A Theory of Software Product Line Refinement— proofs

Paulo Borba[a], Leopoldo Teixeira[a], Rohit Gheyi[b]

[a] *Informatics Center, Federal University of Pernambuco, Recife, PE, Brazil*
[b] *Department of Computing Systems, Federal University of Campina Grande, Campina Grande, PB, Brazil*

Abstract

To safely evolve a software product line, it is important to have a notion of product line refinement that assures behavior preservation of the original product line products. So in this article we present a language independent theory of product line refinement, establishing refinement properties that justify stepwise and compositional product line evolution. Moreover, we instantiate our theory with the formalization of specific languages for typical product lines artifacts, and then introduce and prove soundness of a number of associated product line refinement transformation templates. These templates can be used to reason about specific product lines and as a basis to derive comprehensive product line refinement catalogues.

Key words: Software Product Lines, Software Evolution, Refinement, Refactoring

1. Feature Models

Definition 1. ⟨Feature models⟩

$$
\begin{aligned}
&Name : TYPE \\
&FM : < features : \mathcal{P}[Name], formulae : \mathcal{P}[Formula] > \\
&Configuration : \mathcal{P}[Name] \\
&wfFM(fm : FM) : bool = \forall f \in formulae(fm) \cdot wt(f, fm) \\
&WFM : TYPE = (wfFM) \\
&semantics(fm : WFM) : \mathcal{P}[Configuration] = \\
&\quad \{c \,|\, c \subseteq features(fm) \wedge \forall f \in formulae(fm) \cdot satisfies(f, c)\}
\end{aligned}
$$

Lemma 1. ⟨Feature not in FM results in feature not present in configurations⟩
For feature model F, and feature P, if

$$P \notin features(F)$$

then

$$\forall c \in semantics(F) \cdot P \notin c$$

Proof: For arbitrary F, P assume $P \notin features(F)$. For an arbitrary $c \in semantics(F)$, we have to prove

$$P \notin c$$

Email addresses: phmb@cin.ufpe.br (Paulo Borba), lmt@cin.ufpe.br (Leopoldo Teixeira), rohit@dsc.ufcg.edu.br (Rohit Gheyi)

By Definition 1, we have that $c \in semantics(F)$ amounts to

$$c \subseteq features(F) \wedge \forall f \in formulae(F) \cdot satisfies(f, c)$$

The first part of the conjunction establishes that c only contains features from F. From our assumption, we have that

$$P \notin features(F)$$

Therefore, we deduce that P can not be part of c and conclude our proof.

□

2. Assets

Definition 2. ⟨Program refinement⟩
For programs p_1 and p_2, p_2 refines p_1, denoted by

$$p_1 \sqsubseteq p_2$$

when p_2 is at least as good as p_1 in the sense that it will meet every purpose and satisfies every input-output specification satisfied by p_1. We say that p_2 preserves the (observable) behavior of p_1.

Definition 3. ⟨Contextual class declaration refinement⟩
For sets of class declarations cds_1 and cds_2, cds_2 refines cds_1, denoted by

$$cds_1 \sqsubseteq_{cds,m} cds_2$$

in a context cds of "auxiliary" class declarations for cds_1 and cds_2, and a main command m, when

$$cds_1\ cds \bullet m \sqsubseteq cds_2\ cds \bullet m.$$

Definition 4. ⟨Class declaration refinement⟩
For sets of class declarations cds_1 and cds_2, cds_2 refines cds_1, denoted by

$$cds_1 \sqsubseteq cds_2$$

when, for all commands m and sets of class declarations cds, if $cds_1\ cds \bullet m$ is well-typed then $cds_2\ cds \bullet m$ is also well-typed and

$$cds_1 \sqsubseteq_{cds,m} cds_2.$$

Definition 5. ⟨Asset mapping⟩
Formally, we specify AMs in PVS as a finite function from asset names (the $AssetName$ type) to assets (the $Asset$ type), using a parameterized theory. This mapping theory defines the AM type, basic functions, and corresponding properties such as the fact that asset names only map to a single asset.

Definition 6. ⟨Auxiliary asset mapping functions⟩
Let m be an AM and s a set of asset names.

$$\begin{aligned} &dom(m) : \mathcal{P}[AssetName] = \\ &\quad \{n : AssetName \mid \exists a : Asset \cdot m(n) = a\} \\ &m\langle s\rangle : \mathcal{P}[Asset] = \\ &\quad \{a : Asset \mid \exists n \in s \cdot m(n) = a\} \end{aligned}$$

Lemma 2. ⟨Distributed mapping over union⟩
For asset mapping A, asset a, and finite sets of asset names S and S', if

$$a \in A\langle S \cup S'\rangle$$

then

$$a \in A\langle S\rangle \vee a \in A\langle S'\rangle$$

Proof: For arbitrary A, a, S , and S', assume $a \in A\langle S \cup S'\rangle$. From this and Definition 6 ($A\langle\rangle$) we have

$$a \in \{a : Asset \mid \exists n \in S \cup S' \cdot m(n) = a\}$$

From set union and membership properties, we have

$$a \in \{a : Asset \mid \exists n \in S \cdot m(n) = a \vee \exists n \in S' \cdot m(n) = a\}$$

From set comprehension properties, we have

$$a \in \{a : Asset \mid \exists n \in S \cdot m(n) = a\} \cup \{a : Asset \mid \exists n \in S' \cdot m(n) = a\}$$

By applying twice Definition 6 ($A\langle\rangle$), we derive

$$a \in A\langle S\rangle \cup A\langle S'\rangle$$

The proof follows from the above and set membership properties.

□

Lemma 3. ⟨Distributed mapping over singleton⟩
For asset mapping A, asset name an, and finite set of asset names S, if

$$an \in dom(A)$$

then

$$\exists a : Asset \cdot A(an) = a \ \wedge \ A\langle an \cup S\rangle = a \cup A\langle S\rangle$$

Proof: For arbitrary A, an, and S, assume $an \in dom(A)$. From this, Definition 6 (dom), and set comprehension and membership properties, we have

$$\exists a : Asset \cdot A(an) = a \tag{1}$$

Let a_1 be such a. By Definition 6 ($A\langle\rangle$), we have

$$A\langle an \cup S\rangle = \{a : Asset \mid \exists n \in (an \cup S) \cdot A(n) = a\}$$

By set membership and comprehension properties, we have

$$\begin{aligned} A\langle an \cup S\rangle = \\ \{a : Asset \mid \exists n \in \{an\} \cdot A(n) = a\} \\ \cup \{a : Asset \mid \exists n \in S \cdot A(n) = a\} \end{aligned}$$

By Definition 6 ($A\langle\rangle$), our assumption that A is an asset mapping, and set membership and comprehension properties, we have

$$A\langle\{an\} \cup S\rangle = a_1 \cup A\langle S\rangle$$

From this and remembering that 1 was instantiated with a_1, a_1 provides the a we need to conclude the proof.

□

Lemma 4. ⟨Asset mapping domain membership⟩
For asset mapping A, asset name an, and asset a, if

$$A(an) = a$$

then

$$an \in dom(A)$$

Proof: For arbitrary A, an, and a, assume $A(an) = a$. By Definition 6 (dom), we have to prove that

$$\exists x : Asset \mid A(an) = x$$

Let x be a, and this concludes the proof.

□

Lemma 5. ⟨Distributed mapping over set of non domain elements⟩
For asset mapping A and finite set of asset names S, if

$$\neg\exists n \in S \cdot n \in dom(A)$$

then

$$A\langle S\rangle = \{\}$$

Proof: For arbitrary A and S, assume $\neg\exists n \in S \cdot n \in dom(A)$. By Definition 6 ($A\langle\rangle$), we have to prove that

$$\{a : Asset \mid \exists n \in S \cdot A(n) = a\} = \{\}$$

By Lemma 4, we then have to prove that

$$\{a : Asset \mid \exists n \in S \cdot n \in dom(A) \wedge A(n) = a\} = \{\}$$

The proof follows from the above, our assumption, and set comprehension properties.

□

3. Configuration knowledge

Definition 7. ⟨Configuration knowledge⟩

$$
\begin{aligned}
&Item :< exp : Formula, names : \mathcal{F}[AssetName] > \\
&CK : \mathcal{F}[Item] \\
&eval(K : CK, c : Configuration) : \mathcal{F}[AssetName] = \\
&\quad \{an \,|\exists i \in K \cdot satisfies(exp(i), c) \wedge an \in assets(i)\,\} \\
&semantics(K : CK, A : AM, c : Configuration) : \mathcal{F}[Asset] = \\
&\quad A\langle eval(K, c)\rangle
\end{aligned}
$$

Lemma 6. ⟨CK evaluation distributed over union⟩
For configuration knowledge K, asset mapping A, finite sets of items S and S', and configuration c, if

$$K = S \cup S'$$

then

$$semantics(K, A, c) = semantics(S, A, c) \cup semantics(S', A, c)$$

Proof: For arbitrary K, A, S, S', assume $K = S \cup S'$. We have to prove

$$semantics(K, A, c) = semantics(S, A, c) \cup semantics(S', A, c) \tag{2}$$

By Definition 7 and our assumption, $semantics(K, A, c)$ amounts to

$$\{a \,|\exists an \cdot (\exists it \cdot an \in assets(it) \wedge (it \in S \vee it \in S') \wedge satisfies(exp(it), c)) \wedge A(an) = a\}$$

By set comprehension properties, the above is equivalent to

$$
\begin{aligned}
&\{a \,|(\exists an \cdot (\exists it \cdot an \in assets(it) \wedge it \in S \wedge satisfies(exp(it), c)) \wedge A(an) = a) \\
&\vee (\exists an \cdot (\exists it \cdot an \in assets(it) \wedge it \in S' \wedge satisfies(exp(it), c)) \wedge A(an) = a)\}
\end{aligned}
$$

Again, by set comprehension properties, this is the same as writing

$$
\begin{aligned}
&\{a \,|(\exists an \cdot (\exists it \cdot an \in assets(it) \wedge it \in S \wedge satisfies(exp(it), c)) \wedge A(an) = a)\} \cup \\
&\{a \,|(\exists an \cdot (\exists it \cdot an \in assets(it) \wedge it \in S' \wedge satisfies(exp(it), c)) \wedge A(an) = a)\}
\end{aligned}
$$

The above is equivalent to $semantics(S, A, c) \cup semantics(S', A, c)$, which is what we need to prove— see (2).

□

4. A theory of product line refinement

Definition 8. ⟨Feature model equivalence⟩
Feature models F and F' are equivalent, denoted $F \cong F'$, whenever $[\![F]\!] = [\![F']\!]$.

Theorem 1. ⟨Feature model equivalence⟩

$$\forall F : FeatureModel \cdot F \cong F$$
$$\forall F, F' : FeatureModel \cdot F \cong F' \Rightarrow F' \cong F$$
$$\forall F, F', F'' : FeatureModel \cdot F \cong F' \wedge F' \cong F'' \Rightarrow F \cong F''$$

Proof: Directly from Definition 8 and the reflexivity, symmetry, and transitivity of the equality of configuration sets.

□

Assumption 1. ⟨Asset refinement⟩

$$Asset : TYPE$$
$$\sqsubseteq : \mathcal{P}[Asset], \mathcal{P}[Asset] \rightarrow bool$$
$$wf : \mathcal{P}[Asset] \rightarrow bool$$

For brevity, we use $a \sqsubseteq a'$, for assets a and a', as an abbreviation for $\{a\} \sqsubseteq \{a'\}$. We also use $a\,a'$ as an abbreviation for $\{a, a'\}$.

Axiom 1. ⟨Asset set refinement is pre-order⟩

$$\forall a : \mathcal{P}[Asset] \cdot a \sqsubseteq a$$
$$\forall a, b, c : \mathcal{P}[Asset] \cdot a \sqsubseteq b \wedge b \sqsubseteq c \Rightarrow a \sqsubseteq c$$

Axiom 2. ⟨Asset set refinement compositionality⟩

$$\forall a, a' : \mathcal{P}[Asset] \cdot \forall s : \mathcal{P}[Asset] \cdot$$
$$a \sqsubseteq a' \wedge wf(a \cup s)$$
$$\Rightarrow wf(a' \cup s) \wedge a \cup s \sqsubseteq a' \cup s$$

Definition 9. ⟨Asset mapping refinement⟩
For asset mappings A and A', the second refines the first, denoted

$$A \sqsubseteq A'$$

whenever

$$dom(A) = dom(A')$$
$$\wedge \forall n \in dom(A) \cdot$$
$$\exists a, a' : Asset \cdot A(n) = a \wedge A'(n) = a' \wedge a \sqsubseteq a'$$

Theorem 2. ⟨Asset mapping refinement pre-order⟩

$$\forall A : AM \cdot A \sqsubseteq A$$
$$\forall A, A', A'' : AM \cdot A \sqsubseteq A' \wedge A' \sqsubseteq A'' \Rightarrow A \sqsubseteq A''$$

Proof: Directly from Definition 9 and the reflexivity and transitivity of asset refinement (see Axiom 1).

□

Theorem 3. ⟨Asset mapping compositionality⟩
For asset mapping A and A', if

$$A \sqsubseteq\ A'$$

then

$$\begin{aligned}&\forall ans : \mathcal{F}[AssetName] \cdot \forall as : \mathcal{F}[Asset] \cdot \\ &\quad wf(as \cup A\langle ans\rangle) \\ &\Rightarrow wf(as \cup A'\langle ans\rangle) \wedge as \cup A\langle ans\rangle \sqsubseteq as \cup A'\langle ans\rangle\end{aligned}$$

Proof: For arbitrary A and A', assume $A \sqsubseteq\ A'$. By Definition 9, this amounts to

$$\begin{aligned}&dom(A) = dom(A') \\ &\wedge \forall n \in dom(A) \cdot \\ &\quad \exists a, a' : Asset \cdot A(n) = a \wedge A'(n) = a' \wedge a \sqsubseteq a'\end{aligned} \tag{3}$$

By induction on the cardinality of ans, assume the induction hypothesis

$$\begin{aligned}&\forall ans' : \mathcal{F}[AssetName] \cdot \\ &\quad card(ans') < card(ans) \\ &\Rightarrow \forall as : \mathcal{F}[Asset] \cdot \\ &\quad wf(as \cup A\langle ans'\rangle) \\ &\Rightarrow wf(as \cup A'\langle ans'\rangle) \wedge as \cup A\langle ans'\rangle \sqsubseteq as \cup A'\langle ans\rangle\end{aligned} \tag{4}$$

and we have to prove

$$\begin{aligned}&\forall as : \mathcal{F}[Asset] \cdot \\ &\quad wf(as \cup A\langle ans\rangle) \\ &\Rightarrow wf(as \cup A'\langle ans\rangle) \wedge as \cup A\langle ans\rangle \sqsubseteq as \cup A'\langle ans\rangle\end{aligned} \tag{5}$$

By case analysis, now consider that $\neg(\exists an \in ans \cdot an \in dom(A))$. By Lemma 5, we have that $A\langle ans\rangle = \emptyset$. Similarly, given that $dom(A) = dom(A')$ (see 3), we also have that $A'\langle ans\rangle = \emptyset$. So, by set union properties, we are left to prove that

$$\forall as : \mathcal{F}[Asset] \cdot wf(as) \Rightarrow wf(as) \wedge as \sqsubseteq as$$

The proof trivially follows from asset set refinement reflexivity (see Axiom 1) and propositional calculus.

Let us now consider the case $\exists an \in ans \cdot an \in dom(A)$. By basic set properties, we have that $ans = an \cup ans'$ for some asset name $an \in dom(A)$ and set ans' such that $an \notin ans'$. Then, from 5, we are left to prove that

$$\begin{aligned}&\forall as : \mathcal{F}[Asset] \cdot \\ &\quad wf(as \cup A\langle an \cup ans'\rangle) \\ &\Rightarrow\ \ wf(as \cup A'\langle an \cup ans'\rangle) \\ &\quad \wedge as \cup A\langle an \cup ans'\rangle \sqsubseteq as \cup A'\langle an \cup ans'\rangle\end{aligned}$$

By Lemma 3, given that $an \in dom(A)$ and consequently $an \in dom(A')$, we have that $A\langle an \cup ans'\rangle = a \cup A\langle ans'\rangle$ and $A'\langle an \cup ans'\rangle = a' \cup A'\langle ans'\rangle$ for some assets a and a'. From 3, we also have that $a \sqsubseteq a'$. By equational reasoning, we then have to prove that

$$\begin{aligned}&\forall as : \mathcal{F}[Asset] \cdot \\ &\quad wf(as \cup a \cup A\langle ans'\rangle) \\ &\Rightarrow\ \ wf(as \cup a' \cup A'\langle ans'\rangle) \\ &\quad \wedge as \cup a \cup A\langle ans'\rangle \sqsubseteq as \cup a' \cup A'\langle ans'\rangle\end{aligned}$$

For an arbitrary as, assume $wf(as \cup a \cup A\langle ans' \rangle)$ and then we have to prove that

$$\begin{array}{l} wf(as \cup a' \cup A'\langle ans' \rangle) \\ \wedge\ as \cup a \cup A\langle ans' \rangle \sqsubseteq as \cup a' \cup A'\langle ans' \rangle \end{array} \tag{6}$$

By the induction hypothesis (see 4), instantiating ans' with the ans' just introduced, note that we will have $card(ans') < card(ans)$ and, therefore

$$\begin{array}{l} \forall as : \mathcal{F}[Asset] \cdot \\ \quad wf(as \cup A\langle ans' \rangle) \\ \quad \Rightarrow wf(as \cup A'\langle ans' \rangle) \wedge as \cup A\langle ans' \rangle \sqsubseteq as \cup A'\langle ans \rangle \end{array}$$

From this, instantiating as as $as \cup a$, and remembering that we have already assumed $wf(as \cup a \cup A\langle ans' \rangle)$, we have

$$\begin{array}{l} wf(as \cup A'\langle ans' \rangle) \\ \wedge\ as \cup A\langle ans' \rangle \sqsubseteq as \cup A'\langle ans \rangle \end{array}$$

Now, given that $a \sqsubseteq a'$, from the compositionality axiom (Axiom 2) and the above we have that

$$\begin{array}{l} wf(as \cup a' \cup A'\langle ans' \rangle) \\ \wedge\ as \cup a \cup A\langle ans' \rangle \sqsubseteq as \cup a' \cup A'\langle ans' \rangle \end{array}$$

The proof then follows from 6, the above, and Axiom 1.

□

Assumption 2. ⟨CK semantics⟩

$$\begin{array}{l} CK : TYPE \\ [\![\_]\!] : CK \rightarrow AM \rightarrow Configuration \rightarrow \mathcal{F}[Asset] \end{array}$$

Definition 10. ⟨Configuration knowledge equivalence⟩
Configuration knowledge K is equivalent to K', denoted $K \cong K'$, whenever $[\![K]\!] = [\![K']\!]$.

Theorem 4. ⟨Configuration knowledge equivalence⟩

$$\forall K : CK \cdot K \cong K$$

$$\forall K, K' : CK \cdot K \cong K' \Rightarrow K' \cong K$$

$$\forall K, K', K'' : CK \cdot K \cong K' \wedge K' \cong K'' \Rightarrow K \cong K''$$

Proof: Directly from Definition 10 and the reflexivity, symmetry, and transitivity of the equality of functions.

□

Axiom 3. ⟨Configuration knowledge evaluation over asset mapping refinement⟩

$$\begin{array}{l} \forall A, A' : AM \cdot A \sqsubseteq A' \Rightarrow \\ \quad \forall K : CK, c : Configuration \cdot \\ \quad\quad wf([\![K]\!]_c^A) \\ \quad\quad \Rightarrow wf([\![K]\!]_c^{A'}) \wedge [\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'} \end{array}$$

Theorem 5. ⟨Instantiation of Axiom 3⟩
For asset mapping A and A', if

$$A \sqsubseteq\ A'$$

then

$$\begin{aligned}&\forall K : CK, c : Configuration \cdot \\ &\quad wf([\![K]\!]_c^A) \\ &\Rightarrow wf([\![K]\!]_c^{A'}) \wedge [\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'}\end{aligned}$$

Proof: For arbitrary A and A', assume $A \sqsubseteq\ A'$. For arbitrary K and c, we have to prove

$$\begin{aligned}&wf([\![K]\!]_c^A) \\ &\Rightarrow wf([\![K]\!]_c^{A'}) \wedge [\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'}\end{aligned}$$

By Definition 7, which defines the *semantics* function used to instantiate the general theory, this amounts to

$$\begin{aligned}&wf(A\langle eval(K,c)\rangle) \\ &\Rightarrow wf(A'\langle eval(K,c)\rangle) \wedge A\langle eval(K,c)\rangle \sqsubseteq A'\langle eval(K,c)\rangle\end{aligned} \tag{7}$$

From Lemma 3, instantiated with A and A', we have

$$\begin{aligned}&\forall ans : \mathcal{F}[AssetName] \cdot \forall as : \mathcal{F}[Asset] \cdot \\ &\quad wf(as \cup A\langle ans\rangle) \\ &\quad \Rightarrow wf(as \cup A'\langle ans\rangle) \wedge as \cup A\langle ans\rangle \sqsubseteq as \cup A'\langle ans\rangle\end{aligned}$$

Instantiating ans with $eval(K,c)$ and as with $\emptyset$, we have

$$\begin{aligned}&wf(\emptyset \cup A\langle eval(K,c)\rangle) \\ &\Rightarrow wf(\emptyset \cup A'\langle eval(K,c)\rangle) \wedge \emptyset \cup A\langle eval(K,c)\rangle \sqsubseteq \emptyset \cup A'\langle eval(K,c)\rangle\end{aligned}$$

By set properties, we have

$$\begin{aligned}&wf(A\langle eval(K,c)\rangle) \\ &\Rightarrow wf(A'\langle eval(K,c)\rangle) \wedge A\langle eval(K,c)\rangle \sqsubseteq A'\langle eval(K,c)\rangle\end{aligned}$$

which is what we have to prove— see (7).

□

5. Product lines

Definition 11. ⟨Product line⟩
For a feature model F, an asset mapping A, and a configuration knowledge K, we say that tuple

$$(F, A, K)$$

is a product line when, for all $c \in [\![F]\!]$,

$$wf([\![K]\!]_c^A)$$

Lemma 7. ⟨Well-formedness preservation under feature model equivalence⟩
For feature models F and F', asset mapping A, and configuration knowledge K, if

$$F \cong F' \wedge \forall c \in [\![F]\!] \cdot wf([\![K]\!]_c^A)$$

then

$$\forall c \in [\![F']\!] \cdot wf([\![K]\!]_c^A)$$

Proof: We have that $F \cong F'$. By definition, this amounts to $[\![F]\!] = [\![F']\!]$. Since we have that $\forall c \in [\![F]\!] \cdot wf([\![K]\!]_c^A)$, we replace $[\![F']\!]$ with $[\![F]\!]$ and conclude the proof.

□

Lemma 8. ⟨Well-formedness preservation under configuration knowledge equivalence⟩
For feature model F, asset mapping A, and configuration knowledge K and K', if

$$K \cong K' \wedge \forall c \in [\![F]\!] \cdot wf([\![K]\!]_c^A)$$

then

$$\forall c \in [\![F]\!] \cdot wf([\![K']\!]_c^A)$$

Proof: Using similar reasoning as in the proof of Lemma 7.

□

Lemma 9. ⟨Well-formedness preservation under asset mapping refinement⟩
For feature model F, asset mapping A and A' and configuration knowledge K, if

$$A \sqsubseteq A' \wedge \forall c \in [\![F]\!] \cdot wf([\![K]\!]_c^A)$$

then

$$\forall c \in [\![F]\!] \cdot wf([\![K]\!]_c^{A'})$$

Proof: For arbitrary F, A, A', and K, assume

$$A \sqsubseteq A' \wedge \forall c \in [\![F]\!] \cdot wf([\![K]\!]_c^A) \tag{8}$$

For an arbitrary $c \in [\![F]\!]$, we then have to prove that

$$wf([\![K]\!]_c^{A'}) \tag{9}$$

By properly instantiating the assumption (8) with the just introduced c, we have

$$wf([\![K]\!]_c^A) \tag{10}$$

From Axiom 3, properly instantiated with A and A', and the assumption (8), we have

$$\begin{aligned} &\forall K : CK, c : Configuration \cdot \\ &\quad wf([\![K]\!]_c^A) \\ &\Rightarrow wf([\![K]\!]_c^{A'}) \wedge [\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'} \end{aligned}$$

Instantiating the above with K and c, we have

$$\begin{aligned} &wf([\![K]\!]_c^A) \\ &\Rightarrow wf([\![K]\!]_c^{A'}) \wedge [\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'} \end{aligned}$$

The proof (see 9) then follows from the above and 10.

□

6. Product line refinement

Definition 12. ⟨Product line refinement⟩
For product lines (F, A, K) and (F', A', K'), the second refines the first, denoted

$$(F, A, K) \sqsubseteq (F', A', K')$$

whenever

$$\forall c \in [\![F]\!] \cdot \exists c' \in [\![F']\!] \cdot [\![K]\!]_c^A \sqsubseteq [\![K']\!]_{c'}^{A'}$$

Theorem 6. ⟨Product line refinement reflexivity⟩

$$\forall l : ProductLine \cdot l \sqsubseteq l$$

Proof: Let $l = (F, A, K)$ be an arbitrary PL. By Definition 12, we have to prove that

$$\forall c \in [\![F]\!] \cdot \exists c' \in [\![F]\!] \cdot [\![K]\!]_c^A \sqsubseteq [\![K]\!]_{c'}^A$$

For an arbitrary $c \in [\![F]\!]$, just let c' be c and the proof follows from asset set refinement reflexivity (see Axiom 1).

□

Theorem 7. ⟨Product line refinement transitivity⟩

$$\forall l_1, l_2, l_3 : ProductLine \cdot l_1 \sqsubseteq l_2 \wedge l_2 \sqsubseteq l_3 \Rightarrow l_1 \sqsubseteq l_3$$

Proof: Let $l_1 = (F_1, A_1, K_1), l_2 = (F_2, A_2, K_2), l_3 = (F_3, A_3, K_3)$ be arbitrary PLs. Assume that $l_1 \sqsubseteq l_2 \wedge l_2 \sqsubseteq l_3$. By Definition 12, this amounts to

$$\forall c_1 \in [\![F_1]\!] \wedge \exists c_2 \in [\![F_2]\!] \cdot [\![K_1]\!]_{c_1}^{A_1} \sqsubseteq [\![K_2]\!]_{c_2}^{A_2} \tag{11}$$

and

$$\forall c_2 \in [\![F_2]\!] \cdot \exists c_3 \in [\![F_3]\!] \cdot [\![K_2]\!]_{c_2}^{A_2} \sqsubseteq [\![K_3]\!]_{c_3}^{A_3} \tag{12}$$

We then have to prove that

$$\forall c_1 \in [\![F_1]\!] \cdot \exists c_3 \in [\![F_3]\!] \cdot [\![K_1]\!]_{c_1}^{A_1} \sqsubseteq [\![K_3]\!]_{c_3}^{A_3}$$

For an arbitrary $c_1 \in [\![F_1]\!]$, we have to prove that

$$\exists c_3 \in [\![F_3]\!] \cdot [\![K_1]\!]_{c_1}^{A_1} \sqsubseteq [\![K_3]\!]_{c_3}^{A_3} \tag{13}$$

Properly instantiating c_1 in 11, we have

$$\exists c_2 \in [\![F_2]\!] \cdot [\![K_1]\!]_{c_1}^{A_1} \sqsubseteq [\![K_2]\!]_{c_2}^{A_2}$$

Let c_2' be such c_2. Properly instantiating c_2' in 12, we have

$$\exists c_3 \in [\![F_3]\!] \cdot [\![K_2]\!]_{c_2'}^{A_2} \sqsubseteq [\![K_3]\!]_{c_3}^{A_3}$$

Let c_3' be such c_3. Then we have

$$[\![K_1]\!]_{c_1}^{A_1} \sqsubseteq [\![K_2]\!]_{c_2'}^{A_2} \wedge [\![K_2]\!]_{c_2'}^{A_2} \sqsubseteq [\![K_3]\!]_{c_3'}^{A_3}$$

By asset set refinement transitivity (see Axiom 1), we have

$$[\![K_1]\!]_{c_1}^{A_1} \sqsubseteq [\![K_3]\!]_{c_3'}^{A_3}$$

This gives us the c_3 in 13 that completes our proof.

□

6.1. Product line refinement compositionality

Theorem 8. ⟨Feature model equivalence compositionality⟩
For product lines (F, A, K) and (F', A, K), if

$$F \cong F'$$

then

$$(F, A, K) \sqsubseteq (F', A, K)$$

Proof: For arbitrary F, F', A, K, assume that $F \cong F'$. By Definition 12, we have to prove that

$$\forall c \in [\![F]\!] \cdot \exists c' \in [\![F']\!] \cdot [\![K]\!]_c^A \sqsubseteq [\![K]\!]_{c'}^A$$

From our assumption and Definition 8, this is equivalent to

$$\forall c \in [\![F]\!] \cdot \exists c' \in [\![F]\!] \cdot [\![K]\!]_c^A \sqsubseteq [\![K]\!]_{c'}^A$$

For an arbitrary $c \in [\![F]\!]$, just let c' be c and the proof follows from asset set refinement reflexivity (see Axiom 1).

□

Theorem 9. ⟨Configuration knowledge equivalence compositionality⟩
For product lines (F, A, K) and (F, A, K'), if

$$K \cong K'$$

then

$$(F, A, K) \sqsubseteq (F, A, K')$$

Proof: The proof is similar to that of Theorem 8, using Definition 10 instead of Definition 8.

□

Theorem 10. ⟨Asset mapping refinement compositionality⟩
For product lines (F, A, K) and (F, A', K), if

$$A \sqsubseteq A'$$

then

$$(F, A, K) \sqsubseteq (F, A', K)$$

Proof: For arbitrary F, A, A', and K, assume that (F, A, K) is a PL and that $A \sqsubseteq A'$. By Definition 12, we have to prove that

$$\forall c \in [\![F]\!] \cdot \exists c' \in [\![F]\!] \cdot [\![K]\!]_c^A \sqsubseteq [\![K]\!]_{c'}^{A'}$$

For an arbitrary $c \in [\![F]\!]$, let c' be such c, we then have to prove

$$[\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'} \tag{14}$$

By Axiom 3 and our assumption, we have that

$$\begin{aligned} &\forall K : CK, c : Configuration \cdot \\ &\quad wf([\![K]\!]_c^A) \\ &\Rightarrow wf([\![K]\!]_c^{A'}) \wedge [\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'} \end{aligned} \tag{15}$$

By properly instantiating K and c in 15, we obtain

$$\begin{aligned} &wf([\![K]\!]_c^A) \\ &\Rightarrow wf([\![K]\!]_c^{A'}) \wedge [\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'} \end{aligned} \tag{16}$$

From Definition 11, we have that $wf([\![K]\!]_c^A)$ for all $c \in [\![F]\!]$. Therefore, from this and 16 we obtain

$$wf([\![K]\!]_c^{A'}) \wedge [\![K]\!]_c^A \sqsubseteq [\![K]\!]_c^{A'}$$

concluding the proof (see 14).

□

Theorem 11. ⟨Full compositionality⟩
For product lines (F, A, K) and (F', A', K'), if

$$F \cong F' \wedge A \sqsubseteq A' \wedge K \cong K'$$

then

$$(F, A, K) \sqsubseteq (F', A', K')$$

Proof: First assume that $F \cong F'$, $A \sqsubseteq A'$, and $K \cong K'$. By Lemma 7, the fact that (F, A, K) is a PL, and Definition 11, we have that (F', A, K) is a PL. Then, using Theorem 8, we have

$$(F, A, K) \sqsubseteq (F', A, K) \tag{17}$$

Similarly, from our assumptions, deductions, and Lemma 8 we have that (F', A, K') is a PL. Using Theorem 9, we have

$$(F', A, K) \sqsubseteq (F', A, K') \tag{18}$$

Again, from our assumptions, deductions, and Lemma 9, we have that (F', A', K') is a PL. Using Theorem 10, we have

$$(F', A, K') \sqsubseteq (F', A', K') \tag{19}$$

The proof then follows from 17, 18, 19, and PL refinement transitivity (see Theorem 7).

□

7. Templates

For each transformation, we encode its soundness theorem as follows. We overload the *syntax* and *conditions* predicates with the specific transformation details:

$$\begin{aligned}
&\forall F, A, K, F', A', K' \ldots \cdot \\
&\quad wfPL(F, A, K) \wedge syntax(\ldots) \wedge conditions(\ldots) \\
&\qquad \Rightarrow (F, A, K) \sqsubseteq (F', A', K') \wedge wfPL(F', A', K')
\end{aligned}$$

Definition 13. ⟨Well-formed Product line⟩
For a feature model F, an asset mapping A, and a configuration knowledge K, we say that tuple (F, A, K) is a well-formed PL, denoted by $wfPL(F, A, K)$, when

$$\begin{aligned}
&\forall c \in [\![F]\!] \cdot wf([\![K]\!]_c^A) \wedge \\
&\forall exp \in exps(K) \cdot wt(exp, F) \wedge \\
&\forall c \in [\![F]\!] \cdot eval(K, c) \subseteq dom(A)
\end{aligned}$$

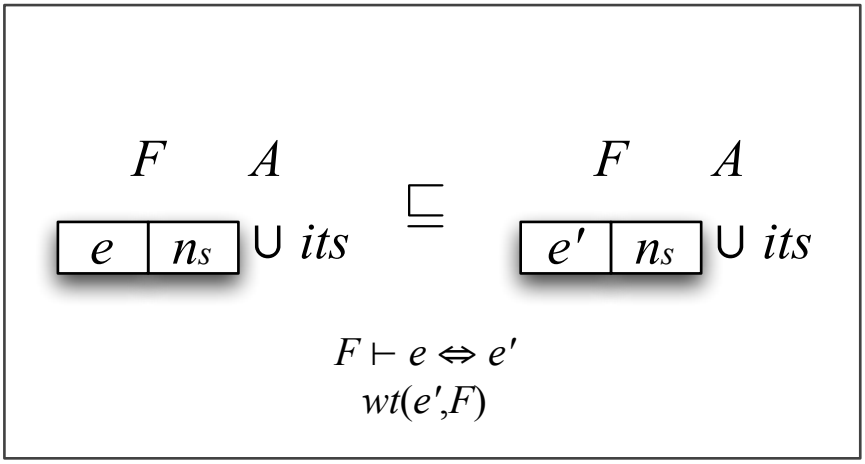

Figure 1: Replace feature expression refinement template

7.1. Replace feature expression

$$\begin{aligned} &syntax(K, K', i_1, i_2, its) = \\ &\quad K = \{i_1\} \cup its \wedge \\ &\quad K' = \{i_2\} \cup its \wedge \\ &\quad names(i_1) = names(i_2) \end{aligned} \tag{20}$$

$$\begin{aligned} &conditions(F, i_1, i_2) = \\ &\quad \forall c \in [\![F]\!] \cdot satisfies(exp(i_1), c) \Leftrightarrow satisfies(exp(i_2), c) \wedge \\ &\quad wt(exp(i_2), F) \end{aligned} \tag{21}$$

Lemma 10. ⟨Replace feature expression results in equal CK evaluation⟩
For product line (F, A, K), configuration knowledge K', CK items i_1, i_2, and the set of CK items its, if

$$syntax(K, K', i_1, i_2, its) \wedge conditions(F, i_1, i_2)$$

then

$$\forall c \in [\![F]\!] \cdot [\![K]\!]_c^A = [\![K']\!]_c^A$$

Proof: For arbitrary F, A, K, K', i_1, i_2, its, assume the *syntax* and *conditions* predicates previously illustrated (see Predicates 20 and 21). We have to prove that

$$\forall c \in [\![F]\!] \cdot [\![K]\!]_c^A = [\![K']\!]_c^A$$

For an arbitrary $c \in [\![F]\!]$, let c' be c and we have then to prove that

$$[\![K]\!]_c^A = [\![K']\!]_c^A$$

From the syntax predicate (see Predicate 20), we have that

$$K = \{i_1\} \cup its \wedge K' = \{i_2\} \cup its$$

Therefore, by Lemma 6, we have to prove that

$$[\![i_1]\!]_c^A \cup [\![its]\!]_c^A = [\![i_2]\!]_c^A \cup [\![its]\!]_c^A$$

As evaluating a CK leads to the assets associated with a valid feature expression for a given configuration, we know that evaluating *its* generates the same assets. We are left with i_1 and i_2. By the *syntax* predicate (see Predicate 20), we also have that $names(i_1) = names(i_2)$. Since from the conditions predicate (see Predicate 21) we have that

$$\forall c \in [\![F]\!] \cdot satisfies(exp(i_1), c) \Leftrightarrow satisfies(exp(i_2), c)$$

The proof follows instantiating the above with the c previously introduced.

□

7.2. Add mandatory feature

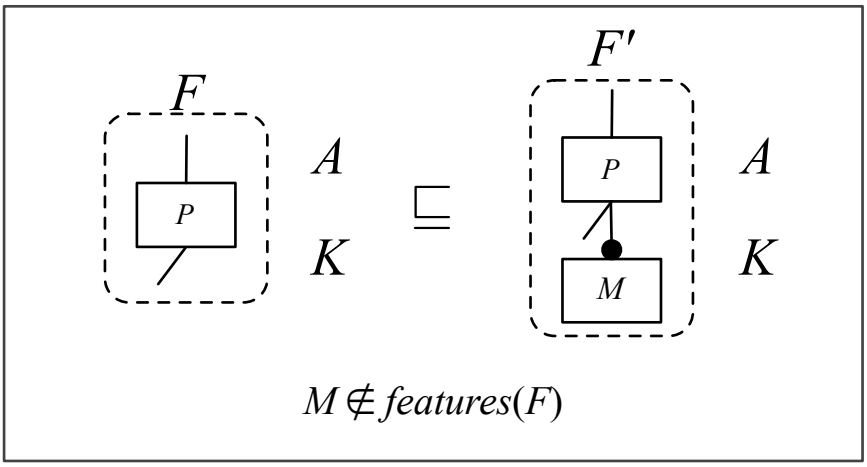

Figure 2: Add mandatory feature refinement template

$$\begin{aligned} &syntax(F, F', P, M) = \\ &\quad features(F') = features(F) \cup \{M\} \wedge \\ &\quad formulae(F') = formulae(F) \cup \{P \Leftrightarrow M\} \wedge \\ &\quad P \in features(F) \end{aligned} \tag{22}$$

$$\begin{aligned} &conditions(F, M) = \\ &\quad M \notin features(F) \end{aligned} \tag{23}$$

Lemma 11. ⟨Add mandatory node to feature model⟩
For feature models F and F', feature names P and M, if

$$syntax(F, F', P, M) \wedge conditions(F, M)$$

then

$$\forall c \in [\![F]\!] \cdot IF\ (P \in c)\ (c \cup \{M\} \in [\![F']\!])\ ELSE\ (c \in [\![F']\!]) \wedge wfFM(F')$$

Proof: For arbitrary F, F', P, M, assume the *syntax* and *conditions* predicates previously illustrated (see Predicates 22 and 23). For the first predicate, we have to prove, for an arbitrary $c \in [\![F]\!]$, that

$$IF\ (P \in c)\ (c \cup \{M\} \in [\![F']\!])\ ELSE\ (c \in [\![F']\!])$$

By case analysis, consider that $P \in c$. We then have to prove that $c \cup \{M\} \in [\![F']\!]$. Considering that *semantics* is the correspondent $[\![\_]\!]$ for the feature model language used (see Definition 1), we have to prove that

$$c \cup \{M\} \subseteq features(F') \wedge \forall f \in formulae(F') \cdot satisfies(f, c \cup \{M\})$$

From the *syntax* predicate, we have that $features(F') = features(F) \cup \{M\}$. From this and the fact that $c \in [\![(]\!]F)$, we know that c only contains features from F, thus, we prove the first part of the conjunction. From the same predicate (*syntax*), we also have that

$$formulae(F') = formulae(F) \cup \{P \Leftrightarrow M\}$$

The elements in $formulae(F)$ are already satisfied in c, and we do not remove any formulae from the original F. Since we have that $P \in c$, $P \Leftrightarrow M$ is satisfied in $c \cup \{M\}$, since M is a mandatory child of P. We now consider the case where $P \notin c$. We then have to prove that

$$c \in [\![F']\!]$$

This is straightforward to prove, since M can not be part of c, due to the mandatory relationship with P.

For the second predicate, we have to prove that $wfFM(F')$. By Definition 1, this amounts to

$$\forall f \in formulae(F') \cdot wt(f, F')$$

From the *syntax* predicate, we have that

$$formulae(F') = formulae(F) \cup \{P \Leftrightarrow M\}$$

From the definition of F, we have that $wfFM(F)$, which amounts to

$$\forall f \in formulae(F) \cdot wt(f, F)$$

From the above, and the fact that we do not remove any features from F— only adding M, we can prove that all formulae only contain names from the feature model F', since the new formula only references P, which is an existing feature in F, and M, which is the new added feature.

□

Lemma 12. ⟨Dead feature does not affect CK evaluation⟩
For product line (F, A, K) and feature name M, if

$$wfPL(F, A, K) \wedge conditions(F, M)$$

then

$$\forall c \in [\![F]\!] \cdot [\![K]\!]^A_c = [\![K]\!]^A_{c \cup \{M\}}$$

Proof: For arbitrary F, A, K, M, assume $wfPL(F, A, K)$ and the *conditions* predicate previously illustrated (see Predicate 23). We have to prove, for an arbitrary $c \in [\![F]\!]$, that

$$[\![K]\!]^A_c = [\![K]\!]^A_{c \cup \{M\}}$$

By the *conditions* predicate we have that

$$M \notin features(F)$$

Since we have that $wfPL(F, A, K)$, by Definition 13, we have that

$$\forall exp \in exps(K) \cdot wt(exp, F)$$

Therefore, from the above we have that no feature expression in K refers to M, since they only refer to features in F. So, evaluating K with or without M does not change the results and we conclude the proof.

□

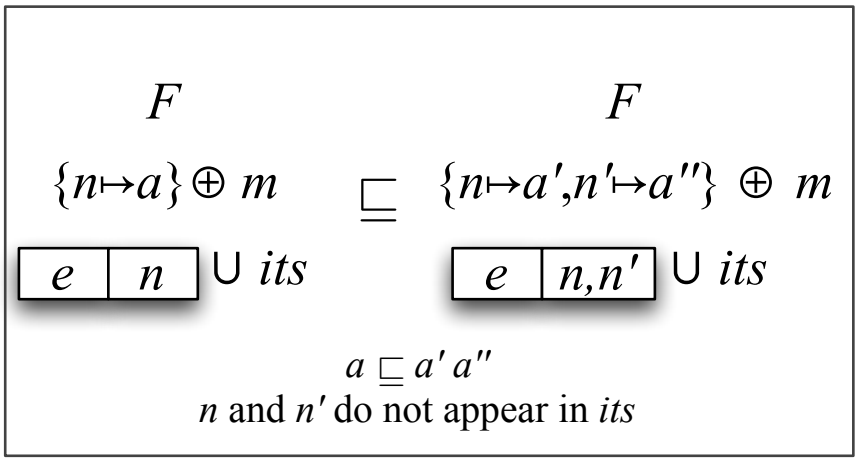

Figure 3: Split assets refinement template

$$
\begin{aligned}
&syntax(A, K, A', K', i_1, i_2, its, n, n', a, a', a'', m) = \\
&\quad A = \{n \to a\} \oplus m \wedge \\
&\quad A' = \{n \to a', n' \to a''\} \oplus m \wedge \\
&\quad K = \{i_1\} \cup its \wedge \\
&\quad K' = \{i_2\} \cup its \wedge \\
&\quad exp(i_1) = exp(i_2) \wedge \\
&\quad names(i_1) = \{n\} \wedge \\
&\quad names(i_2) = \{n, n'\}
\end{aligned}
\tag{24}
$$

$$
\begin{aligned}
&conditions(a, a', a'', n, n', its) = \\
&\quad a \sqsubseteq a'\, a'' \wedge \\
&\quad \forall\, it \in its \cdot n \notin names(it) \wedge n' \notin names(it)
\end{aligned}
\tag{25}
$$

Lemma 13. ⟨Configuration knowledge evaluation is the same for remaining CK items⟩
For product line (F, A, K), asset mapping A', configuration knowledge K', CK items i_1 and i_2, CK items set its, asset names n and n', assets a, a' and a'', and set of mappings m, if

$$syntax(A, K, A', K', i_1, i_2, its, n, n', a, a', a'', m) \wedge conditions(a, a', a'', n, n', its)$$

then

$$\forall c \in [\![F]\!] \Rightarrow [\![its]\!]_c^A = [\![its]\!]_c^{A'}$$

Proof: For arbitrary F, A, K, A', K', i_1, i_2, its, n, n', a, a', a'', m, assume the *syntax* and *conditions* predicates previously illustrated (see Predicates 24 and 25). We have to prove that

$$\forall c \in [\![F]\!] \Rightarrow [\![its]\!]_c^A = [\![its]\!]_c^{A'}$$

For arbitrary $c \in [\![F]\!]$, we have to prove that

$$[\![its]\!]_c^A = [\![its]\!]_c^{A'}$$

From the *syntax* predicate, we have that

$$A = \{n \to a\} \oplus m \ \wedge \ A' = \{n \to a', n' \to a''\} \oplus m$$

We see that the differences between A and A' are related to asset names n and n'. From the *conditions* predicate, we have that

$$\forall it \in its \cdot n \notin names(it) \wedge n' \notin names(it)$$

Since items in its do not refer to n or n', we have that evaluating its with A or A' yields the same assets set, for any $c \in [\![F]\!]$.

□

Lemma 14. ⟨Configuration knowledge evaluation is not affected when item is not activated⟩
For product line (F, A, K), asset mapping A', configuration knowledge K', CK items i_1 and i_2, CK items set its, asset names n and n', assets a, a' and a'', and set of mappings m, if

$$syntax(A, K, A', K', i_1, i_2, its, n, n', a, a', a'', m) \wedge conditions(a, a', a'', n, n', its)$$

then

$$\forall c \in [\![F]\!] \cdot \neg satisfies(exp(i_1), c) \Rightarrow [\![K]\!]_c^A = [\![K']\!]_c^{A'}$$

Proof: For arbitrary F, A, K, A', K', i_1, i_2, its, n, n', a, a', a'', m, assume the *syntax* and *conditions* predicates previously illustrated (see Predicates 24 and 25). We have to prove that

$$\forall c \in [\![F]\!] \cdot \neg satisfies(exp(i_1), c) \Rightarrow [\![K]\!]_c^A = [\![K']\!]_c^{A'}$$

For arbitrary $c \in [\![F]\!]$, we assume $\neg satisfies(exp(i_1), c)$ and have to prove

$$[\![K]\!]_c^A = [\![K']\!]_c^{A'}$$

From the *syntax* predicate, we have that

$$K = \{i_1\} \cup its \ \wedge \ K' = \{i_2\} \cup its$$

Thus, by Lemma 6, we have to prove that

$$[\![i_1]\!]_c^A \cup [\![its]\!]_c^A = [\![i_2]\!]_c^{A'} \cup [\![its]\!]_c^{A'}$$

From the *syntax* predicate, we also have that

$$exp(i_1) = exp(i_2)$$

Since we have that $\neg satisfies(exp(i_1), c)$, we also have that $\neg satisfies(exp(i_2), c)$. CK evaluation for these items then does not yield any assets. Therefore, from the above, we know that we only have to prove

$$[\![its]\!]_c^A = [\![its]\!]_c^{A'}$$

The proof follows by applying Lemma 13, properly instantiated with the variables introduced above and c.

□

Lemma 15. ⟨Configuration knowledge evaluation when item is activated⟩
For product line (F, A, K), asset mapping A', configuration knowledge K', CK items i_1 and i_2, CK items set its, asset names n and n', assets a, a' and a'', and set of mappings m, if

$$syntax(A, K, A', K', i_1, i_2, its, n, n', a, a', a'', m) \wedge conditions(a, a', a'', n, n', its)$$

then

$$\forall c \in [\![F]\!] \cdot satisfies(exp(i_1), c) \Rightarrow$$
$$[\![K]\!]_c^A = \{a\} \cup [\![its]\!]_c^A \wedge [\![K']\!]_c^{A'} = \{a', a''\} \cup [\![its]\!]_c^{A'}$$

Proof: For arbitrary F, A, K, A', K', i_1, i_2, its, n, n', a, a', a'', m, assume the *syntax* and *conditions* predicates previously illustrated (see Predicates 24 and 25). For arbitrary $c \in [\![F]\!]$, we assume $satisfies(exp(i_1), c)$. We have to prove that

$$[\![K]\!]_c^A = \{a\} \cup [\![its]\!]_c^A \wedge [\![K']\!]_c^{A'} = \{a', a''\} \cup [\![its]\!]_c^{A'}$$

From the *syntax* predicate, we have that

$$K = \{i_1\} \cup its \wedge K' = \{i_2\} \cup its$$

Thus, evaluating K consists on evaluating i_1 and its. We do likewise with K'. From the *syntax* predicate, we also have that

$$exp(i_1) = exp(i_2)$$

Since we have that $satisfies(exp(i_1), c)$, we also have that $satisfies(exp(i_2), c)$. Therefore, when evaluating these items, we yield the assets related to them. Again, from the *syntax* predicate, we have

$$names(i_1) = \{n\} \wedge names(i_2) = \{n, n'\}$$

and

$$A = \{n \to a\} \oplus m \wedge A' = \{n \to a', n' \to a''\} \oplus m$$

Using the mappings above, and the fact that the items are successfully evaluated, we obtain

$$[\![K]\!]_c^A = \{a\} \cup [\![its]\!]_c^A \wedge [\![K']\!]_c^{A'} = \{a', a''\} \cup [\![its]\!]_c^{A'}$$

as desired.

□

7.4. Add optional feature

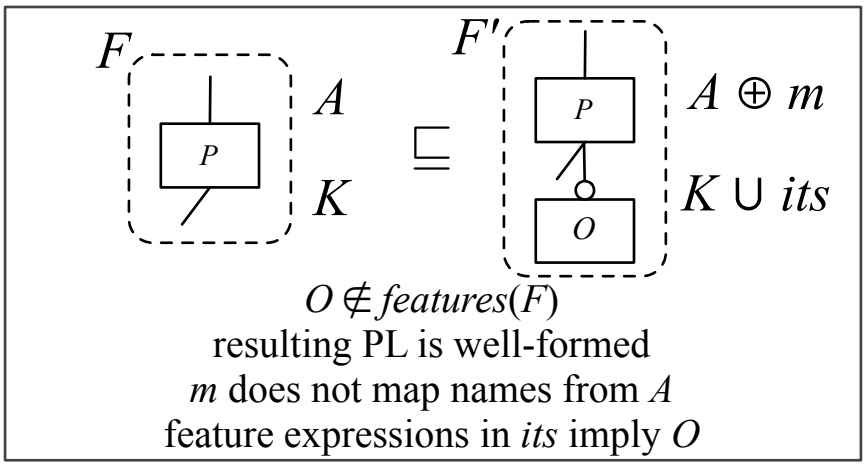

Figure 4: Add optional feature refinement template

$$
\begin{aligned}
&syntax(F, A, K, F', A', K', P, O, its, m) = \\
&\quad features(F') = features(F) \cup \{O\} \wedge \\
&\quad formulae(F') = formulae(F) \cup \{O \Rightarrow P\} \wedge \\
&\quad P \in features(F) \wedge \\
&\quad A' = A \oplus m \wedge \\
&\quad K' = K \cup its
\end{aligned}
\tag{26}
$$

$$
\begin{aligned}
&conditions(F, A, K', F', A', its, O, m) = \\
&\quad wfPL(F', A', K') \wedge \\
&\quad O \notin features(F) \wedge \\
&\quad \forall n \in dom(m) \cdot n \notin dom(A) \wedge \\
&\quad \forall c \cdot \forall exp \in exps(its) \cdot satisfies(exp, c) \Rightarrow satisfies(O, c)
\end{aligned}
\tag{27}
$$

Lemma 16. ⟨Add optional node to feature model⟩
For feature models F, F', and feature names P and O, if

$$
\begin{aligned}
features(F') = features(F) \cup \{O\} \wedge \\
formulae(F') = formulae(F) \cup \{O \Rightarrow P\} \wedge \\
P \in features(F) \ \wedge \ O \notin features(F)
\end{aligned}
$$

then

$$
\forall c \in [\![F]\!] \cdot c \in [\![F']\!] \ \wedge \ wfFM(F')
$$

Proof: For arbitrary F, F', P, O, assume the predicates illustrated above. For the first predicate, we have to prove, for an arbitrary $c \in [\![F]\!]$, that

$$
c \in [\![F']\!]
$$

Considering that *semantics* is the correspondent $[\![\_]\!]$ for the feature model language used (see Definition 1), we have to prove that

$$
c \subseteq features(F') \wedge \forall f \in formulae(F') \cdot satisfies(f, c)
$$

From our assumptions, we have that $features(F') = features(F) \cup \{O\}$. Therefore, since c only contains features from F, we prove the first part of the conjunction. From the same assumptions, we also have that

$$
formulae(F') = formulae(F) \cup \{O \Rightarrow P\}
$$

Since $c \in [\![F]\!]$, we have that all $formulae(F)$ are already satisfied in F. The $O \Rightarrow P$ formula is trivially satisfied in c, since O is not a feature in F, thus $O \notin c$.

For the second predicate, we have to prove that $wfFM(F')$. From Definition 1, this amounts to

$$
\forall f \in formulae(F') \cdot wt(f, F')
$$

From the assumptions, we have that

$$
formulae(F') = formulae(F) \cup \{O \Rightarrow P\}
$$

From the definition of F, we have that $wfFM(F)$, which amounts to

$$
\forall f \in formulae(F) \cdot wt(f, F)
$$

From the above, and the fact that we do not remove any features from F— only adding O, we can prove that all formulae only contain names from the feature model F', since the new formula only references P, which is an existing feature in F, and O, which is the new added feature.

□

Lemma 17. ⟨Configuration knowledge evaluation not affected by new optional feature with guarded items⟩ For product line (F, A, K), feature model F', asset mapping A', CK items $items$, features P and O, if

$$wfPL(F, A, K) \wedge syntax(F, A, K, F', A', K', P, O, its, m) \wedge conditions(F, A, K', F', A', its, O, m)$$

then

$$\forall c \in [\![F]\!] \cdot [\![K]\!]_c^A = [\![K']\!]_c^{A'}$$

Proof: For arbitrary F, A, K, F', A', K', P, O, its, assume $wfPL(F, A, K)$, and the *syntax* and *conditions* predicates previously illustrated (see Predicates 26 and 27). For arbitrary $c \in [\![F]\!]$, we have to prove that

$$[\![K]\!]_c^A = [\![K']\!]_c^{A'}$$

From the *syntax* predicate, we have that

$$K' = K \cup its$$

Thus, by Lemma 6, we have to prove that

$$[\![K]\!]_c^A = [\![K]\!]_c^{A'} \cup [\![its]\!]_c^{A'}$$

Adding new mappings to A does not affect CK evaluation, so

$$[\![K]\!]_c^A = [\![K]\!]_c^{A'}$$

From the *conditions* predicate, we have that

$$O \notin features(F)$$

Therefore, we can deduce that $O \notin c$. Thus, evaluating its against c does not yield any asset, since we have, from the *conditions* predicate, that when feature expressions from its are satisfied in c, O is satisfied also. Therefore, we prove that

$$[\![K]\!]_c^A = [\![K']\!]_c^{A'}$$

□