ is a bounded sublattice of $\mathfrak{E}(X)$ and a Boolean lattice, with the same top and $(X, 2^X)$ as bottom. It is clear that for every $\mathcal{M}_1, \mathcal{M}_2 \in \mathfrak{E}_{\text{eql}}(X)$ we have that $\mathcal{M}_1 \bullet \mathcal{M}_2 \in \mathfrak{E}_{\text{eql}}(X)$ and $\mathcal{M}_1 \star \mathcal{M}_2 \in \mathfrak{E}_{\text{eql}}(X)$. It follows that $\mathfrak{E}_{\text{fin}}(X)$ is a sublattice of $\mathfrak{E}(X)$ with $(X, \emptyset)$ as top. Moreover, it is easy to see that $(X, 2^X) \in \mathfrak{E}_{\text{eql}}(X)$ and that for all $\mathcal{M}_1 \in \mathfrak{E}_{\text{eql}}(X)$, we have $(X, 2^X) \bullet \mathcal{M} = \mathcal{M}$. Let now prove the distributive law. Let us consider $\mathcal{M}_1 = (X, \mathcal{P}_1)$, $\mathcal{M}_2 = (X, \mathcal{P}_2)$, $\mathcal{M}_3 = (X, \mathcal{P}_3) \in \mathfrak{E}_{\text{eql}}(X)$, we have:

$$\begin{aligned}\mathcal{M}_1 \sqcap (\mathcal{M}_2 \sqcup \mathcal{M}_3) &= (X, \mathcal{P}_1 \cup (\mathcal{P}_2 \cap \mathcal{P}_3)) \\ &= (X, (\mathcal{P}_1 \cup \mathcal{P}_2) \cap (\mathcal{P}_1 \cup \mathcal{P}_3)) = (\mathcal{M}_1 \sqcap \mathcal{M}_2) \sqcup (\mathcal{M}_1 \sqcap \mathcal{M}_3)\end{aligned}$$

Finally, it is easy to see that $\mathcal{M}_1 \sqcap \overline{\mathcal{M}_1} = (X, 2^X)$ and $\mathcal{M}_1 \sqcup \overline{\mathcal{M}_1} = (X, \emptyset)$. □

3.3 On Fragments and Interfaces

Feature model slices were defined by Acher et al. [4] as a unary operator Π_Y that restricts a feature model to the set Y of features.

Definition 7 (Feature model slice operator). *Let $\mathcal{M} = (\mathcal{F}, \mathcal{P})$ be a feature model. The* slice *operator Π_Y on feature models, where Y is a set of features, is defined by: $\Pi_Y(\mathcal{M}) = (\mathcal{F} \cap Y, \mathcal{P}|_Y)$.*

More recently, Schröter et al. [31] introduced the following notion of feature model interface.

Definition 8 (Feature model interface relation). *A feature model $\mathcal{M}_1 = (\mathcal{F}_1, \mathcal{P}_1)$ is an* interface *of feature model $\mathcal{M}_2 = (\mathcal{F}_2, \mathcal{P}_2)$, denoted as $\mathcal{M}_1 \preceq \mathcal{M}_2$, whenever both $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\mathcal{P}_1 = \mathcal{P}_2|_{\mathcal{F}_1}$ hold.*

Remark 1 (On feature model interfaces and slices). As pointed out in [31], feature model slices and interfaces are closely related. Namely: $\mathcal{M}_1 \preceq \mathcal{M}_2$ holds if and only if there exists a set of features Y such that $\mathcal{M}_1 = \Pi_Y(\mathcal{M}_2)$.

Example 5 (A slice of the glibc feature model). Applying the operator $\Pi_{\{\mathsf{glibc},\ \mathsf{glibc:v}\}}$ to the feature model $\mathcal{M}_{\mathsf{glibc}}$ of Example 2 yields the feature model

$$\mathcal{F} = \{\mathsf{glibc},\ \mathsf{glibc:v}\} \qquad \mathcal{P} = \{\emptyset, \{\mathsf{glibc}\}, \{\mathsf{glibc},\ \mathsf{glibc:v}\}\},$$

which (according to Remark 1) is an interface for $\mathcal{M}_{\mathsf{glibc}}$.

The following theorem points out the relationship between the feature model interface relation (designed to abstract away a set of features from a feature model) and the feature model fragment relation (designed to support feature model decomposition).

Theorem 2 (Interfaces are fragments). *If $\mathcal{M}_1 \preceq \mathcal{M}_2$ then $\mathcal{M}_1 \leq \mathcal{M}_2$.*

Proof. Immediate by Definition 8 and Lemma 1. □

We conclude this section by providing some algebraic properties that relate the slice operator and the interface an fragment relations.

Lemma 3 (Monotonocity properties of the feature model slice operator). *For all* $\mathcal{F}, \mathcal{F}_1, \mathcal{F}_2 \subseteq X$ *and* $\mathcal{M}, \mathcal{M}_1, \mathcal{M}_2 \in \mathfrak{E}(X)$

1. *If* $\mathcal{F}_1 \subseteq \mathcal{F}_2$ *then* $\Pi_{\mathcal{F}_1}(\mathcal{M}) \preceq \Pi_{\mathcal{F}_2}(\mathcal{M})$.
2. *If* $\mathcal{F}_1 \subseteq \mathcal{F}_2$ *then* $\Pi_{\mathcal{F}_1}(\mathcal{M}) \leq \Pi_{\mathcal{F}_2}(\mathcal{M})$.
3. *If* $\mathcal{M}_1 \preceq \mathcal{M}_2$ *then* $\Pi_{\mathcal{F}}(\mathcal{M}_1) \preceq \Pi_{\mathcal{F}}(\mathcal{M}_2)$.
4. *If* $\mathcal{M}_1 \leq \mathcal{M}_2$ *then* $\Pi_{\mathcal{F}}(\mathcal{M}_1) \leq \Pi_{\mathcal{F}}(\mathcal{M}_2)$.

Proof 1. Clearly $\Pi_{\mathcal{F}_1}(\mathcal{M}) \bullet \Pi_{\mathcal{F}_2}(\mathcal{M}) = \Pi_{\mathcal{F}_2}(\mathcal{M})$. Thus the proof follows by Definition 8.
2. Immediate by Lemma 3.1 and Theorem 2.
3. By Definition 8, we have that $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\mathcal{P}_1 = \mathcal{P}_2 \,|_{\mathcal{F}_1}$. Consequently, for all $\mathcal{F} \subseteq X$, we have $(\mathcal{F}_1 \cap \mathcal{F}) \subseteq (\mathcal{F}_2 \cap \mathcal{F})$ and $\mathcal{P}_1 \,|_{\mathcal{F}} = \mathcal{P}_2 \,|_{\mathcal{F}_1} \,|_{\mathcal{F}}$. Still, $\Pi_{\mathcal{F}}(\mathcal{M}_1) \preceq \Pi_{\mathcal{F}}(\mathcal{M}_2)$ by Definition 8.
4. By Lemma 1, we have that $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\mathcal{P}_1 \supseteq \mathcal{P}_2 \,|_{\mathcal{F}_1}$. Consequently, for all $\mathcal{F} \subseteq X$, we have $(\mathcal{F}_1 \cap \mathcal{F}) \subseteq (\mathcal{F}_2 \cap \mathcal{F})$ and $\mathcal{P}_1 \,|_{\mathcal{F}} \supseteq \mathcal{P}_2 \,|_{\mathcal{F}_1} \,|_{\mathcal{F}}$. Still, $\Pi_{\mathcal{F}}(\mathcal{M}_1) \leq \Pi_{\mathcal{F}}(\mathcal{M}_2)$ by Lemma 1.

We remark that Lemma 3.3 and Theorem 2 do not imply Lemma 3.4.

Theorem 3 (Algebraic properties of the feature model slice operator). *For all* $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3 \in \mathfrak{E}(X)$ *and* $\mathcal{F}_4, \mathcal{F}_5 \subseteq X$, *we have*

$\leq$-Monotonicity. *If* $\mathcal{M}_1 \leq \mathcal{M}_2$ *and* $\mathcal{F}_4 \subseteq \mathcal{F}_5$, *then* $\Pi_{\mathcal{F}_4}(\mathcal{M}_1) \leq \Pi_{\mathcal{F}_5}(\mathcal{M}_2)$.
$\preceq$-Monotonicity. *If* $\mathcal{M}_1 \preceq \mathcal{M}_2$ *and* $\mathcal{F}_4 \subseteq \mathcal{F}_5$, *then* $\Pi_{\mathcal{F}_4}(\mathcal{M}_1) \preceq \Pi_{\mathcal{F}_5}(\mathcal{M}_2)$.
Commutativity. $\Pi_{\mathcal{F}_4}(\Pi_{\mathcal{F}_5}(\mathcal{M}_3)) = \Pi_{\mathcal{F}_5}(\Pi_{\mathcal{F}_4}(\mathcal{M}_3))$.

Proof. **$\leq$-Monotonicity.** Straightforward by Lemma 3.2 and Lemma 3.4.
$\preceq$-Monotonicity. Straightforward by Lemma 3.1 and Lemma 3.3.
Commutativity. In accordance with Definition 8, it is sufficient to observe that $\Pi_{\mathcal{F}_4}(\Pi_{\mathcal{F}_5}(\mathcal{M}_3)) = \Pi_{\mathcal{F}_4 \cup \mathcal{F}_5}(\mathcal{M}_3) = \Pi_{\mathcal{F}_5}(\Pi_{\mathcal{F}_4}(\mathcal{M}))$ holds. □

4 Propositional Characterization of Feature Models Operations and Relations

In Sect. 4.1 we introduce a mapping that associates each propositional feature model to its corresponding extensional representation (cf. Sect. 2.1). Then, in Sect. 4.2, we provide a propositional characterization for the fragment relation ($\leq$), for the composition ($\bullet$) and the meet ($\star$) operations; for the the bottom of the Boolean lattice $\mathfrak{E}_{\text{eql}}(X)$ (the feature model $\mathcal{M}_X = (X, 2^X)$), for the bottom of the bounded lattice $\mathfrak{E}(X)$ (the feature model $\mathcal{M}_\emptyset = (\emptyset, \{\emptyset\})$) and for the top of bounded the lattice $\mathfrak{E}(X)$) (the feature model $\mathcal{M}^X = (X, \emptyset)$); and for the complement operation ($\bar{}$). Finally, in Sect. 4.3, we provide a propositional characterization for the slice operator (Π_Y) and for the interface relation ($\preceq$).

4.1 Relating Extensional and Propositional Feature Models

As stated at the beginning of Sect. 3.2, in our theoretical development we consider also feature models with infinitely many features and products, where each product may have infinitely many features. The following definition introduces a notion for three different sets of propositional feature models (see Definition 1) over a set of features (cf. Definition 6).

Definition 9 (Sets of propositional feature models over a set of features). *Let X be a set of features. We denote:*

- $\mathfrak{P}(X)$ *the set of the propositional feature models* $(\mathcal{F}, \phi)$ *such that* $\mathcal{F} \subseteq X$*;*
- $\mathfrak{P}_{\text{fin}}(X)$ *the subset of the finite elements of* $\mathfrak{P}(X)$*, i.e.,* $(\mathcal{F}, \phi)$ *such that* $\mathcal{F} \subseteq_{\text{fin}} X$*; and*
- $\mathfrak{P}_{\text{eql}}(X)$ *the subset of elements of* $\mathfrak{P}(X)$ *that have exactly the features* X*, i.e.,* $(\mathcal{F}, \phi)$ *such that* $\mathcal{F} = X$*.*

We denote by $\text{ftrs}(\phi)$ the (finite) set of features occurring in a propositional formula ϕ, and as usual we say that ϕ is *ground* whenever $\text{ftrs}(\phi)$ is empty. We recall that an *interpretation* (a.k.a. truth assignment or valuation) $\mathcal{I}$ is a function which maps propositional logic variables to true or false [7,9]. As usual, $\text{dom}(\mathcal{I})$ denotes the domain of an interpretation $\mathcal{I}$ and we write $\mathcal{I} \models \phi$ to mean that the propositional formula ϕ is true under the interpretation $\mathcal{I}$ (i.e., $\text{ftrs}(\phi) \subseteq \text{dom}(\mathcal{I})$ and the ground formula obtained from ϕ by replacing each feature x occurring in ϕ by $\mathcal{I}(x)$ evaluates to true). We write $\models \phi$ to mean that ϕ is valid (i.e., it evaluates to true under all the interpretations $\mathcal{I}$ such that $\text{ftrs}(\phi) \subseteq \text{dom}(\mathcal{I})$). We write $\phi_1 \models \phi_2$ to mean that ϕ_2 is a logical consequence of ϕ_1 (i.e., for all interpretations $\mathcal{I}$ with $\text{ftrs}(\phi_1) \cup \text{ftrs}(\phi_2) \subseteq \text{dom}(\mathcal{I})$, if $\mathcal{I} \models \phi_1$ then $\mathcal{I} \models \phi_2$), and we write $\phi_1 \equiv \phi_2$ to mean that ϕ_1 and ϕ_2 are logically equivalent (i.e., they are satisfied by exactly the same interpretations with domain including $\text{ftrs}(\phi_1) \cup \text{ftrs}(\phi_2)$). We recall that: (i) $\mathcal{I}_1$ is *included* in $\mathcal{I}_2$, denoted $\mathcal{I}_1 \subseteq \mathcal{I}_2$, whenever $\text{dom}(\mathcal{I}_1) \subseteq \text{dom}(\mathcal{I}_2)$ and $\mathcal{I}_1(x) = \mathcal{I}_2(x)$, for all $x \in \text{dom}(\mathcal{I}_1)$; (ii) $\mathcal{I}_1$ and $\mathcal{I}_2$ are *compatible* whenever $\mathcal{I}_1(x) = \mathcal{I}_2(x)$, for all $x \in \text{dom}(\mathcal{I}_1) \cap \text{dom}(\mathcal{I}_2)$; and (iii) if $\mathcal{I}_1 \models \phi$ then its *restriction* $\mathcal{I}_0$ to $\text{ftrs}(\phi)$ is such that $\mathcal{I}_0 \models \phi$ and, for all interpretations $\mathcal{I}_2$ such that $\mathcal{I}_0 \subseteq \mathcal{I}_2$, it holds that $\mathcal{I}_2 \models \phi$.

The following definition gives a name to the interpretations that represent the products of the feature models with a given set of features.

Definition 10 (Interpretation representing a product). *Let* $(\mathcal{F}, \mathcal{P})$ *be an extensional feature model and* $p \in \mathcal{P}$*. The interpretation that* represents *the product* p*, denoted by* $\mathcal{I}_p^{\mathcal{F}}$*, is the interpretation with domain* $\mathcal{F}$ *such that:* $\mathcal{I}_p^{\mathcal{F}}(x) =$ true *if* $x \in p$*; and* $\mathcal{I}_p^{\mathcal{F}}(x) = \mathsf{false}$ *if* $x \in \mathcal{F} \setminus p$*.*

The following definition gives a name to the mapping that associates each propositional feature model to its corresponding extensional representation.

Definition 11 (The `ext` mapping). *Let* $(\mathcal{F}, \phi) \in \mathfrak{P}(X)$*. We denote by* $\texttt{ext}((\mathcal{F}, \phi))$ *(or* $\texttt{ext}(\mathcal{F}, \phi)$*, for short) the extensional feature model* $(\mathcal{F}, \mathcal{P}) \in \mathfrak{E}(X)$ *such that* $\mathcal{P} = \{p \mid p \subseteq \mathcal{F} \text{ and } \mathcal{I}_p^{\mathcal{F}} \models \phi\}$ *. In particular,* `ext` *maps* $\mathfrak{P}_{\text{fin}}(X)$ *to* $\mathfrak{E}_{\text{fin}}(X)$*, and maps* $\mathfrak{P}_{\text{eql}}(X)$ *to* $\mathfrak{E}_{\text{eql}}(X)$*.*

We denote by $\equiv$ the equivalence relation over feature models defined by: $(\mathcal{F}_1, \phi_1) \equiv (\mathcal{F}_2, \phi_2)$ if and only if both $\mathcal{F}_1 = \mathcal{F}_2$ and $\phi_1 \equiv \phi_2$. We write $[\mathfrak{P}(X)]$, $[\mathfrak{P}_{\text{fin}}(X)]$ and $[\mathfrak{P}_{\text{eql}}(X)]$ as short for the quotient sets $\mathfrak{P}(X)/\equiv$, $\mathfrak{P}_{\text{fin}}(X)/\equiv$ and $\mathfrak{P}_{\text{eql}}(X)/\equiv$, respectively.

Note that, if X has infinitely many elements and $(\mathcal{F}, \phi) \in \mathfrak{P}(X)$, then $\mathcal{F}$ may contain infinite many features, while the propositional formula ϕ is syntactically finite (cf. Definition 1). Moreover, $\mathfrak{P}_{\text{fin}}(X)$ has infinitely many elements (even when X is finite). It is also worth observing that, if X is finite, then $\mathfrak{P}(X)$ and $\mathfrak{P}_{\text{fin}}(X)$ coincide and the quotient set $[\mathfrak{P}_{\text{fin}}(X)]$ is finite. Moreover, for all $\Phi_1, \Phi_2 \in \mathfrak{P}(X)$, we have that: $\texttt{ext}(\Phi_1) = \texttt{ext}(\Phi_2)$ if and only if $\Phi_1 \equiv \Phi_2$.

All the finite feature models have a propositional representation, i.e., if $(\mathcal{F}, \mathcal{P}) \in \mathfrak{E}_{\text{fin}}(X)$, then there exists $(\mathcal{F}, \phi) \in \mathfrak{P}_{\text{fin}}(X)$ such that $\texttt{ext}(\mathcal{F}, \phi) = (\mathcal{F}, \mathcal{P})$. Take, for instance, the formula in disjunctive normal form $\phi = \bigvee_{p \in \mathcal{P}} \big((\wedge_{f \in p} f) \wedge (\wedge_{f \in \mathcal{F} \setminus p} \neg f)\big)$. Given $[\Phi] \in [\mathfrak{P}(X)]$, we define (with an abuse of notation) $\texttt{ext}([\Phi]) = \texttt{ext}(\Phi)$. Then, we have that $\texttt{ext}$ is an injection from $[\mathfrak{P}(X)]$ to $\mathfrak{E}(X)$, an injection from $[\mathfrak{P}_{\text{eql}}(X)]$ to $\mathfrak{E}_{\text{eql}}(X)$, and a bijection from $[\mathfrak{P}_{\text{fin}}(X)]$ to $\mathfrak{E}_{\text{fin}}(X)$.

As shown by the following example, if X has infinitely many elements, then there are feature models in $\mathfrak{E}(X) \setminus \mathfrak{E}_{\text{fin}}(X)$ that have no propositional representation.

Example 6 (Extensional feature models without a propositional representation). Consider the natural numbers as features. Then the extensional feature models $(\mathbb{N}, \{\{3\}\})$, $(\mathbb{N}, \{\{n \mid n \text{ is even}\}\})$ (which has a single product with infinitely many features) and $(\mathbb{N}, \{\{n\} \mid n \text{ is even}\})$ (which has infinitely many products with one feature each) have no propositional representation.

Remark 2 (On $\mathfrak{P}_{\text{fin}}(X)$ and $\mathfrak{P}(X)$). It is worth observing that, since $\texttt{ext}(\mathcal{F}, \phi) = \texttt{ext}(\text{ftrs}(\phi), \phi) \bullet (\mathcal{F} \setminus \text{ftrs}(\phi), 2^{\mathcal{F} \setminus \text{ftrs}(\phi)})$ and the set $\text{ftrs}(\phi)$ is finite, then any infinite propositional feature model (i.e., in $\mathfrak{P}(X) \setminus \mathfrak{P}_{\text{fin}}(X)$ with X infinite) is decomposable into a finite one (i.e., in $\mathfrak{P}_{\text{fin}}(X)$) and a "free" one (i.e., one where all the features are optional). Therefore, if X has infinitely many elements, then there are *infinitely* many elements of $\mathfrak{E}(X) \setminus \mathfrak{E}_{\text{fin}}(X)$ that do not have a propositional representation.

4.2 Propositional Characterization of the Lattices of Feature Models

The following theorem states that the feature model fragment relation $\leq$ corresponds to (the converse of) logical consequence.

Theorem 4 (Propositional characterization of the relation $\leq$). *Given $\Phi_1 = (\mathcal{F}_1, \phi_1)$ and $\Phi_2 = (\mathcal{F}_2, \phi_2)$ in $\mathfrak{P}(X)$, we write $\Phi_1 \leq \Phi_2$ to mean that both $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\phi_2 \models \phi_1$ hold. Then: $\texttt{ext}(\Phi_1) \leq \texttt{ext}(\Phi_2)$ holds if and only $\Phi_1 \leq \Phi_2$ holds.*

Proof. We have: $(\mathcal{F}_1, \mathcal{P}_1) = \texttt{ext}(\Phi_1) \leq \texttt{ext}(\Phi_2) = (\mathcal{F}_2, \mathcal{P}_2)$
iff $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\mathcal{P}_1 \supseteq \mathcal{P}_2|_{\mathcal{F}_1}$ (by Lemma 1)
iff $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\{p_1 \mid \mathcal{I}_{p_1}^{\mathcal{F}_1} \models \phi_1\} \supseteq \{p_2 \cap \mathcal{F}_1 \mid \mathcal{I}_{p_2}^{\mathcal{F}_2} \models \phi_2\}$
iff $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and, for all $p \in \mathcal{P}_2$, $\mathcal{I}_p^{\mathcal{F}_2} \models \phi_2$ implies $\mathcal{I}_p^{\mathcal{F}_2} \models \phi_1$
iff $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\phi_2 \models \phi_1$
iff $\Phi_1 \leq \Phi_2$. □

The following theorem shows that the feature model composition operator $\bullet$ corresponds to propositional conjunction (cf. Sect. 2.2).

Theorem 5 (Propositional characterization of the operator $\bullet$). *Given $\Phi_1 = (\mathcal{F}_1, \phi_1)$ and $\Phi_2 = (\mathcal{F}_2, \phi_2)$ in $\mathfrak{P}(X)$, we define: $\Phi_1 \bullet \Phi_2 = (\mathcal{F}_1 \cup \mathcal{F}_2, \phi_1 \wedge \phi_2)$. Then: $\texttt{ext}(\Phi_1) \bullet \texttt{ext}(\Phi_2) = \texttt{ext}(\Phi_1 \bullet \Phi_2)$.*

Proof. Let $\texttt{ext}(\mathcal{F}_i, \phi_i) = (\mathcal{F}_i, \mathcal{P}_i)$, for $i = 1, 2$.
$\texttt{ext}(\Phi_1) \bullet \texttt{ext}(\Phi_2) = (\mathcal{F}_3, \mathcal{P}_3)$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cup \mathcal{F}_2$ and iff $\mathcal{P}_3 = \{p_1 \cup p_2 \mid \mathcal{I}_{p_1}^{\mathcal{F}_1} \models \phi_1, \mathcal{I}_{p_2}^{\mathcal{F}_2} \models \phi_1, p_1 \cap \mathcal{F}_2 = p_2 \cap \mathcal{F}_1\}$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cup \mathcal{F}_2$ and $\mathcal{P}_3 = \{p \mid p_1 \cup p_2 \subseteq p$ and $\mathcal{I}_p^X \models \phi_1, \mathcal{I}_p^X \models \phi_2\}$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cup \mathcal{F}_2$ and $\mathcal{P}_3 = \{p \mid \mathcal{I}_p^{\mathcal{F}_3} \models \phi_1 \wedge \phi_2\}$
iff $(\mathcal{F}_3, \mathcal{P}_3) = \texttt{ext}(\Phi_1 \bullet \Phi_2)$. □

In order to provide a propositional characterization of the meet operator $\star$ (introduced in Theorem 1), we introduce an auxiliary notation expressing a propositional encoding of the existentially quantified formula $\exists x_1. \cdots \exists x_n. \phi$, where ϕ is a propositional formula. Given $Y = \{x_1, ..., x_n\}$, we define:

$$(\underset{Y}{\bigvee}\phi) = \begin{cases} \phi & \text{if } Y = \emptyset, \\ (\underset{Y-\{x\}}{\bigvee}(\phi[x := \mathsf{true}]) \vee (\phi[x := \mathsf{false}])) & \text{otherwise.} \end{cases}$$

Theorem 6 (Propositional characterization of the operator $\star$). *Given $\Phi_1 = (\mathcal{F}_1, \phi_1)$ and $\Phi_2 = (\mathcal{F}_2, \phi_2)$ in $\mathfrak{P}(X)$, we define:*

$$\Phi_1 \star \Phi_2 = \big(\mathcal{F}_1 \cap \mathcal{F}_2, (\underset{\text{ftrs}(\phi_1)\setminus\mathcal{F}_2}{\bigvee}\phi_1) \vee (\underset{\text{ftrs}(\phi_2)\setminus\mathcal{F}_1}{\bigvee}\phi_2)\big).$$

Then: $\texttt{ext}(\Phi_1) \star \texttt{ext}(\Phi_2) = \texttt{ext}(\Phi_1 \star \Phi_2)$.

Proof. Let $\texttt{ext}(\mathcal{F}_i, \phi_i) = (\mathcal{F}_i, \mathcal{P}_i)$ for $i = 1, 2$.
Since $\texttt{ext}(\mathcal{F}_1, \phi_1) \star \texttt{ext}(\mathcal{F}_2, \phi_2) = (\mathcal{F}_1 \cap \mathcal{F}_2, \mathcal{P}_1|_{\mathcal{F}_2} \cup \mathcal{P}_2|_{\mathcal{F}_1})$, we have that:
$\texttt{ext}(\Phi_1) \star \texttt{ext}(\Phi_2) = (\mathcal{F}_3, \mathcal{P}_3)$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cap \mathcal{F}_2$ and $\mathcal{P}_3 = \{p_1 \cap \mathcal{F}_2 \mid \mathcal{I}_{p_1}^{\mathcal{F}_1} \models \phi_1\} \cup \{p_2 \cap \mathcal{F}_1 \mid \mathcal{I}_{p_2}^{\mathcal{F}_2} \models \phi_2\}$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cap \mathcal{F}_2$ and $\mathcal{P}_3 = \{p_1 \cap \mathcal{F}_3 \mid \mathcal{I}_{p_1}^{\mathcal{F}_1} \models \phi_1\} \cup \{p_2 \cap \mathcal{F}_3 \mid \mathcal{I}_{p_2}^{\mathcal{F}_2} \models \phi_2\}$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cap \mathcal{F}_2$ and $\mathcal{P}_3 = \mathcal{P}_1|_{\mathcal{F}_3} \cup \mathcal{P}_2|_{\mathcal{F}_3}$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cap \mathcal{F}_2$ and, $p \in \mathcal{P}_3$ implies
either $\exists p_1$ s.t. $p = p_1 \cap \mathcal{F}_3$ and $\mathcal{I}_{p_1}^{\mathcal{F}_1} \models \phi_1$ or $\exists p_2$ s.t. $p = p_2 \cap \mathcal{F}_3$ and $\mathcal{I}_{p_2}^{\mathcal{F}_2} \models \phi_2$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cap \mathcal{F}_2$ and, $p \in \mathcal{P}_3$ implies
either $\mathcal{I}_p^{\mathcal{F}_3} \models (\underset{\text{ftrs}(\phi_1)\setminus\mathcal{F}_2}{\bigvee}\phi_1)$ or $\mathcal{I}_p^{\mathcal{F}_3} \models (\underset{\text{ftrs}(\phi_2)\setminus\mathcal{F}_1}{\bigvee}\phi_2)$
iff $\mathcal{F}_3 = \mathcal{F}_1 \cap \mathcal{F}_2$ and, $p \in \mathcal{P}_3$ implies $\mathcal{I}_p^{\mathcal{F}_3} \models (\underset{\text{ftrs}(\phi_1)\setminus\mathcal{F}_2}{\bigvee}\phi_1) \vee (\underset{\text{ftrs}(\phi_2)\setminus\mathcal{F}_1}{\bigvee}\phi_2)$
iff $(\mathcal{F}_3, \mathcal{P}_3) = \texttt{ext}(\Phi_1 \star \Phi_2)$.
□

The following theorem states that the feature models of the form $\mathcal{M}_{\mathcal{F}} = (\mathcal{F}, 2^{\mathcal{F}})$ and $\mathcal{M}^{\mathcal{F}} = (\mathcal{F}, \emptyset)$ correspond to true and false, respectively—recall that (see Theorem 1) $\mathcal{M}_{\emptyset}$ is the bottom of the lattices $(\mathfrak{E}(X), \leq)$ and $(\mathfrak{E}_{\mathrm{fin}}(X), \leq)$, while $\mathcal{M}_X$ is the bottom of the Boolean lattice $(\mathfrak{E}_{\mathrm{eql}}(X), \leq)$, and $\mathcal{M}^X$ is the top of the lattice $(\mathfrak{E}(X), \leq)$ and of the Boolean lattice $(\mathfrak{E}_{\mathrm{eql}}(X), \leq)$ and, if X is finite, of the lattice $(\mathfrak{E}_{\mathrm{fin}}(X), \leq)$.

Theorem 7 (Propositional characterization of the feature models $\mathcal{M}_{\mathcal{F}}$ and $\mathcal{M}^{\mathcal{F}}$). *Let $(\mathcal{F}, \phi) \in \mathfrak{P}(X)$.*

1. $\mathtt{ext}(\mathcal{F}, \phi) = \mathcal{M}_{\mathcal{F}} = (\mathcal{F}, 2^{\mathcal{F}})$ *if and only if* $\phi \equiv \mathsf{true}$.
2. $\mathtt{ext}(\mathcal{F}, \phi) = \mathcal{M}^{\mathcal{F}} = (\mathcal{F}, \emptyset)$ *if and only if* $\phi \equiv \mathsf{false}$.

Proof 1. Immediate, because true is satisfied by all interpretations. *2.* Immediate, because no interpretation satisfies false. □

The following theorem shows that the feature model complement operator $^-$ (introduced in Theorem 1) corresponds to logical negation.

Theorem 8 (Propositional characterization of the operator $^-$). *Given $\Phi = (\mathcal{F}, \phi)$ in $\mathfrak{P}(X)$, we define: $\overline{\Phi} = (\mathcal{F}, \neg\phi)$. Then $\overline{\mathtt{ext}(\Phi)} = \mathtt{ext}(\overline{\Phi})$.*

Proof. Straightforward. □

Lemma 4 below provides a representation of logical disjunction in terms of a novel feature model operator, that we denote by $+$. Then, Lemma 5 sheds some light on the Boolean lattice $\mathfrak{E}_{\mathrm{eql}}(X)$, by showing that on $\mathfrak{E}_{\mathrm{eql}}(X)$ the meet operator $\star$ and the operator $+$ coincide.

Lemma 4 (The operator $+$ and its propositional characterization). *Given two sets of sets Y and Z, we define: $Y \Cup Z = \{y \cup z \mid y \in Y, z \in Z\}$. Given two feature models $\mathcal{M}_1 = (\mathcal{F}_1, \mathcal{P}_1)$ and $\mathcal{M}_2 = (\mathcal{F}_2, \mathcal{P}_2)$ in $\mathfrak{E}(X)$, we define: $\mathcal{M}_1 + \mathcal{M}_2 = (\mathcal{F}_1 \cup \mathcal{F}_2, (\mathcal{P}_1 \Cup 2^{(\mathcal{F}_2 \setminus \mathcal{F}_1)}) \cup (\mathcal{P}_2 \Cup 2^{(\mathcal{F}_1 \setminus \mathcal{F}_2)}))$. Given $\Phi_1 = (\mathcal{F}_1, \phi_1)$ and $\Phi_2 = (\mathcal{F}_2, \phi_2)$ in $\mathfrak{P}(X)$, we define: $\Phi_1 + \Phi_2 = (\mathcal{F}_1 \cup \mathcal{F}_2, \phi_1 \vee \phi_2)$. Then: $\mathtt{ext}(\Phi_1) + \mathtt{ext}(\Phi_2) = \mathtt{ext}(\Phi_1 + \Phi_2)$.*

Proof. Let $\mathtt{ext}(\mathcal{F}_i, \phi_i) = (\mathcal{F}_i, \mathcal{P}_i)$ for $i = 1, 2$. We have that
$\mathtt{ext}(\Phi_1) + \mathtt{ext}(\Phi_2) = (\mathcal{F}_3, \mathcal{P}_3)$
iff $\mathcal{P}_3 = \mathcal{F}_1 \cup \mathcal{F}_2$ and $\mathcal{P}_3 = \{p_1 \Cup 2^{(\mathcal{F}_2 \setminus \mathcal{F}_1)}) \mid \mathcal{I}^{\mathcal{F}_1}_{p_1} \models \phi_1\} \cup \{p_2 \Cup 2^{(\mathcal{F}_1 \setminus \mathcal{F}_2)} \mid \mathcal{I}^{\mathcal{F}_2}_{p_2} \models \phi_2\}$
iff $\mathcal{P}_3 = \mathcal{F}_1 \cup \mathcal{F}_2$ and, $p \in \mathcal{P}_3$ implies
either $p \in \{p_1 \Cup 2^{(\mathcal{F}_2 \setminus \mathcal{F}_1)}) \mid \mathcal{I}^{\mathcal{F}_1}_{p_1} \models \phi_1\}$ or $p \in \{p_2 \Cup 2^{(\mathcal{F}_1 \setminus \mathcal{F}_2)} \mid \mathcal{I}^{\mathcal{F}_2}_{p_2} \models \phi_2\}$
iff $\mathcal{P}_3 = \mathcal{F}_1 \cup \mathcal{F}_2$ and, $p \in \mathcal{P}_3$ implies either $\mathcal{I}^{\mathcal{F}_3}_{p} \models \phi_1$ or $\mathcal{I}^{\mathcal{F}_3}_{p} \models \phi_2$
iff $\mathcal{P}_3 = \mathcal{F}_1 \cup \mathcal{F}_2$ and, $p \in \mathcal{P}_3$ implies $\mathcal{I}^{\mathcal{F}_3}_{p \cap \mathcal{F}_1} \models \phi_1 \vee \phi_2$
iff $(\mathcal{F}_3, \mathcal{P}_3) = \mathtt{ext}(\Phi_1 + \Phi_2)$. □

Lemma 5 (The operators $\star$ and $+$ on $\mathfrak{E}_{\mathrm{eql}}(X)$). *Given two feature models $\mathcal{M}_1 = (X, \mathcal{P}_1)$ and $\mathcal{M}_2 = (X, \mathcal{P}_2)$ in $\mathfrak{E}_{\mathrm{eql}}(X)$, we have that: $\mathcal{M}_1 \star \mathcal{M}_2 = \mathcal{M}_1 + \mathcal{M}_2 = (X, \mathcal{P}_1 \cup \mathcal{P}_2)$.*

Proof. Straightforward from the definitions of $\star$ and $+$. □

Given $[\Phi_1], [\Phi_2] \in [\mathfrak{P}(X)]$, we define (with an abuse of notation): $[\Phi_1] \leq [\Phi_2]$ as $\Phi_1 \leq \Phi_2$, $[\Phi_1] \bullet [\Phi_2] = [\Phi_1 \bullet \Phi_2]$, $[\Phi_1] \star [\Phi_2] = [\Phi_1 \star \Phi_2]$, $[\Phi_1] + [\Phi_2] = [\Phi_1 + \Phi_2]$, and $\overline{[\Phi_1]} = [\overline{\Phi_1}]$. Recall that a *homomorphism* is a structure-preserving map between two algebraic structures of the same type (e.g.., between two lattices), a *monomorphism* is an injective homomorphism, and an *isomorphism* is a bijective homomorphism.

Theorem 9 (ext is a lattice monomorphism). *Given a set X of features:*

1. *$([\mathfrak{P}(X)], \leq)$ is a bounded lattice with join $\bullet$, meet $\star$, bottom $[(\emptyset, \mathsf{true})]$ and top $[(X, \mathsf{false})]$. Moreover, $\mathtt{ext}$ is a bounded lattice monomorphism from $([\mathfrak{P}(X)], \leq)$ to $(\mathfrak{E}(X), \leq)$.*
2. *If X has infinitely many elements, then $[\mathfrak{P}_{\text{fin}}(X)]$ is a sublattice of $[\mathfrak{P}(X)]$ with the same bottom and no top. Moreover, $\mathtt{ext}$ is a lattice isomorphism from $[\mathfrak{P}_{\text{fin}}(X)]$ to $\mathfrak{E}_{\text{fin}}(X)$.*
3. *$[\mathfrak{P}_{\text{eql}}(X)]$ is a sublattice of $[\mathfrak{P}(X)]$ and it is a Boolean lattice with bottom $[(X, \mathsf{true})]$, same top of $[\mathfrak{P}(X)]$, complement $^-$, and where the meet behaves like $+$. Moreover, $\mathtt{ext}$ is a Boolean lattice monomorphism from $[\mathfrak{P}_{\text{eql}}(X)]$ to $\mathfrak{E}_{\text{eql}}(X)$ and it is an isomorphism whenever X is finite.*

Proof. Straightforward from Theorems 1, 4–8 and Lemmas 2, 4 and 5. □

4.3 Propositional Characterization of Slices and Interfaces

The following theorem provides a propositional characterization of the slice operator.

Theorem 10 (Propositional characterization of the operator Π_Y). *Let $\Phi = (\mathcal{F}, \phi)$ be in $\mathfrak{P}(X)$. We define: $\Pi_Y(\Phi) = (Y \cap \mathcal{F}, (\bigvee_{\text{ftrs}(\phi)\setminus Y} \phi))$. Then: $\Pi_Y(\mathtt{ext}(\Phi)) = \mathtt{ext}(\Pi_Y(\Phi))$.*

Proof. We have: $\Pi_Y(\mathtt{ext}(\Phi)) = (\mathcal{F}_0, \mathcal{P}_0)$
iff $\mathcal{F}_0 = \mathcal{F} \cap Y$ and $\mathcal{P}_0 = \{p \mid \mathcal{I}_p^{\mathcal{F}} \models \phi\}|_Y$
iff $\mathcal{F}_0 = \mathcal{F} \cap Y$ and $\mathcal{P}_0 = \{p \cap Y \mid \mathcal{I}_p^{\mathcal{F}} \models \phi\}$
iff $\mathcal{F}_0 = \mathcal{F} \cap Y$ and $\mathcal{P}_0 = \{p \cap Y \mid \mathcal{I}_p^{\mathcal{F} \cap Y} \models \phi\}$
iff $\mathcal{F}_0 = \mathcal{F} \cap Y$ and, $p_0 \in \mathcal{P}_0$ implies $\mathcal{I}_{p_0}^{\mathcal{F} \cap Y} \models (\bigvee_{\text{ftrs}(\phi)\setminus Y} \phi)$
iff $(\mathcal{F}_0, \mathcal{P}_0) = \mathtt{ext}(\Pi_Y(\Phi))$. □

The following corollary provides a propositional characterization of the interface relation $\mathcal{M}_1 \preceq \mathcal{M}_2$ which is the same as the interpretation of the slice operator $\mathcal{M}_1 = \Pi_Y(\mathcal{M}_2)$ when Y are the features of $\mathcal{M}_1$ (cf. Theorem 10 and Remark 1).

Corollary 1 (Propositional characterization of the relation $\preceq$). *Given $\Phi_1 = (\mathcal{F}_1, \phi_1)$ and $\Phi_2 = (\mathcal{F}_2, \phi_2)$ in $\mathfrak{P}(X)$, we write $\Phi_1 \preceq \Phi_2$ to mean that both $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\phi_1 \equiv (\bigvee_{\text{ftrs}(\phi_2)\setminus \mathcal{F}_1} \phi_2)$ hold. Then: $\mathtt{ext}(\Phi_1) \preceq \mathtt{ext}(\Phi_2)$ holds if and only $\Phi_1 \preceq \Phi_2$ holds.*

Proof. We have:
$(\mathcal{F}_1, \mathcal{P}_1) = \texttt{ext}(\Phi_1) \preceq \texttt{ext}(\Phi_2) = (\mathcal{F}_2, \mathcal{P}_2)$
iff $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\mathcal{P}_1 = \mathcal{P}_2|_{\mathcal{F}_1}$ (by Definition 8)
iff $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\{p_1 \mid \mathcal{I}_{p_1}^{\mathcal{F}_1} \models \phi_1\} = \{p_2 \cap \mathcal{F}_1 \mid \mathcal{I}_{p_2}^{\mathcal{F}_2} \models \phi_2\}$
iff $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and, for all $p \in \mathcal{P}_2$, both $\phi_2 \models \phi_1$ and $\phi_1 \models (\bigvee_{\text{ftrs}(\phi_2)\setminus\mathcal{F}_1} \phi_2)$
iff $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\phi_1 \equiv (\bigvee_{\text{ftrs}(\phi_2)\setminus\mathcal{F}_1} \phi_2)$
iff $\Phi_1 \preceq \Phi_2$. □

5 Related Work

Although the propositional representation of feature models is well known in the literature (see, e.g., Sect. 2.3 of Apel *et al.* [7]), we are not aware of any work that (as done in the present paper) provides a formal account of the correspondence between the algebraic and the propositional characterizations of feature model operators and relations, and encompasses the general case of feature models with infinitely many features. The investigation presented in this paper started from the feature model composition operator $\bullet$ and the induced fragment partial order relation $\leq$. In the following we briefly discuss relevant related work on feature model composition operators and on feature model relations.

Feature-model composition operators are often investigated in connection with multi software product lines, which are sets of interdependent product lines [22]. Eichelberger and Schmid [19] present an overview of textual-modeling languages which support variability-model composition (like FAMILIAR [5], VELVET [29], TVL [13], VSL [1]) and discuss their support for composition, modularity, and evolution. Acher *et al.* [6] consider different feature-model composition operators together with possible implementations and discuss advantages and drawbacks.

The feature-model fragment relation introduced in this paper generalizes the feature-model interface relation introduced by Schröter *et al.* [31], which (see Remark 1) is closely related to the feature model slice operator introduced by Acher *et al.* [4]. The work of Acher *et al.* [4] focuses on feature model decomposition. In subsequent work [2], Acher *et al.* use the slice operator in combination with a merge operator to address evolutionary changes for extracted variability models, focusing on detecting differences between feature-model versions during evolution. Analyzing fragmented feature models usually requires to compose the fragments in order to apply existing techniques [21,34]. Schröter *et al.* [31] proposed feature model interfaces to support evolution of large feature models composed by several feature models fragments. Namely, they propose to analyze a fragmented feature model where some fragments have been replaced by carefully chosen feature model interface to obtain results that hold for the original feature model and for all its evolution where the evolved version of the fragments replaced by the interfaces are still compatible with the interfaces. More recently, Lienhardt *et al.* [25] strengthen feature model interfaces to support efficient automated product discovery in fragmented feature models.

6 Conclusion and Future Work

The formalization presented in this paper sheds new light on the correspondence between the algebraic and propositional characterizations of feature model operations and relations. It aims to foster the development of a formal framework for supporting practical exploitation of future theoretical developments on feature models and software product lines. For instance, recent works [25,31] which introduced novel feature model relations by relying on the extensional representation for theory and on the propositional representation for experiments, do not show the propositional representation of the relations.

In future work we would like to extend this picture by considering other feature model representations [7,8], operators [6] and relations [25]. Moreover, we would like to investigate whether there are classes of infinite feature models without a propositional representation (see Remark 2) that have practical relevance—such feature models might admit convenient representations (e.g., by first order logic). We are also planning to extend out formalization to encompass cardinality-based feature models, which can also enable infinitely many products, without necessarily requiring infinitely many features [15]. Moreover, we want to investigate more in detail the feature model fragment relation and how it can be used to decompose large feature models in manageable parts. Recently [17], we have introduced the notion of software product signature in order to express dependencies between different product lines, and we have lifted to software product lines the notions of feature model composition and interface. In future work we would like to lift to software product lines other feature model operations and relations and to provide a formal account of the connection between different software product line implementation approaches [7,30,34]. This formalization would enable formal reasoning on multi software product lines comprising software product lines implemented according to different approaches.

Acknowledgments. We thank the anonymous reviewers for their useful comments.

References

1. Abele, A., Papadopoulos, Y., Servat, D., Törngren, M., Weber, M.: The CVM framework - a prototype tool for compositional variability management. In: Proceedings of 4th International Workshop on Variability Modelling of Software-Intensive Systems. ICB-Research Report, vol. 37, pp. 101–105. Universität Duisburg-Essen (2010)
2. Acher, M., Cleve, A., Collet, P., Merle, P., Duchien, L., Lahire, P.: Extraction and evolution of architectural variability models in plugin-based systems. Softw. Syst. Model. **13**(4), 1367–1394 (2014). https://doi.org/10.1007/s10270-013-0364-2
3. Acher, M., Collet, P., Lahire, P., France, R.: Comparing approaches to implement feature model composition. In: Kühne, T., Selic, B., Gervais, M.-P., Terrier, F. (eds.) ECMFA 2010. LNCS, vol. 6138, pp. 3–19. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13595-8_3

4. Acher, M., Collet, P., Lahire, P., France, R.B.: Slicing feature models. In: 26th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2011, pp. 424–427 (2011). https://doi.org/10.1109/ASE.2011.6100089
5. Acher, M., Collet, P., Lahire, P., France, R.B.: Familiar: a domain-specific language for large scale management of feature models. Sci. Comput. Program. **78**(6), 657–681 (2013). https://doi.org/10.1016/j.scico.2012.12.004
6. Acher, M., Combemale, B., Collet, P., Barais, O., Lahire, P., France, R.B.: Composing your compositions of variability models. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (eds.) MODELS 2013. LNCS, vol. 8107, pp. 352–369. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41533-3_22
7. Apel, S., Batory, D.S., Kästner, C., Saake, G.: Feature-Oriented Software Product Lines: Concepts and Implementation. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37521-7
8. Batory, D.: Feature models, grammars, and propositional formulas. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 7–20. Springer, Heidelberg (2005). https://doi.org/10.1007/11554844_3
9. Ben-Ari, M.: Mathematical Logic for Computer Science, 3rd edn. Springer, Heidelberg (2012). https://doi.org/10.1007/978-1-4471-4129-7
10. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: a literature review. Inf. Syst. **35**(6), 615–636 (2010). https://doi.org/10.1016/j.is.2010.01.001
11. Berger, T., et al.: A survey of variability modeling in industrial practice. In: Proceedings of 7th International Workshop on Variability Modelling of Software-Intensive Systems, pp. 7:1–7:8. ACM Press (2013)
12. Berger, T., She, S., Lotufo, R., Wąsowski, A., Czarnecki, K.: Variability modeling in the real: a perspective from the operating systems domain. In: Proceedings of 25th International Conference on Automated Software Engineering (ASE 2010), pp. 73–82. ACM Press (2010). https://doi.org/10.1145/1858996.1859010
13. Classen, A., Boucher, Q., Heymans, P.: A text-based approach to feature modelling: syntax and semantics of TVL. Sci. Comput. Program. **76**(12), 1130–1143 (2011). https://doi.org/10.1016/j.scico.2010.10.005
14. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**(6), 377–387 (1970). https://doi.org/10.1145/362384.362685
15. Czarnecki, K., Helsen, S., Eisenecker, U.: Formalizing cardinality-based feature models and their specialization. Softw. Process: Improv. Pract. **10**(1), 7–29 (2005). https://doi.org/10.1002/spip.213
16. Damiani, F., Lienhardt, M., Paolini, L.: A formal model for multi SPLs. In: Dastani, M., Sirjani, M. (eds.) FSEN 2017. LNCS, vol. 10522, pp. 67–83. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68972-2_5
17. Damiani, F., Lienhardt, M., Paolini, L.: A formal model for multi software product lines. Sci. Comput. Program. **172**, 203–231 (2019). https://doi.org/10.1016/j.scico.2018.11.005
18. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press, Cambridge (2002). https://doi.org/10.1017/CBO9780511809088
19. Eichelberger, H., Schmid, K.: A systematic analysis of textual variability modeling languages. In: Proceedings of 17th International Software Product Line Conference (SPLC 2013), pp. 12–21. ACM Press (2013). https://doi.org/10.1145/2491627.2491652
20. Foundation, G.: Gentoo Linux (2019). https://gentoo.org. Accessed 20 Aug 2019

21. Galindo, J.A., Benavides, D., Trinidad, P., Gutiérrez-Fernández, A.M., Ruiz-Cortés, A.: Automated analysis of feature models: Quo vadis? Computing **101**(5), 387–433 (2019). https://doi.org/10.1007/s00607-018-0646-1
22. Holl, G., Grünbacher, P., Rabiser, R.: A systematic review and an expert survey on capabilities supporting multi product lines. Inf. Softw. Technol. **54**(8), 828–852 (2012). https://doi.org/10.1016/j.infsof.2012.02.002
23. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. Tech. rep. CMU/SEI-90-TR-21, Carnegie Mellon Software Engineering Institute (1990)
24. Lienhardt, M., Damiani, F., Donetti, S., Paolini, L.: Multi software product lines in the wild. In: Proceedings of 12th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2018, pp. 89–96. ACM (2018). https://doi.org/10.1145/3168365.3170425
25. Lienhardt, M., Damiani, F., Johnsen, E.B., Mauro, J.: Lazy product discovery in huge configuration spaces. In: Proceedings of the 42th International Conference on Software Engineering, ICSE 2020. ACM (2020). https://doi.org/10.1145/3377811.3380372
26. Lotufo, R., She, S., Berger, T., Czarnecki, K., Wąsowski, A.: Evolution of the Linux kernel variability model. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 136–150. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15579-6_10
27. Mendonca, M., Wasowski, A., Czarnecki, K.: SAT-based analysis of feature models is easy. In: Muthig, D., McGregor, J.D. (eds.) Proceedings of the 13th International Software Product Line Conference. ACM International Conference Proceeding Series, vol. 446, pp. 231–240. ACM (2009). https://doi.org/10.5555/1753235.1753267
28. Rosenmüller, M., Siegmund, N., Kästner, C., Rahman, S.S.U.: Modeling dependent software product lines. In: Proceeding Workshop on Modularization, Composition and Generative Techniques for Product Line Engineering, pp. 13–18 (2008)
29. Rosenmüller, M., Siegmund, N., Thüm, T., Saake, G.: Multi-dimensional variability modeling. In: Proceedings of the 5th International Workshop on Variability Modelling of Software-Intensive Systems, pp. 11–20. ACM Press (2011). https://doi.org/10.1145/1944892.1944894
30. Schaefer, I., et al.: Software diversity: state of the art and perspectives. Int. J. Softw. Tools Technol. Transf. **14**(5), 477–495 (2012). https://doi.org/10.1007/s10009-012-0253-y
31. Schröter, R., Krieter, S., Thüm, T., Benduhn, F., Saake, G.: Feature-model interfaces: the highway to compositional analyses of highly-configurable systems. In: Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, pp. 667–678. ACM (2016). https://doi.org/10.1145/2884781.2884823
32. Schröter, R., Thüm, T., Siegmund, N., Saake, G.: Automated analysis of dependent feature models. In: Proceedings of 7th International Workshop on Variability Modelling of Software-Intensive Systems, pp. 9:1–9:5. ACM Press (2013). https://doi.org/10.1145/2430502.2430515
33. Tartler, R., Lohmann, D., Sincero, J., Schröder-Preikschat, W.: Feature consistency in compile-time-configurable system software: facing the Linux 10,000 feature problem. In: Proceedings of 6th European Conference on Computer systems (EuroSys 2011), pp. 47–60. ACM Press (2011). https://doi.org/10.1145/1966445.1966451
34. Thüm, T., Apel, S., Kästner, C., Schaefer, I., Saake, G.: A classification and survey of analysis strategies for software product lines. ACM Comput. Surv. **47**(1), 6:1–6:45 (2014). https://doi.org/10.1145/2580950