Towards a Feature mu-Calculus Targeting SPL Verification

Maurice H. ter Beek
ISTI–CNR, Pisa, Italy
m.terbeek@isti.cnr.it

Erik P. de Vink
TU/e, Eindhoven, The Netherlands
CWI, Amsterdam, The Netherlands
e.p.d.vink@tue.nl

Tim A. C. Willemse
TU/e, Eindhoven, The Netherlands
t.a.c.willemse@tue.nl

Abstract The modal μ-calculus μL is a well-known fixpoint logic to express and model check properties interpreted over labeled transition systems. In this paper, we propose two variants of the μ-calculus, μL_f and $\mu L'_f$, for feature transition systems. For this, we explicitly incorporate feature expressions into the logics, allowing operators to select transitions and behavior restricted to specific products and subfamilies. We provide semantics for μL_f and $\mu L'_f$ and relate the two new μ-calculi and μL to each other. Next, we focus on the analysis of SPL behavior and show how our formalism can be applied for product-based verification with μL_f as well as family-based verification with $\mu L'_f$. We illustrate by means of a toy example how properties can be model checked, exploiting an embedding of $\mu L'_f$ into the μ-calculus with data.

1 Introduction

Formal methods and analysis tools for the specification and verification of SPL models are widely studied [11, 32, 33]. Since many SPL applications concern embedded, safety-critical systems, guaranteeing their correct behavior by means of formal verification is an important subject of study. However, when the system to be analyzed is a product line, i.e. a family of systems, the number of possible products leads to an exponential blow-up: growing numbers of products on top of increasing numbers of states. Hence, enumerative product-by-product analysis methods will quickly prove infeasible for larger SPL models. Family-based verification, as opposed to product-based verification, seeks to exploit the commonalities of products that underlies a product line to tackle large SPL models [33].

In recent years, numerous variants of known behavioral models have been tailored to deal with the variability of SPL with the aim of verifying temporal properties of SPL models. These include modal transition systems (MTS) [4, 22], I/O automata [27, 28], process calculi [5, 21, 29, 30, 35] and feature transition systems (FTS) [15, 16]. In particular the latter have gained substantial popularity: they offer a compact representation of a family of product behaviors, individually modeled as labeled transition systems (LTS), in a single transition system model in which actions are guarded by feature expressions whose satisfaction (or not) indicate the presence (or absence) of these actions in product behaviors. This has resulted in dedicated SPL model checkers [6, 14, 19] as well as the application of existing model checkers like `NuSMV` [15], `mCRL2` [8] and `FMC` [3] to SPL.

In [7, 8, 10], we showed how the formal specification language `mCRL2` and its industrial-strength toolset can be exploited to model and analyze SPL. The `mCRL2` toolset supports parametrized modeling, model reduction and quality assurance techniques like model checking. For more details, the reader is referred to [20, 24] and `www.mcrl2.org`. In particular, we illustrated the use of `mCRL2`'s parametrized data language to model and select valid product configurations, in the presence of feature attributes and quantitative constraints, and to model and check the behavior of individually generated products (or of a set of products, by tweaking the selection process). Hence, the SPL model-checking analyses with `mCRL2` studied so far fall in the category of product-based analyses. While we did equip our `mCRL2`

To appear in EPTCS.

models of product families with an FTS-like semantics, to be able to perform family-based verification also the supporting logic (a variant of the first-order modal μ-calculus augmented with data) needs to be able to deal with the transitions of FTS labeled with feature expressions.

The modal μ-calculus μL, going back to [26], is used to express and model check properties interpreted over LTS, which subsumes more intuitive popular temporal logics like LTL and CTL. The model-checking approaches of [5, 14, 16, 18, 19] are based on LTL, those of [4, 6, 15, 17, 28] on CTL and those of [3, 7, 8, 10, 29, 31] on the μ-calculus. In line with the recommendations from [1] to "adopt and extend state-of-the-art analysis tools" and to "analyze feature combinations corresponding to products of the product line", in this paper we propose two variants of μL, coined μL_f and $\mu L'_f$, for FTS. For this, we explicitly incorporate feature expressions into the logics, thus allowing operators to single out transitions and behavior restricted to specific products and subfamilies. We provide semantics for μL_f and $\mu L'_f$ and relate the three logics to each other. Given that LTL and CTL are strict, partly overlapping subsets of μL, each of the feature-oriented variants introduced in this paper can express properties that the approaches based on LTL or CTL cannot (cf. [13] for examples of such properties).

In line with the extensions feature LTL (fLTL) [16] and feature CTL (fCTL) [15], we extend the standard μ-calculus to account for feature expressions that define the set of products over which a formula is to be verified. However, while fLTL and fCTL do not change the semantics of the temporal operators, but only limit or parametrize the set of products over which they are evaluated by the addition of a feature expression as quantifier or guard, we do change the semantics. In detail, we replace operators $\langle a \rangle$ and $[a]$ of μL by 'feature' operators $\langle a|\chi \rangle$ and $[a|\chi]$ for μL_f and $\langle\!\langle a|\chi \rangle\!\rangle$ and $[a|\chi]$ for $\mu L'_f$, for χ an arbitrary feature expression. Intuitively, the classical diamond operator $\langle a \rangle \varphi$ (may modality) is valid if there exists an a-transition that leads to a state satisfying φ, while the classical box operator $[a]\varphi$ (must modality) is valid if all a-transitions lead to a state where φ is valid (i.e. if no such transition exists it holds trivially).

The logic μL_f is product-oriented. Informally, a product p satisfies formula $\langle a|\chi \rangle \varphi$ with respect to an FTS F in some state, if p meets χ and an a-transition exists for p in F to a state where the formula φ holds for p. Similarly, p satisfies $[a|\chi]\varphi$ with respect to F in a state, if p meets χ and for all a-transitions for p in F the formula φ holds for p in the target state, or p does not meet χ. So, $\langle a|\chi \rangle \varphi$ *does not* hold if p does not satisfy the feature expression χ, while $[a|\chi]\varphi$ *does* hold if p does not satisfy χ.

The logic $\mu L'_f$ is family-oriented. The formula $\langle\!\langle a|\chi \rangle\!\rangle \varphi$ holds for a set of products P with respect to FTS F in a state s, if all products in P meet the feature expression χ and there exists a single a-transition, possible for all products in P, to a state where φ holds for the set P. In a way, the modality $\langle\!\langle a|\chi \rangle\!\rangle$ of $\mu L'_f$ is a global variant of the local modality $\langle a|\chi \rangle$ of μL_f. A formula $[a|\chi]\varphi$ of $\mu L'_f$ holds in a state of F for a set of products P, if for each subset P' of P for which an a-transition is possible, for all products of the subset P' the formula φ holds for P' in the target state. Note how, on the one hand, $[a|\chi]\varphi$ is fulfilled for P if no product of P meets the feature expression χ. On the other hand, φ is checked against subsets P' of P, cut out by χ and the feature expressions decorating the transitions of the FTS F.

Jumping ahead, for a product p and a formula φ of μL_f, possibly involving feature expressions in the modalities, we have a corresponding formula φ_p without feature expressions in μL and a corresponding formula φ' using $\langle\!\langle a|\chi \rangle\!\rangle$ rather than $\langle a|\chi \rangle$ in $\mu L'_f$. In this paper, we will show

$$\forall p \in P\colon\ \models_{F|p} \varphi_p \quad \overset{(i)}{\Longleftrightarrow} \quad \forall p \in P\colon\ p \models_F \varphi \quad \overset{(ii)}{\Longleftarrow} \quad P \models'_F \varphi'$$

for all sets of products P and given an FTS F. Thus, (i) φ holds for p with respect to the FTS F iff φ_p holds with respect to the LTS $F|p$, which is the projection of F on p obtained by including an a-transition in $F|p$ iff $p \in \gamma$ for a transition of F labelled with $a|\gamma$. And (ii), with respect to F, φ holds for

all products p in a family of products P if the formula φ' holds for the family P. This provides us both with a correct FTS-based semantics of μL_f with respect to a standard LTS-based semantics, due to (i), and with a possibility for family-based model checking due to (ii). However, the latter requires that we have means to actually verify $\mu L'_f$-formulas. We provide an outline for this exploiting the `mCRL2` toolset. We sketch an embedding of $\mu L'_f$ into the μ-calculus with data for a small example. We have already started to work on larger SPL models from the literature, such as the well-known minepump model on which we evaluate our approach in a companion paper [9].

The paper is organized as follows. Section 2 describes the starting point of the research described in this paper: product-based verification of product-based behavior with μL. In Section 3, we introduce the μL_f-variant of μL and prove the soundness of its semantics, after which we introduce the $\mu L'_f$-variant in Section 4 and show how it can be used for family-based verification of family-based behavior in Section 5. We then discuss our results and planned future work in Section 6.

2 Product-based behavior—product-based verification

In this section we recall the definition of an LTS and of a variant μL of Kozen's modal μ-calculus [26], and its semantics.

Definition 1. *An LTS L over the set $\mathscr{A}$, the set of actions, is a triple $L = (S, \to, s_*)$, with S a finite set, the set of states, $\to \subseteq S \times \mathscr{A} \times S$ the transition relation, and $s_* \in S$ the initial state.*

The modal μ-calculus involves modalities $\langle a \rangle$ and $[a]$, and fixpoint constructions. Its formulas are to be interpreted over LTSs. See [12] for an overview.

Definition 2. *Fix a set $\mathscr{X}$ of variables, ranged over by X. The μ-calculus μL over $\mathscr{A}$ and $\mathscr{X}$, containing formulas φ, is given by*

$$\varphi ::= \bot \mid \top \mid \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \langle a \rangle \varphi \mid [a]\varphi \mid X \mid \mu X.\varphi \mid \nu X.\varphi$$

where for $\mu X.\varphi$ and $\nu X.\varphi$ all free occurrences of X in φ are in the scope of an even number of negations.

Example 3. *Assuming $a, b, c \in \mathscr{A}$, formulas of μL include the following:*

(i) $\langle a \rangle ([b]\bot \wedge \langle c \rangle \top)$ *"it is possible to do action a after which action b is not possible but action c is"*

(ii) $\mu X.(\langle a \rangle X \vee \langle b \rangle \top)$ *"there exists a finite sequence of a-actions followed by a b-action"*

(iii) $\nu X.\big(\mu Y.[a]Y \wedge [b]X\big)$ *"on all infinite runs with actions a and b, action b occurs infinitely often"*

The syntactic restriction for a variable X that is bound by a fixpoint construct to occur in the scope of an even number of negations guarantees monotonicity, with respect to set inclusion, of functionals used for the semantic definition below. From the monotonicity it follows by the Knaster-Tarski theorem that the least fixpoint and greatest fixpoint of the functionals exist.

Definition 4. *Let an LTS L, with set of states S, be given. Define $sSet$, the set of state sets by $sSet = 2^S$, and define $sEnv$, the set of state-based environments, by $sEnv = \mathscr{X} \to sSet$. Then the semantic function $[\![\cdot]\!]_L : \mu L \to sEnv \to sSet$ is given by*

$$\begin{aligned} [\![\bot]\!]_L(\varepsilon) &= \varnothing \\ [\![\top]\!]_L(\varepsilon) &= S \\ [\![\neg\varphi]\!]_L(\varepsilon) &= S \setminus [\![\varphi]\!]_L(\varepsilon) \end{aligned}$$

$$
\begin{aligned}
[\![(\varphi \vee \psi)]\!]_L(\varepsilon) &= [\![\varphi]\!]_L(\varepsilon) \cup [\![\psi]\!]_L(\varepsilon) \\
[\![(\varphi \wedge \psi)]\!]_L(\varepsilon) &= [\![\varphi]\!]_L(\varepsilon) \cap [\![\psi]\!]_L(\varepsilon) \\
[\![\langle a \rangle \varphi]\!]_L(\varepsilon) &= \{\, s \mid \exists t\colon s \xrightarrow{a} t \wedge t \in [\![\varphi]\!]_L(\varepsilon) \,\} \\
[\![[a]\varphi]\!]_L(\varepsilon) &= \{\, s \mid \forall t\colon s \xrightarrow{a} t \Rightarrow t \in [\![\varphi]\!]_L(\varepsilon) \,\} \\
[\![X]\!]_L(\varepsilon) &= \varepsilon(X) \\
[\![\mu X.\varphi]\!]_L(\varepsilon) &= \mathit{lfp}(U \mapsto [\![\varphi]\!]_L(\varepsilon[U/X])) \\
[\![\nu X.\varphi]\!]_L(\varepsilon) &= \mathit{gfp}(U \mapsto [\![\varphi]\!]_L(\varepsilon[U/X]))
\end{aligned}
$$

By $\varepsilon[U/X]$ we denote the environment in *sEnv* which yields $\varepsilon(Y)$ for variables Y different from X and which yields the set U for the variable X. For an LTS L with initial state s_* and a closed μL-formula φ, we write $\models_L \varphi$ iff $s_* \in [\![\varphi]\!]_L(\varepsilon_0)$, where the environment $\varepsilon_0 \in \mathit{sEnv}$ is such that $\varepsilon_0(X) = \varnothing$ for all $X \in \mathscr{X}$.

There are various approaches to model checking of μL formulas on an LTS, in particular exploiting BDDs and using equation systems or parity games (see [12, 13] for more details and references). The papers [7, 8, 10] present a model-checking approach to SPL using μL for a prototypical coffee machine. Basically, the approach in these papers is product-based, in the sense that it provides a non-deterministic choice of the product space before entering product behavior captured by an LTS. Thus, although some tricks are possible, behavior is specified product-based as is the verification approach.

3 Family-based behavior—product-based verification

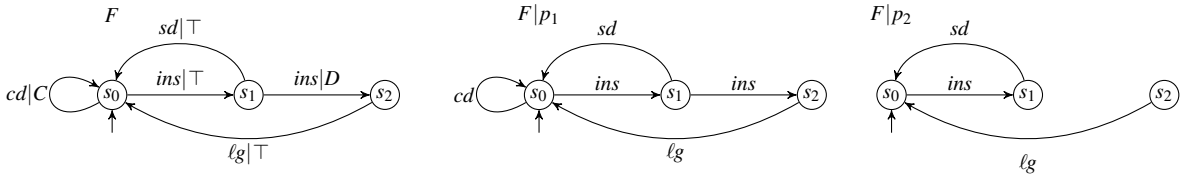
In this section we extend the logic μL to a logic μL_f incorporating feature expressions in its modalities. Formulas of μL_f are to be evaluated against an FTS as first proposed in [18]. We note that, for the ease of presentation in this paper, the definition of an FTS below is slightly more abstract.

We fix a finite non-empty set $\mathscr{F}$ of features and a set $\mathscr{A}$ of actions, and we let $\mathbb{B}[\mathscr{F}]$ denote the set of Boolean expressions over $\mathscr{F}$. We have f as a typical element of $\mathscr{F}$. Elements χ and γ of $\mathbb{B}[\mathscr{F}]$ are referred to as feature expressions. The constant $\top$ is used to denote the feature expression that is always true. A product is a set of features; $\mathscr{P}$ is the set of products, thus $\mathscr{P} \subseteq \mathbf{2}^{\mathscr{F}}$. We use p to range over $\mathscr{P}$. A product p induces an assignment α of features, viz. $\alpha_p : \mathscr{F} \to \mathbb{B}$ with $\alpha_p(f) = \mathbf{true}$ iff $f \in p$. We write $p \in \chi$ for $\alpha_p \models \chi$. We also identify a feature expression χ with the set of products that satisfy χ.

Definition 5. *An FTS F over $\mathscr{A}$ and $\mathscr{F}$ is a triple $F = (S, \theta, s_*)$, with S a finite set, the set of states, $\theta : S \times \mathscr{A} \times S \to \mathbb{B}[\mathscr{F}]$ the transition constraint function, and $s_* \in S$ the initial state.*

For states $s,t \in S$, we write $s \xrightarrow{a|\chi}_F t$ if $\theta(s,a,t) = \chi$. Given an FTS $F = (S, \theta, s_*)$ and a product $p \in \mathscr{P}$, the projection of F with respect to p is the LTS $F|p = (S, \to_{F|p}, s_*)$ over $\mathscr{A}$ with $s \xrightarrow{a}_{F|p} t$ iff $p \in \gamma$ for a transition $s \xrightarrow{a|\gamma}_F t$ of F.

Example 6. *Consider the FTS F modeling a family of coffee machines, an SPL of four products, with the set of features $\{C,D,E\}$ representing the presence of a* clean/descale *unit, a* dollar *unit and a* euro *unit, depicted below (left). The actions of F represent inserting a coin (action ins), pouring a standard or large coffee (actions sd and ℓg) and cleaning/descaling the machine (action cd). A coffee machine accepts either dollar coins or euro coins. Large cups of coffee require two coins and are only available for dollar machines. Cleaning can only occur when the machine is idle with no coins inserted. For the products $p_1 = \{C,D\}$ and $p_2 = \{E\}$, we have the projections $F|p_1$ and $F|p_2$, also depicted below (right).*



The LTS $F|p_1$ has all transitions of F, with the feature expressions decorating the arrows omitted. For $F|p_2$, the cd-loop that depends on the feature C and the second insert action ins of the transition from s_1 to s_2, which depends on the feature D, are not present. Although s_2 is unreachable in $F|p_2$ according to the definition, since $p_2 \in \top$, the transition from s_2 to s_0 is present.

We next introduce the feature μ-calculus μL_f.

Definition 7. *The feature μ-calculus μL_f over $\mathscr{A}$, $\mathscr{F}$, and $\mathscr{X}$, containing formulas φ_f, is given by*

$$\varphi_f ::= \bot \mid \top \mid \neg\varphi_f \mid \varphi_f \vee \psi_f \mid \varphi_f \wedge \psi_f \mid \langle a|\chi\rangle \varphi_f \mid [a|\chi]\varphi_f \mid X \mid \mu X.\varphi_f \mid \nu X.\varphi_f$$

where for $\mu X.\varphi_f$ and $\nu X.\varphi_f$ all free occurrences of X in φ_f are in the scope of an even number of negations.

Example 8. *Assuming feature expressions $\chi, \chi_1, \chi_2 \in \mathbb{B}[\mathscr{F}]$, the following are example formulas of μL_f:*

(i) $\langle a|\chi_1 \wedge \chi_2\rangle(\,[b|\chi_1]\bot \wedge \langle c|\chi_2\rangle\top\,)$ *"it is possible to do action a for products satisfying $\chi_1 \wedge \chi_2$ after which action b is not possible for products satisfying χ_1 but action c is for products satisfying χ_2"*

(ii) $\mu X.(\langle a|\chi_1\rangle X \vee \langle b|\chi_2\rangle\top)$ *"there exists a finite sequence of a-actions for products satisfying χ_1, followed by a b-action for products satisfying χ_2 as well"*

(iii) $\nu X.\big(\mu Y.[a|\chi]Y \wedge [b|\top]X\big)$ *"for products satisfying χ infinitely often action b can occur, after a finite number of times action a"*

The semantic function $[\![\cdot]\!]_F$ given below returns for an FTS F all state-product pairs in which a formula $\varphi_f \in \mu L_f$ holds.

Definition 9. *Let an FTS F, with set of states S, be given. Define* spSet*, the set of state-product sets, by $\mathit{spSet} = 2^{S\times\mathscr{P}}$, and define* spEnv*, the set of state-product environments, by $\mathit{spEnv} = \mathscr{X} \to \mathit{spSet}$. Then the semantic function $[\![\cdot]\!]_F : \mu L_f \to \mathit{spEnv} \to \mathit{spSet}$ is given by*

$$\begin{aligned}
[\![\bot]\!]_F(\eta) &= \varnothing \\
[\![\top]\!]_F(\eta) &= S \times \mathscr{P} \\
[\![\neg\varphi_f]\!]_F(\eta) &= (S\times\mathscr{P}) \setminus [\![\varphi_f]\!]_F(\eta) \\
[\![(\varphi_f \vee \psi_f)]\!]_F(\eta) &= [\![\varphi_f]\!]_F(\eta) \cup [\![\psi_f]\!]_F(\eta) \\
[\![(\varphi_f \wedge \psi_f)]\!]_F(\eta) &= [\![\varphi_f]\!]_F(\eta) \cap [\![\psi_f]\!]_F(\eta) \\
[\![\langle a|\chi\rangle\varphi_f]\!]_F(\eta) &= \{\,(s,p) \mid p \in \chi \wedge \\
&\qquad \big(\exists \gamma, t :\, s \xrightarrow{a|\gamma}_F t \wedge p \in \gamma \wedge (t,p) \in [\![\varphi_f]\!]_F(\eta)\big)\,\} \\
[\![[a|\chi]\varphi_f]\!]_F(\eta) &= \{\,(s,p) \mid p \in \chi \Rightarrow \\
&\qquad \big(\forall \gamma, t :\, s \xrightarrow{a|\gamma}_F t \wedge p \in \gamma \Rightarrow (t,p) \in [\![\varphi_f]\!]_F(\eta)\big)\,\}
\end{aligned}$$

$$\begin{aligned}
[\![X]\!]_F(\eta) &= \eta(X)\\
[\![\mu X.\varphi_f]\!]_F(\eta) &= lfp(V \mapsto [\![\varphi_f]\!]_F(\eta[V/X]))\\
[\![\nu X.\varphi_f]\!]_F(\eta) &= gfp(V \mapsto [\![\varphi_f]\!]_F(\eta[V/X]))
\end{aligned}$$

For an FTS F with initial state s_*, a product $p \in \mathscr{P}$, and a closed μL_f-formula φ_f, we write $p \models_F \varphi_f$ iff $(s_*, p) \in [\![\varphi_f]\!]_F(\eta_0)$, where the environment $\eta_0 \in spEnv$ is such that $\eta_0(X) = \varnothing$ for all $X \in \mathscr{X}$.

As we shall see, model checking of a μL_f-formula for an individual product reduces to model checking a μL-formula. To make this precise we introduce the translation function $sm : \mu L_f \times \mathscr{P} \to \mu L$ as follows:

$$\begin{aligned}
sm(\bot, p) &= \bot & sm(X, p) &= X\\
sm(\top, p) &= \top & sm(\mu X.\varphi_f, p) &= \mu X.sm(\varphi_f, p)\\
sm(\varphi_f \vee \psi_f, p) &= sm(\varphi_f) \vee sm(\psi_f) & sm(\nu X.\varphi_f, p) &= \nu X.sm(\varphi_f, p)\\
sm(\varphi_f \wedge \psi_f, p) &= sm(\varphi_f) \wedge sm(\psi_f)
\end{aligned}$$

$$\begin{aligned}
sm(\langle a|\chi\rangle\,\varphi_f, p) &= \textbf{if } p \in \chi \textbf{ then } \langle a\rangle\, sm(\varphi_f, p) \textbf{ else } \bot \textbf{ end}\\
sm([a|\chi]\,\varphi_f, p) &= \textbf{if } p \in \chi \textbf{ then } [a]\, sm(\varphi_f, p) \textbf{ else } \top \textbf{ end}
\end{aligned}$$

Thus, given a formula $\varphi_f \in \mu L_f$, $sm(\varphi_f)$ is the μL-formula obtained from φ_f by replacing a subformula $\langle a|\chi\rangle\,\psi_f$ by $\bot$ and a subformula $[a|\chi]\,\psi_f$ by $\top$, respectively, in case $p \notin \chi$, while omitting the feature expression χ otherwise.

The semantic function $[\![\cdot]\!]_{F|p}$ in the theorem below is an instance of Definition 4, returning for the LTS $F|p$ all states in which a formula $\varphi \in \mu L_f$ holds, given an environment.

Theorem 10. *Let a state $s \in S$ and a product $p \in \mathscr{P}$ be given. Suppose the environments $\eta \in spEnv$ and $\varepsilon \in sEnv$ are such that $(s,p) \in \eta(X) \iff s \in \varepsilon(X)$, for all $X \in \mathscr{X}$. Then it holds that*

$$(s,p) \in [\![\varphi_f]\!]_F(\eta) \quad \textit{iff} \quad s \in [\![sm(\varphi_f)]\!]_{F|p}(\varepsilon)$$

for all $\varphi_f \in \mu L_f$.

Proof. The theorem is proven by induction on the structure of φ_f. Here we exhibit two cases.

Case 1, $\langle a|\chi\rangle$: Suppose $p \in \chi$. Then $sm(\langle a|\chi\rangle\,\varphi_f, p) = \langle a\rangle\,\varphi$, with $sm(\varphi_f) = \varphi$. We have

$$\begin{aligned}
&(s,p) \in [\![\langle a|\chi\rangle\,\varphi_f]\!]_F(\eta)\\
\iff\quad & p \in \chi \wedge \big(\exists \gamma, t :\, s \xrightarrow{a|\gamma}_F t \wedge p \in \gamma \wedge (t,p) \in [\![\varphi_f]\!]_F(\eta)\big)\\
&\qquad \text{(definition } [\![\cdot]\!]_F\text{)}\\
\iff\quad & \exists t :\, s \xrightarrow{a}_{F|p} t \wedge t \in [\![\varphi]\!]_{F|p}(\varepsilon)\\
&\qquad \text{($p \in \chi$, definition } F|p\text{, and induction hypothesis)}\\
\iff\quad & s \in [\![\langle a|\chi\rangle\varphi]\!]_{F|p}(\varepsilon)\\
&\qquad \text{(definition } [\![\cdot]\!]_{F|p}\text{)}
\end{aligned}$$

Suppose $p \notin \chi$. Then $sm(\langle a|\chi\rangle\,\varphi_f, p) = \bot$. We have $(s,p) \notin [\![\langle a|\chi\rangle\,\varphi_f]\!]_F(\eta)$ by definition of $[\![\cdot]\!]_F$ and $s \notin [\![\bot]\!]_{F|p}(\varepsilon)$ by definition of $[\![\cdot]\!]_{F|p}$.

Case 2, $\mu X.\varphi_f$: Then $sm(\mu X.\varphi_f) = \mu X.\varphi$, with $sm(\varphi_f) = \varphi$. If suffices to prove

$$(s,p) \in \bigcup_{i=0}^{\infty} V_i \quad \text{iff} \quad s \in \bigcup_{i=0}^{\infty} U_i \tag{1}$$

where $V_0 = \varnothing$, $V_{i+1} = [\![\varphi_f]\!]_F(\eta[V_i/X])$ and $U_0 = \varnothing$, $U_{i+1} = [\![\varphi]\!]_{F|_p}(\varepsilon[U_i/X])$. Define η_i and ε_i, for $i \geqslant 0$, by $\eta_0 = \eta$, $\eta_{i+1} = \eta[V_i/X]$ and $\varepsilon_0 = \varepsilon$, $\varepsilon_{i+1} = \varepsilon[U_i/X]$. We claim

$$\forall Y \in \mathscr{X} : (s,p) \in \eta_i(Y) \iff s \in \varepsilon_i(Y) \quad \text{and} \quad (s,p) \in V_i \iff s \in U_i$$

Proof of the claim. Induction on i. Basis, $i = 0$: Clear, by definition of V_0 and U_0 and by the assumption on η and ε. Induction step, i+1: We first check $(s,p) \in \eta_{i+1}(Y) \iff s \in \varepsilon_{i+1}(Y)$, only for $Y = X$.

$$\begin{aligned}(s,p) \in \eta_{i+1}(X) &\iff (s,p) \in \eta[V_i/X](X) \iff (s,p) \in V_i \iff \\ &s \in U_i \text{ (by induction hypothesis for } i) \iff s \in \varepsilon[U_i/X](X) \iff s \in \varepsilon_{i+1}(X)\end{aligned}$$

Next we verify $(s,p) \in V_{i+1} \iff s \in U_{i+1}$.

$$\begin{aligned}(s,p) \in V_{i+1} &\iff (s,p) \in [\![\varphi_f]\!]_F(\eta[V_i/X]) \iff (s,p) \in [\![\varphi_f]\!]_F(\eta_{i+1}) \iff \\ &s \in [\![\varphi]\!]_{F|_p}(\varepsilon_{i+1}) \text{ (by induction hypothesis for } \varphi_f) \iff \\ &s \in [\![\varphi]\!]_{F|_p}(\varepsilon[U_i/X]) \iff s \in U_{i+1}\end{aligned}$$

From the claim Equation (1) follows directly. □

From Theorem 10 we have the following immediate consequence:

$$\models_{F|_p} sm(\varphi_f, p) \iff p \models_F \varphi_f \tag{2}$$

for all $p \in \mathscr{P}$, $\varphi_f \in \mu L_f$ closed.

Theorem 10 and its corollary show the strong connection of LTS model checking for μL and FTS model checking for μL_f. In the next section we introduce an adapted feature μ-calculus, called $\mu L_f'$, for which there is a looser relationship of FTS model checking for μL_f and FTS model checking for $\mu L_f'$ which only is sound for formulas without negation. However, because of the duality of μL_f regarding modalities and fixpoints shown in the sequel of this section, the latter is not a major restriction.

We note that although the behavior of products is specified from the family perspective by means of an FTS, model checking based on Definition 9 will have single products as unit of granularity. However, one may take advantage of compact representations when building up pieces of information during the recursive exploration of subformulas and groups of products. In particular, as in [15, 17], BDDs can be used to efficiently represent subsets of $S \times \mathscr{P}$.

Lemma 11. *It holds that*

(a) $[\![\neg[a|\chi]\varphi]\!]_F(\eta) = [\![\langle a|\chi\rangle\neg\varphi]\!]_F(\eta)$ *and* $[\![\neg\langle a|\chi\rangle\varphi]\!]_F(\eta) = [\![[a|\chi]\neg\varphi]\!]_F(\eta)$, *for all* $a \in \mathscr{A}$, $\chi \in \mathbb{B}[\mathscr{F}]$, $\varphi \in \mu L_f$, $\eta \in spEnv$, *and*

(b) $[\![\neg\mu X.\varphi]\!]_F(\eta) = [\![\nu X.\varphi[\neg X/X]]\!]_F(\eta)$ *and* $[\![\neg\nu X.\varphi]\!]_F(\eta) = [\![\mu X.\varphi[\neg X/X]]\!]_F(\eta)$, *for all* $X \in \mathscr{X}$, $\varphi \in \mu L_f$ *and* $\eta \in spEnv$.

Proof. We only present part (a), part (b) being exactly as standard. We have

$$\begin{aligned}&[\![\neg[a|\chi]\varphi]\!]_F(\eta) \\ &= (S \times \mathscr{P}) \setminus \{ (s,p) \mid p \in \chi \Rightarrow \big(\forall \gamma, t : s \xrightarrow{a|\gamma}_F t \wedge p \in \gamma \Rightarrow (t,p) \in [\![\varphi]\!]_F(\eta)\big) \} \qquad \text{(by definition } [\![\cdot]\!]_F) \\ &= \{ (s,p) \mid p \in \chi \wedge \big(\exists \gamma, t : s \xrightarrow{a|\gamma}_F t \wedge p \in \gamma \wedge (t,p) \notin [\![\varphi]\!]_F(\eta)\big) \} \\ &= [\![\langle a|\chi\rangle\neg\varphi]\!]_F(\eta) \qquad \text{(by definition } [\![\cdot]\!]_F)\end{aligned}$$

$$
\begin{aligned}
&[\![\neg\langle a|\chi\rangle\varphi]\!]_F(\eta)\\
&= (S\times\mathscr{P})\setminus\{\,(s,p)\mid p\in\chi\wedge\big(\exists\gamma,t\colon s\xrightarrow{a|\gamma}_F t\wedge p\in\gamma\Rightarrow(t,p)\in[\![\varphi]\!]_F(\eta)\big)\,\} \qquad \text{(by definition } [\![\cdot]\!]_F)\\
&= \{\,(s,p)\mid p\notin\chi\vee\big(\forall\gamma,t\colon s\xrightarrow{a|\gamma}_F t\wedge p\in\gamma\wedge(t,p)\notin[\![\varphi]\!]_F(\eta)\big)\,\}\\
&= [\![[a|\chi]\neg\varphi]\!]_F(\eta) \qquad \text{(by definition } [\![\cdot]\!]_F)
\end{aligned}
$$

□

Recall that in φ_f, all free occurrences of variables used in fixpoints are in the scope of an even number of negations. The duality of the modalities and of the fixpoint operators, together with the De Morgan laws, allow for (closed) formulas to 'push' negation inside without effecting the meaning of a formula. Therefore, in μL_f, every formula φ_f has a negation-free equivalent formula ψ_f.

4 Family-based behavior—family-based verification

In this section we consider a variation of the μ-calculus μL_f introduced in Definition 7 that allows for model checking taking sets of products as point of view. Modalities $\langle a|\chi\rangle$ of μL_f will be replaced by modalities $\langle\!\langle a|\chi\rangle\!\rangle$. The point is that the transition required in the semantics of $\langle\!\langle a|\chi\rangle\!\rangle$ in the adapted μ-calculus requires the existence of a specific transition in the underlying FTS that applies for all products under consideration at the same time.

Definition 12. *The feature μ-calculus $\mu L'_f$ over $\mathscr{A}$, $\mathscr{F}$ and $\mathscr{X}$, consisting of formulas φ', is given by*

$$\varphi' ::= \bot \mid \top \mid \neg\varphi' \mid \varphi'\vee\psi' \mid \varphi'\wedge\psi' \mid \langle\!\langle a|\chi\rangle\!\rangle\varphi' \mid [a|\chi]\varphi' \mid X \mid \mu X.\varphi' \mid \nu X.\varphi'$$

where for $\mu X.\varphi'$ and $\nu X.\varphi'$ all free occurrences of X in φ' are in the scope of an even number of negations.

The semantics of $\mu L'_f$ is given in terms of product families, i.e. sets of products. Therefore, we consider now sets of state-family pairs $\{(s_i,P_i)\mid i\in I\}$, for some index set I, i.e. pairs (s_i,P_i) of states and sets of products, rather than sets of state-product pairs $\{(s_i,p_i)\mid i\in I\}$. We put $sPSet = 2^{S\times 2^{\mathscr{P}}}$.

Definition 13. *Let an FTS F, with set of states S, be given. Define sPEnv, the set of state-family environments, by $sPEnv = \mathscr{X}\to sPSet$. Then the semantic function $[\![\cdot]\!]'_F\colon \mu L'_f\to sPEnv\to sPSet$ is given by*

$$
\begin{aligned}
[\![\bot]\!]'_F(\zeta) &= \varnothing\\
[\![\top]\!]'_F(\zeta) &= S\times 2^{\mathscr{P}}\\
[\![\neg\varphi']\!]'_F(\zeta) &= (S\times 2^{\mathscr{P}})\setminus[\![\varphi']\!]'_F(\zeta)\\
[\![(\varphi'\vee\psi')]\!]'_F(\zeta) &= [\![\varphi']\!]'_F(\zeta)\cup[\![\psi']\!]'_F(\zeta)\\
[\![(\varphi'\wedge\psi')]\!]'_F(\zeta) &= [\![\varphi']\!]'_F(\zeta)\cap[\![\psi']\!]'_F(\zeta)\\
[\![\langle\!\langle a|\chi\rangle\!\rangle\varphi']\!]'_F(\zeta) &= \{\,(s,P)\mid P\subseteq\chi\wedge\big(\exists\gamma,t\colon s\xrightarrow{a|\gamma}_F t\wedge\\
&\qquad P\subseteq\gamma\wedge(t,P\cap\chi\cap\gamma)\in[\![\varphi']\!]'_F(\zeta)\big)\,\}\\
[\![[a|\chi]\varphi']\!]'_F(\zeta) &= \{\,(s,P)\mid P\cap\chi\neq\varnothing\Rightarrow\big(\forall\gamma,t\colon s\xrightarrow{a|\gamma}_F t\wedge\\
&\qquad P\cap\chi\cap\gamma\neq\varnothing\Rightarrow(t,P\cap\chi\cap\gamma)\in[\![\varphi']\!]'_F(\zeta)\big)\,\}
\end{aligned}
$$

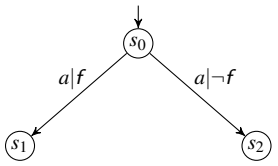
$$\begin{aligned}
[\![X]\!]'_F(\zeta) &= \zeta(X)\\
[\![\mu X.\varphi']\!]'_F(\zeta) &= lfp(W \mapsto [\![\varphi']\!]'_F(\zeta[W/X]))\\
[\![\nu X.\varphi']\!]'_F(\zeta) &= gfp(W \mapsto [\![\varphi']\!]'_F(\zeta[W/X]))
\end{aligned}$$

Well-definedness of the semantic function $[\![\cdot]\!]'_F$ can be checked as usual.

For an FTS F with initial state s_*, a set of products $P \subseteq \mathscr{P}$ and a closed $\mu L'_f$-formula φ', we write $P \models'_F \varphi'$ iff $(s_*, P) \in [\![\varphi']\!]_F(\zeta_0)$, where the environment $\zeta_0 \in sPEnv$ is such that $\zeta_0(X) = \varnothing$ for all $X \in \mathscr{X}$.

The main difference between μL_f and $\mu L'_f$ lies in their respective 'diamond' modalities, $\langle a|\chi\rangle$ for μL_f and $\langle\!\langle a|\chi\rangle\!\rangle$ for $\mu L'_f$. Here, for $\langle\!\langle a|\chi\rangle\!\rangle\varphi'$ of $\mu L'_f$ and $(s,P) \in sPSet$ to hold, we require all products in P to fulfill the feature expression χ, and we require that the witnessing transition $s \xrightarrow{a|\gamma}_F t$, guarded by the feature expression γ, is the same for all products in P.

Example 14. *Consider the FTS given on the right.*



The transition from s_0 to s_1 is possible for products having the feature f and the transition from s_0 to s_2 for products not having this feature. Consider the products $p_1 = \{f, g\}$ and $p_2 = \{g\}$. Thus, p_2 does not have feature f. Then $p_1 \models_F \langle a|\top\rangle\top$ because of the transition to s_1, and $p_2 \models_F \langle a|\top\rangle\top$ because of the transition to s_2. However, we do not *have $\{p_1, p_2\} \models'_F \langle\!\langle a|\top\rangle\!\rangle\top$, since there is no a-transition from s_0 possible for both p_1 and p_2. Note that we do not have $\{p_1, p_2\} \models'_F [a|\top]\bot$ either, showing that the modalities $\langle\!\langle a|\chi\rangle\!\rangle$ and $[a|\chi]$ of $\mu L'_f$ are not each others dual.*

Despite this example, the semantics of Definition 13 based on sets of state-family pairs is in several ways consistent with the semantics of Definition 9 based on sets of state-product pairs, as we will see next.

A formula $\varphi' \in \mu L'_f$ has a corresponding formula $\varphi_f \in \mu L_f$: the formulas φ' and φ_f are the same except that one involves a modality $\langle\!\langle a|\chi\rangle\!\rangle$ where the other uses $\langle a|\chi\rangle$. More specifically, we define a translation function $fm : \mu L'_f \to \mu L_f$ by

$$\begin{array}{ccc}
fm(\bot) = \bot & fm(\varphi' \wedge \psi') = fm(\varphi') \wedge fm(\psi') & fm(X) = X\\
fm(\top) = \top & fm(\langle\!\langle a|\chi\rangle\!\rangle\varphi') = \langle a|\chi\rangle fm(\varphi') & fm(\mu X.\varphi') = \mu X.fm(\varphi')\\
fm(\varphi' \vee \psi') = fm(\varphi') \vee fm(\psi') & fm([a|\chi]\varphi') = [a|\chi] fm(\varphi') & fm(\nu X.\varphi') = \nu X.fm(\varphi')
\end{array}$$

Moreover, for an environment $\zeta \in sPEnv$ and an environment $\eta \in spEnv$ we say that ζ relates to η if $(s,\{p\}) \in \zeta(X) \iff (s,p) \in \eta(X)$, for all variables $X \in \mathscr{X}$.

Lemma 15. *Let an FTS F, with set of states S, a state $s \in S$ and a product $p \in \mathscr{P}$, be given. Suppose the environments $\zeta \in sPEnv$ and $\eta \in spEnv$ are such that ζ relates to η. Then it holds that*

$$(s,\{p\}) \in [\![\varphi']\!]'_F(\zeta) \quad \textit{iff} \quad (s,p) \in [\![fm(\varphi')]\!]_F(\eta)$$

for all $\varphi' \in \mu L'_f$.

Proof. The proof is by induction on the structure of φ'. We only treat two cases.

Case 1, $\langle\!\langle a|\chi\rangle\!\rangle$: Recall $fm(\langle\!\langle a|\chi\rangle\!\rangle\,\varphi') = \langle a|\chi\rangle\,\varphi_f$ if $fm(\varphi') = \varphi_f$. We have

$$\begin{aligned}
&(s,p) \in [\![\, \langle\!\langle a|\chi\rangle\!\rangle\,\varphi' \,]\!]'_F(\zeta)\\
&\quad\iff\quad \{p\} \subseteq \chi \wedge \big(\exists \gamma,t\colon s \xrightarrow{a|\gamma}_F t \wedge \{p\} \subseteq \gamma \wedge (t,\{p\}\cap\chi\cap\gamma) \in [\![\, \varphi' \,]\!]'_F(\zeta)\big)\\
&\qquad\qquad \text{(by definition of } [\![\cdot]\!]'_F)\\
&\quad\iff\quad p \in \chi \wedge \big(\exists \gamma,t\colon s \xrightarrow{a|\gamma}_F t \wedge p \in \gamma \wedge (t,p) \in [\![\, \varphi_f \,]\!]_F(\eta)\big)\\
&\qquad\qquad (\{p\}\cap\chi\cap\gamma = \{p\} \text{ and by induction hypothesis})\\
&\quad\iff\quad (s,p) \in [\![\, \langle a|\chi\rangle\,\varphi_f \,]\!]_F(\eta)\\
&\qquad\qquad \text{(by definition of } [\![\cdot]\!]_F)
\end{aligned}$$

Case 2, $\mu X.\varphi'$: We have $fm(\mu X.\varphi') = \mu X.\varphi_f$ if $fm(\varphi') = \varphi_f$. Put $W_0 = \varnothing$, $W_{i+1} = [\![\, \varphi' \,]\!]'_F(\zeta[W_i/X])$ and $V_0 = \varnothing$, $V_{i+1} = [\![\, \varphi_f \,]\!]_F(\eta[V_i/X])$. We claim

$$(s,\{p\}) \in W_i \iff (s,p) \in V_i \quad \text{and} \quad \zeta[W_i/X] \text{ relates to } \eta[V_i/X] \tag{3}$$

for all $i \geqslant 0$.

Proof of the claim. Induction on i. Basis, $i = 0$: Straightforward. Induction step, i+1: Note, $\zeta[W_i/X]$ relates to $\eta[V_i/X]$ by induction hypothesis for i. Therefore we have $(s,\{p\}) \in [\![\, \varphi' \,]\!]'_F(\zeta[W_i/X]) = W_{i+1}$ iff $(s,p) \in [\![\, \varphi_f \,]\!]_F(\eta[V_i/X]) = V_{i+1}$ by induction hypothesis for φ'. Because of this it follows that $\zeta[W_{i+1}/X]$ relates to $\eta[V_{i+1}/X]$.

Now, unfolding the various definitions, we have $[\![\, \mu X.\varphi' \,]\!]'_F(\zeta) = \bigcup_{i=0}^{\infty} [\![\, \varphi' \,]\!]'_F(\zeta[W_i/X]) = \bigcup_{i=0}^{\infty} W_i$ and $[\![\, \mu X.\varphi_f \,]\!]_F(\eta) = \bigcup_{i=0}^{\infty} [\![\, \varphi_f \,]\!]_F(\eta[V_i/X]) = \bigcup_{i=0}^{\infty} V_i$. Using Equation (3) we obtain that $(s,\{p\}) \in [\![\, \mu X.\varphi' \,]\!]'_F(\zeta)$ iff $(s,p) \in [\![\, \mu X.\varphi_f \,]\!]_F(\eta)$, as was to be shown. □

Rephrasing Lemma 15 in terms of satisfaction, we have $\{p\} \models'_F \varphi'$ iff $p \models_F fm(\varphi')$ for $\varphi' \in \mu L'_f$ closed.

To relate the semantics of $\mu L'_f$ and μL_f, we use a projection function $fp : sPSet \to spSet$ from sets of state-family pairs to sets of state-product pairs, given by $fp(\{\,(s_i,P_i) \mid i \in I\,\}) = \{\,(s_i,p) \mid i \in I,\, p \in P_i\,\}$.

Theorem 16. *Let an FTS F, with state $s \in S$ and a set of products $P \subseteq \mathscr{P}$, be given. Suppose ζ and η are such that $fp(\zeta(X)) \subseteq \eta(X)$, for all $X \in \mathscr{X}$. Then it holds that*

$$(s,P) \in [\![\, \varphi' \,]\!]'_F(\zeta) \implies \forall p \in P\colon (s,p) \in [\![\, fm(\varphi') \,]\!]_F(\eta)$$

for all negation-free $\varphi' \in \mu L'_f$.

Proof. Induction on the structure of φ'. We only exhibit two cases.

Case 1, $\langle\!\langle a|\chi\rangle\!\rangle$: If $(s,P) \in [\![\, \langle\!\langle a|\chi\rangle\!\rangle\,\varphi' \,]\!]'_F(\zeta)$, then we have $P \subseteq \chi$ and γ,t exist such that $s \xrightarrow{a|\gamma}_F t$, $P \subseteq \gamma$ and $(t,P) \in [\![\, \varphi' \,]\!]'_F(\zeta)$. Thus, since $p \in P$ we have $p \in \chi$, $p \in \gamma$ and $(t,p) \in [\![\, \varphi_f \,]\!]_F(\eta)$ by induction hypothesis. Therefore, $(s,p) \in [\![\, \langle a|\chi\rangle\,\varphi \,]\!]_F(\eta)$.

Case 2, $\mu X.\varphi'$: We have $[\![\, \mu X.\varphi' \,]\!]'_F(\zeta) = \bigcup_{i=0}^{\infty} W_i$ where $W_0 = \varnothing$, $W_{i+1} = [\![\, \varphi' \,]\!]'_F(\zeta[W_i/X])$ and $[\![\, \mu X.\varphi_f \,]\!]_F(\eta) = \bigcup_{i=0}^{\infty} V_i$ where $V_0 = \varnothing$, $V_{i+1} = [\![\, \varphi_f \,]\!]_F(\eta[V_i/X])$. We claim:

$$fp(W_i) \subseteq V_i \qquad \text{for } i \geqslant 0 \tag{4}$$

Proof of the claim. Induction on i. Basis, $i = 0$: Clear. Induction step, i+1: By induction hypothesis $fp(W_i) \subseteq V_i$. Hence, $fp(\zeta[W_i/X](Y)) \subseteq \eta[V_i/X](Y)$ for all $Y \in \mathscr{X}$. Thus, by the induction hypothesis for φ', we have that $(s,P) \in [\![\, \varphi' \,]\!]'_F(\zeta[W_i/X])$ implies $(s,p) \in [\![\, \varphi_f \,]\!]_F(\eta[V_i/X])$ given $p \in P$, hence $fp([\![\, \varphi' \,]\!]'_F(\zeta[W_i/X]) \subseteq [\![\, \varphi_f \,]\!]_F(\eta[V_i/X])$. Therefore, $fp(W_{i+1}) \subseteq V_{i+1}$, as was to be shown.

Now, from claim (4), exploiting the continuity of fp, we obtain $fp(\llbracket \mu X.\varphi' \rrbracket'_F(\zeta)) = fp(\bigcup_{i=0}^{\infty} W_i) = \bigcup_{i=0}^{\infty} fp(W_i) \subseteq \bigcup_{i=0}^{\infty} fp(V_i) = fp(\llbracket \mu X.\varphi_f \rrbracket_F(\eta))$, or phrased differently $(s,P) \in \llbracket \mu X.\varphi' \rrbracket'_F(\zeta)$ implies $(s,p) \in \llbracket \mu X.\varphi_f \rrbracket_F(\eta))$ which completes the induction step for $\mu X.\varphi'$. □

In terms of satisfaction, Theorem 16 can be reformulated as

$$P \models'_F \varphi' \implies \forall p \in P\colon p \models_F fm(\varphi') \tag{5}$$

for all $P \subseteq \mathscr{P}$, $\varphi' \in \mu L'_f$ closed. Thus, given a set of products and a formula $\varphi_f \in \mu L_f$, instead of verifying φ_f for each individual product, we may seek to verify the corresponding φ' into $\mu L'_f$ of φ at once for the complete set of products. In case of an affirmative answer, Equation (5) guarantees that the formula φ_f will hold for the separate products.

For formulas in the intersection of μL_f and $\mu L'_f$ that are negation-free, i.e. formulas without any negation or modalities $\langle a|\chi\rangle$ or $\langle\!\langle a|\chi\rangle\!\rangle$, we have a stronger result.

Theorem 17. *Suppose the formula $\varphi' \in \mu L_f \cap \mu L'_f$ is negation-free, and $\zeta \in sPEnv$ and $\eta \in spEnv$ are such that $(s,P) \in \zeta(X) \iff \forall p \in P\colon (s,p) \in \eta(X)$, for all $s \in S$, $P \subseteq \mathscr{P}$ and $X \in \mathscr{X}$. Then it holds that*

$$(s,P) \in \llbracket \varphi' \rrbracket'_F(\zeta) \iff \forall p \in P\colon (s,p) \in \llbracket \varphi' \rrbracket_F(\eta)$$

for all states $s \in S$ and sets of products $P \subseteq \mathscr{P}$.

Proof. Induction on the structure of φ. We exhibit two cases only.

Case 1, $[a|\chi]$: ($\Longrightarrow$) Suppose $(s,P) \in \llbracket [a|\chi]\varphi \rrbracket'_F(\zeta)$ and pick $p \in P$. As in the proof of Theorem 16, we reason as follows: Suppose $p \in \chi$ and γ, t are such that $s \xrightarrow{a|\gamma}_F t$ and $p \in \gamma$. Then we have $P \cap \chi \neq \varnothing$ and $P \cap \chi \cap \gamma \neq \varnothing$. Hence $(t, P \cap \chi \cap \gamma) \in \llbracket \varphi' \rrbracket'_F(\zeta)$ by definition of $\llbracket [a|\chi]\varphi' \rrbracket'_F(\zeta)$. Since $p \in P \cap \chi \cap \gamma$ it follows that $(t,p) \in \llbracket \varphi \rrbracket_F(\eta)$ by induction hypothesis.

($\Longleftarrow$) Suppose $(s,p) \in \llbracket [a|\chi]\varphi \rrbracket_F(\eta)$ for all $p \in P$. Assume furthermore, $P \cap \chi \neq \varnothing$ and γ, t are such that $s \xrightarrow{a|\gamma}_F t$ and $P \cap \chi \cap \gamma \neq \varnothing$. Clearly, for all $p \in P \cap \chi \cap \gamma$ we have $p \in \chi$ and $p \in \gamma$. Thus, by definition of $\llbracket [a|\chi]\varphi \rrbracket_F(\eta)$, we have $(t,p) \in \llbracket \varphi \rrbracket_F(\eta)$ for all $p \in P \cap \chi \cap \gamma$. By induction hypothesis, $(t, P \cap \chi \cap \gamma) \in \llbracket \varphi \rrbracket'_F(\zeta)$. It follows that $(s,P) \in \llbracket [a|\chi]\varphi \rrbracket'_F(\zeta)$ by definition of $\llbracket [a|\chi]\varphi \rrbracket'_F(\zeta)$.

Case 2, $\mu X.\varphi$: We have that $\llbracket \mu X.\varphi \rrbracket'_F(\zeta) = \bigcup_{i=0}^{\infty} Z_i$ where $Z_0 = \varnothing$, $Z_{i+1} = \llbracket \varphi \rrbracket'_F(\zeta[Z_i/X])$ and $\llbracket \mu X.\varphi \rrbracket_F(\eta) = \bigcup_{i=0}^{\infty} V_i$ where $V_0 = \varnothing$, $V_{i+1} = \llbracket \varphi \rrbracket_F(\eta[V_i/X])$. We claim:

$$(s,P) \in Z_i \iff \forall p \in P\colon (s,p) \in V_i \qquad \text{for } i \geqslant 0 \tag{6}$$

Proof of the claim. Induction on i. Basis, $i = 0$: Trivial. Induction step, $i{+}1$: By induction hypothesis and the assumption for ζ and η we have, for any $s \in S$, any $P \subseteq \mathscr{P}$ and $Y \in \mathscr{X}$, that $(s,P) \in \zeta[Z_i/X](Y)$ iff $(s,p) \in \eta[V_i/X](Y)$ for all $p \in P$. Thus, $(s,P) \in Z_{i+1}$ iff $(s,P) \in \llbracket \varphi \rrbracket'_F(\zeta[Z_i/X])$ iff $\forall p \in P\colon (s,p) \in \llbracket \varphi \rrbracket_F(\eta[V_i/X])$, by induction hypothesis for φ, iff $\forall p \in P\colon (s,p) \in V_{i+1}$.

Now, using claim (6), we derive, for $s \in S$ and $P \subseteq \mathscr{P}$, $(s,P) \in \bigcup_{i=0}^{\infty} Z_i$ iff $\exists i \geqslant 0\colon (s,P) \in Z_i$ iff $\exists i \geqslant 0\colon \forall p \in P\colon (s,p) \in V_i$ iff $\forall p \in P\colon (s,p) \in \bigcup_{i=0}^{\infty} V_i$. For the latter equivalence we use that $(V_i)_{i=0}^{\infty}$ is an ascending chain, i.e. $V_i \subseteq V_{i+1}$ for all i, and that the set P is finite. From this it follows that $(s,P) \in \llbracket \mu X.\varphi \rrbracket'_F(\zeta)$ iff $\forall p \in P\colon (s,p) \in \llbracket \mu X.\varphi \rrbracket_F(\eta)$, for all $s \in S$ and $P \subseteq \mathscr{P}$. □

Following the above theorem, the strengthening of Equation (5) for closed, negation-free as well as $\langle a|\chi\rangle$-free and $\langle\!\langle a|\chi\rangle\!\rangle$-free feature μ-formulas reads

$$P \models'_F \varphi' \iff \forall p \in P\colon p \models_F \varphi' \tag{7}$$

for all $P \subseteq \mathscr{P}$, $\varphi' \in \mu L'_f \cap \mu L_f$ closed and negation-free.

5 Family-based model checking of $\mu L_f'$

In view of Theorem 16 and Equation (5), as presented in the previous section, we may divert to verifying $P \models_F' \varphi'$ family-wise when we aim to check a property φ expressed in the feature μ-calculus μL_f for a set of products P. The $\mu L_f'$-formula φ' is obtained from the μL_f-formula φ just by replacing modalities $\langle a|\chi\rangle$ by modalities $\langle\!\langle a|\chi\rangle\!\rangle$. However, the granularity for model checking for $\mu L_f'$ is that of sets of products rather than individual products as for μL_f.

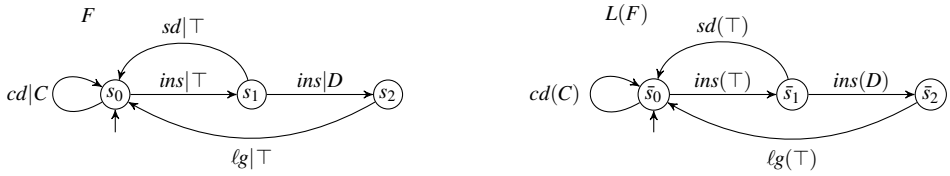
In principle, a dedicated model checker for $\mu L_f'$ can be built starting from Definition 13 following a recursion scheme. However, in such a scenario specific optimization techniques and performance enhancing facilities need to be constructed from scratch. Here, we sketch an alternative approach. The problem of deciding $P \models_F' \varphi'$ can be formalized using multi-sorted first-order modal μ-calculus, hereafter referred to as μL_{FO}, as proposed by [23, 25]. Such a calculus is given in the context of a data signature $\Sigma = (S, O)$, where S is a set of sorts and O is a set of operations, and of a set of sorted actions $\mathscr{A}$. Formulas φ and sorted actions $a(\mathrm{v})$ of (a fragment of) μL_{FO} are given by

$$\begin{aligned}\varphi ::= {} & b \mid \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists \mathrm{v{:}D}.\varphi \mid \forall \mathrm{v{:}D}.\varphi \mid \\ & \langle a(\mathrm{v})\rangle\varphi \mid [a(\mathrm{v})]\varphi \mid X(t) \mid \mu X(\mathrm{v{:}D} = t).\varphi \mid \nu X(\mathrm{v{:}D} = t).\varphi\end{aligned}$$

where for $\mu X(\mathrm{v{:}D} = t).\varphi$ and $\nu X(\mathrm{v{:}D} = t).\varphi$ all free occurrences of X in φ are in the scope of an even number of negations, b is an expression of Boolean sort, t is an arbitrary expression and v is a variable of sort D.

Expressions over μL_{FO} are to be interpreted over specific LTS where actions carry parameters. The expressions given by α in the above grammar allow for reasoning about sets of such parametrized actions. In our approach to SPL, we use these parameters to model sets of products. Moreover, we use the parameter v of the fixpoint operators to keep track of the set of products for which we are evaluating the fixpoint formula. Note, this set is dynamic: whenever we encounter a modality such as $[a|\chi]\varphi$ or $\langle\!\langle a|\chi\rangle\!\rangle\varphi$, the set of products for which we need to evaluate φ is restricted by χ. The bottom line is that we can devise a translation T that answers whether $(s,P) \in [\![\varphi]\!]_F'$, where s is a state of an FTS, by answering $\bar{s} \in [\![T(P,\varphi)]\!]$, where $\bar{s}$ is a state of an LTS with parametrized actions. While a detailed exposition is beyond the scope of the current paper, we illustrate the approach using a small SPL example.

Example 18. *Consider the FTS F modeling a family of (four) coffee machines from Example 6, recalled below (left). Also depicted below (right) is the LTS $L(F)$ with parametrized actions that represents F.*



Let φ' be the $\mu L_f'$-formula $\nu X.\mu Y.\big(([ins|E]Y \wedge [cd|E]Y \wedge [\ell g|E]Y) \wedge [sd|E]X\big)$ expressing that on all infinite runs involving actions ins, cd, ℓg, and sd, the action sd occurs infinitely often. Note that this formula holds in state s_0 and only for products that do not feature the dollar unit (feature D) nor the cleaning and descale unit (feature C). For, if the cleaning and descale unit is present, there is a violating infinite run consisting of cd-actions only, whereas when the dollar unit is present, there is an infinite run containing only ins and ℓg actions.

Assume that the sort PSet represents the set $\mathbf{2}^{\mathscr{F}}$. The set of features $\mathscr{F}$ is finite and, therefore, the sort PSet is easily defined and can be used to effectively compute with. Moreover, we presume all usual

set operators on $2^{\mathscr{F}}$ to have counterparts for PSet too. The μL_{FO}-formula $T(P,\varphi')$ that corresponds to $\varphi' \in \mu L'_f$ above is then the following:

$$\begin{array}{l} \nu X(P_x{:}PSet = P).\mu Y(P_y{:}PSet = P_x).(\\ \quad (P_y \cap E = \varnothing \vee \forall e{:}PSet.[ins(e)](P_y \cap E \cap e = \varnothing \vee Y(P_y \cap E \cap e))) \wedge \\ \quad (P_y \cap E = \varnothing \vee \forall e{:}PSet.[\,cd(e)](P_y \cap E \cap e = \varnothing \vee Y(P_y \cap E \cap e))) \wedge \\ \quad (P_y \cap E = \varnothing \vee \forall e{:}PSet.[\,\ell g(e)](P_y \cap E \cap e = \varnothing \vee Y(P_y \cap E \cap e))) \wedge \\ \quad (P_y \cap E = \varnothing \vee \forall e{:}PSet.[\,sd(e)](P_y \cap E \cap e = \varnothing \vee X(P_y \cap E \cap e))) \;) \end{array}$$

In the resulting formula $T(P,\varphi')$, the modal operators of φ' are essentially mapped to the modal operators of μL_{FO} and the information concerning the feature expressions is handled by the data parameters and appropriate conditions, mirroring the semantics of $\mu L'_f$. Deciding whether $(s_0, \neg(C \vee D)) \in [\![\, \varphi' \,]\!]'_F$ then translates to verifying whether $\bar{s}_0 \in [\![\, T(\neg(C \vee D), \varphi') \,]\!]$. The latter can be done using a toolset such as mCRL2, which supports μL_{FO} and which allows for representing LTS with parametrized actions.

6 Concluding remarks and future work

We have introduced two variants of the modal μ-calculus, each with explicit FTS semantics, by incorporating feature expressions, and we have compared these logics among each other. This resembles work done for LTL [16] and CTL [15], but we can also express typical μ-calculus properties not expressible in LTL or CTL. We have then shown how to achieve family-based model checking of SPL with the existing mCRL2 toolset by exploiting an embedding of the newly introduced feature-oriented μ-calculus variant, with an FTS semantics in terms of sets of products, into the μ-calculus with data. It follows that from a logical point of view the featured μ-calculi proposed here are a sublogic of the μ-calculus with data. However, methodologically the new modalities in μL_f and $\mu L'_f$ highlight via feature expressions the variability and allow to direct the analysis to specific families of products, which may give better insight and, in the preliminary casestudies conducted, quicker response times of the model checker. In [9], we evaluate the application of our approach to a larger SPL model from the literature, the well-known minepump model.

In this paper we have considered family-based *model checking*, but the same principle has also been applied to other analysis techniques, like theorem proving, static analysis, and type checking [33]. Oftentimes this requires extending existing tools, but in some cases—like the one described in this paper—the analysis problem can be encoded in an existing specification language to allow off-the-shelf tools to be reused. In [34], for instance, all Java feature modules and their corresponding feature-based specifications in a Java Modeling Language extension are translated into a single family-based metaspecification that can be passed as-is to the KeY theorem prover [2].

Finally, we outline a couple of possibilities for future work. First, the main results presented in this paper (Theorems 10 and 16) demonstrate that the validity of a formula for a family of products implies its validity for the family's individual products. As is done for MTS in [4], it would be interesting to establish complementary results concerning the preservation of the *in*validity of a formula for a product family by the family's individual products. One possible way to try to achieve this is by providing an alternative semantics for the diamond and box operators denoting may and must modalities.

Second, the implication of Theorem 16 is strengthened to an equivalence in Theorem 17 (i.e. a formula is valid for a family of products iff it is valid for the family's individual products) for a specific subset of feature μ-calculus formulas. It would be interesting to study the conditions under which this equivalence can be obtained for a larger set of feature μ-calculus formulas. Possible strategies include

considering an alternative set of FTS (e.g. with a different structure) or, following [15, 16], separating the feature expressions from the diamond and box operators and instead parametrizing each formula with a feature-based operator that quantifies the specific set of products for which the formula has to be verified.

Acknowledgments

Maurice ter Beek is supported by the EU FP7–ICT FET–Proactive project QUANTICOL, 600708.

References

[1] J.M. Atlee, S. Beidu, N.A. Day, F. Faghih & P. Shaker (2013): *Recommendations for Improving the Usability of Formal Methods for Product Lines*. In S. Gnesi & N. Plat, editors: *FormaliSE*, IEEE, pp. 43–49, doi:10.1109/FormaliSE.2013.6612276.

[2] B. Beckert, R. Hähnle & P.H. Schmitt (2007): *Verification of Object-Oriented Software: The KeY Approach*. Springer, doi:10.1007/978-3-540-69061-0.

[3] M.H. ter Beek, A. Fantechi, S. Gnesi & F. Mazzanti (2015): *Using FMC for Family-Based Analysis of Software Product Lines*. In D.C. Schmidt, editor: *SPLC*, ACM, pp. 432–439, doi:10.1145/2791060.2791118.

[4] M.H. ter Beek, A. Fantechi, S. Gnesi & F. Mazzanti (2016): *Modelling and analysing variability in product families: model checking of modal transition systems with variability constraints*. *J. Log. Algebr. Meth. Program.* 85(2), pp. 287–315, doi:10.1016/j.jlamp.2015.11.006.

[5] M.H. ter Beek, A. Lluch Lafuente & M. Petrocchi (2013): *Combining Declarative and Procedural Views in the Specification and Analysis of Product Families*. In D. Clarke, editor: *FMSPLE*, *SPLC*, 2, ACM, pp. 10–17, doi:10.1145/2499777.2500722.

[6] M.H. ter Beek, F. Mazzanti & A. Sulova (2012): *VMC: A Tool for Product Variability Analysis*. In D. Giannakopoulou & D. Méry, editors: *FM*, *LNCS* 7436, Springer, pp. 450–454, doi:10.1007/978-3-642-32759-9_36.

[7] M.H. ter Beek & E.P. de Vink (2014): *Towards Modular Verification of Software Product Lines with mCRL2*. In T. Margaria & B. Steffen, editors: *ISoLA*, *LNCS* 8802, Springer, pp. 368–385, doi:10.1007/978-3-662-45234-9_26.

[8] M.H. ter Beek & E.P. de Vink (2014): *Using mCRL2 for the Analysis of Software Product Lines*. In S. Gnesi & N. Plat, editors: *FormaliSE*, IEEE, pp. 31–37, doi:10.1145/2593489.2593493.

[9] M.H. ter Beek, E.P. de Vink & T.A.C. Willemse (2016): *Family-based model checking with mCRL2*. Submitted.

[10] T. Belder, M.H. ter Beek & E.P. de Vink (2015): *Coherent branching feature bisimulation*. In J.M. Atlee & S. Gnesi, editors: *FMSPLE*, *EPTCS* 182, pp. 14–30, doi:10.4204/EPTCS.182.2.

[11] P. Borba, M.B. Cohen, A. Legay & A. Wasowski (2013): *Analysis, Test and Verification in The Presence of Variability (Dagstuhl Seminar 13091)*. *Dagstuhl Reports* 3(2), pp. 144–170, doi:10.4230/DagRep.3.2.144.

[12] J.C. Bradfield & C. Stirling (2001): *Modal Logics and μ-Calculi: An Introduction*. In J.A. Bergstra, A. Ponse & S.A. Smolka, editors: *Handbook of Process Algebra*, chapter 4, Elsevier, pp. 293–330, doi:10.1016/B978-044482830-9/50022-9.

[13] E.M. Clarke, O. Grumberg & D.A. Peled (1999): *Model Checking*. The MIT Press.

[14] A. Classen, M. Cordy, P. Heymans, A. Legay & P.-Y. Schobbens (2012): *Model checking software product lines with SNIP*. *Int. J. Softw. Tools Technol. Transf.* 14(5), pp. 589–612, doi:10.1007/s10009-012-0234-1.

[15] A. Classen, M. Cordy, P. Heymans, A. Legay & P.-Y. Schobbens (2014): *Formal semantics, modular specification, and symbolic verification of product-line behaviour*. *Sci. Comput. Program.* 80(B), pp. 416–439, doi:10.1145/2499777.2499781.

[16] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay & J.-F. Raskin (2013): *Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking*. *IEEE Trans. Softw. Eng.* 39(8), pp. 1069–1089, doi:10.1109/TSE.2012.86.

[17] A. Classen, P. Heymans, P.-Y. Schobbens & A. Legay (2011): *Symbolic Model Checking of Software Product Lines*. In R.N. Taylor, H.C. Gall & N. Medvidovic, editors: *ICSE*, ACM, pp. 321–330, doi:10.1145/1985793.1985838.

[18] A. Classen, P. Heymans, P.-Y. Schobbens, A. Legay & J.-F. Raskin (2010): *Model Checking Lots of Systems: Efficient Verification of Temporal Properties in Software Product Lines*. In J. Kramer, J. Bishop, P.T. Devanbu & S. Uchitel, editors: *ICSE*, ACM, pp. 335–344, doi:10.1145/1806799.1806850.

[19] M. Cordy, A. Classen, P. Heymans, P.-Y. Schobbens & A. Legay (2013): *ProVeLines: a product line of verifiers for software product lines*. In: *SPLC*, 2, ACM, pp. 141–146, doi:10.1145/2499777.2499781.

[20] S. Cranen, J.F. Groote, J.J.A. Keiren, F.P.M. Stappers, E.P. de Vink, W. Wesselink & T.A.C. Willemse (2013): *An Overview of the mCRL2 Toolset and Its Recent Advances*. In N. Piterman & S.A. Smolka, editors: *TACAS*, *LNCS* 7795, Springer, pp. 199–213, doi:10.1007/978-3-642-36742-7_15.

[21] M. Erwig & E. Walkingshaw (2011): *The Choice Calculus: A Representation for Software Variation*. *ACM Trans. Softw. Eng. Methodol.* 21(1):6, doi:10.1145/2063239.2063245.

[22] D. Fischbein, S. Uchitel & V.A. Braberman (2006): *A foundation for behavioural conformance in software product line architectures*. In R.M. Hierons & H. Muccini, editors: *ROSATEA*, ACM, pp. 39–48, doi:10.1145/1147249.1147254.

[23] J.F. Groote & R. Mateescu (1999): *Verification of Temporal Properties of Processes in a Setting with Data*. In A.M. Haeberer, editor: *AMAST*, *LNCS* 1548, Springer, pp. 74–90, doi:10.1007/3–540–49253–4_8.

[24] J.F. Groote & M.R. Mousavi (2014): *Modeling and Analysis of Communicating Systems*. The MIT Press.

[25] J.F. Groote & T.A.C. Willemse (2005): *Model-checking processes with data*. *Sci. Comput. Program.* 56(3), pp. 251–273, doi:10.1016/j.scico.2004.08.002.

[26] D. Kozen (1983): *Results on the propositional μ-calculus*. *Theoret. Comput. Sci.* 27(3), pp. 333–354, doi:10.1016/0304–3975(82)90125–6.

[27] K.G. Larsen, U. Nyman & A. Wasowski (2007): *Modal I/O Automata for Interface and Product Line Theories*. In R. De Nicola, editor: *ESOP*, *LNCS* 4421, Springer, pp. 64–79, doi:10.1007/978-3-540-71316-6_6.

[28] K. Lauenroth, K. Pohl & S. Töhning (2009): *Model Checking of Domain Artifacts in Product Line Engineering*. In: *ASE*, IEEE, pp. 269–280, doi:10.1109/ASE.2009.16.

[29] M. Leucker & D. Thoma (2012): *A Formal Approach to Software Product Families*. In T. Margaria & B. Steffen, editors: *ISoLA*, *LNCS* 7609, Springer, pp. 131–145, doi:10.1007/978-3-642-34026-0_11.

[30] M. Lochau, S. Mennicke, H. Baller & L. Ribbeck (2014): *DeltaCCS: A Core Calculus for Behavioral Change*. In T. Margaria & B. Steffen, editors: *ISoLA*, *LNCS* 8802, Springer, pp. 320–335, doi:10.1007/978-3-662-45234-9_23.

[31] M. Lochau, S. Mennicke, H. Baller & L. Ribbeck (2016): *Incremental model checking of delta-oriented software product lines*. *J. Log. Algebr. Meth. Program.* 85(1), pp. 245–267, doi:10.1016/j.jlamp.2015.09.004.

[32] I. Schaefer & R. Hähnle (2011): *Formal Methods in Software Product Line Engineering*. *IEEE Comp.* 44(2), pp. 82–85, doi:10.1109/MC.2011.47.

[33] T. Thüm, S. Apel, C. Kästner, I. Schaefer & G. Saake (2014): *A Classification and Survey of Analysis Strategies for Software Product Lines*. *ACM Comput. Surv.* 47(1), pp. 6:1–6:45, doi:10.1145/2580950.

[34] T. Thüm, I. Schaefer, M. Hentschel & S. Apel (2012): *Family-Based Deductive Verification of Software Product Lines*. In K. Ostermann & W. Binder, editors: *GPCE*, ACM, pp. 11–20, doi:10.1145/2371401.2371404.

[35] M. Tribastone (2014): *Behavioral Relations in a Process Algebra for Variants*. In S. Gnesi, A. Fantechi, P. Heymans, J. Rubin & K. Czarnecki, editors: *SPLC*, ACM, pp. 82–91, doi:10.1145/2648511.2648520.