Proofs of the convergence of the rewriting system for the weaving of aspects in the AO-PFA language

by

Qinglei Zhang and Ridha Khedri
Proofs of the convergence of the rewriting system for the weaving of aspects in the AO-PFA language *

Qinglei Zhang, and Ridha Khedri†
Technical Report CAS-13-01-RK
Department of Computing and Software
McMaster University
April 18, 2013

Abstract

AO-PFA is an aspect oriented feature modelling language. This report discusses several theoretical issues regarding the weaving process of AO-PFA. The weaving of aspects is to extract subterms from product family terms in the base specification, and then to replace the extracted subterm (selected join points) by another term (advice). The problem of extracting a subterm from another term is related to what is known as the word problem, which is undecidable in general.

In this technical report, we give the detailed proofs related to the termination and confluence (i.e., convergence) of the rewriting system for weaving aspects in AO-PFA. We show that the word problem associated to AO-PFA is decidable. We also prove the unambiguosness of the weaving results by putting some restrictions on the kind of the pointcut of the aspects in AO-PFA.

Keywords: Aspect orientation, Feature modelling, AO-PFA, product family algebra, term rewriting systems, termination, confluence

---

*This research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC) through the grant RGPIN227806-09.
†E-mail: {zhangq33, khedri}@mcmaster.ca
# Contents

1 Introduction
   1.1 Problem Statement .................................................. 2

2 Background
   2.1 Aspect-Oriented Feature Modeling Language: AO-PFA ............. 2
      2.1.1 Product Family Algebra ....................................... 2
      2.1.2 Grammar of AO-PFA ........................................... 3
   2.2 Rewriting Systems .................................................... 4
      2.2.1 Basic notations .................................................. 4
      2.2.2 Basic results related to termination .......................... 7
      2.2.3 Basic results related to confluence ........................... 7

3 Convergence of the Weaving Process .................................. 8
   3.1 Concerns Regarding to the Weaving Process of AO-PFA .......... 9
   3.2 Convergence of the Rewriting System ............................ 10
      3.2.1 Proofs of the termination ..................................... 10
      3.2.2 Proofs of the confluence ..................................... 13
   3.3 Unambiguosity of the Weaving Results ........................... 21
      3.3.1 The restriction on the form of the kind pointcut .......... 22

4 Conclusion and Future Work ......................................... 26
1 Introduction

Feature-modelling techniques involve the modelling of the commonality and variability of a set of related products in terms of features. The set of related products is referred to as a product family. The development, maintenance, and evolution of complex and large feature models are among the main challenges faced by feature-modeling practitioners. In response to the challenges for large feature models, we have proposed a feature modeling language called AO-PFA (*aspect-oriented product family algebra*), which contributes to a modular management of feature models.

The language AO-PFA is developed from two basic concepts, product family algebra [5, 6, 7, 8] and the aspect-oriented paradigm [10, 11]. Product family algebra is a formal feature modeling approach that is proposed by Höfner et al. to capture different notations and terms for feature modeling. In [1], we find a thorough discussion on how other feature-modelling techniques can be easily translated into product family algebra. More details on product family algebra is given in Section 2.1.1. Comparing with other feature modeling techniques, techniques based on product family algebra can benefit from the mathematical nature of product family algebra. In particular, the AO-PFA language enable specifying feature models formally and concisely. On the other hand, the aspect-oriented paradigm is an emerging paradigm used to decompose systems by separation of concerns. More specially, the aspect-oriented paradigm takes extra efforts on the encapsulation of crosscutting concerns with a newly introduced construct called “aspect”. Crosscutting concerns, such as security concerns, are inherently scattering over and tangling with other concerns in the system. The development, maintenance and evolution of crosscutting concerns is recognised as tedious and costly in general. Feature modeling techniques are suffering the same problems for feature models due to the existence of crosscutting concerns at feature modeling level. In summary, AO-PFA is a specification language regarding the adaptation of the aspect-oriented paradigm to product family algebra, which can improve the scalability and modifiability of feature models.

With the aspect-oriented paradigm, a system is composed of base concerns and crosscutting concerns. The crosscutting concern is specified as an aspect. The part of a system with only base concerns is called the base system, and the process to compose an aspect with the base system is called the weaving process. The concept of “aspect” was firstly introduced at the programming language level in [10], and has been adapted to the whole software development life-cycle of software development. In this technical report, we concentrate on techniques at the feature modeling level. However, several commonly used terms in the aspect-oriented paradigm are helpful to understand the process of weaving aspects in general. The points in the base system at where aspects can be implicitly invoked are referred to as join points. An aspect specifies a pointcut and an advice. The advice specifies what the aspect is realising, and the pointcut specifies where the aspect should be introduced. In particular, the weaving of aspects is archived by firstly selecting a set of join points with accordance to the pointcut, and then introducing the advice at those selected join points. In order to automate the weaving process, it is essential to formalise the above informal description of the weaving process and show its convergence.
1.1 Problem Statement

There are two types of specifications in AO-PFA, base specifications and aspect specifications. We give a short discussion in Section 2.1.2 on how those essential terms of the aspect-oriented paradigm, such as aspects, join points, pointcut, and advice, can be adapted to the context of product family algebra. With regard to the weaving process, product family terms are basic constructs for both base and aspect specifications in AO-PFA. The weaving of aspects is to extract subterms (with accordance to the pointcut) from product family terms (in the base specification), and then to replace the extracted subterm (selected join points) by another term (advice). The problem of extracting a subterm from another term is related to the word problem, which is to decide the semantic equivalence of two arbitrary terms w.r.t. to a set of equations (i.e., aslo mentioned as the equational theory) [2]. The word problem in general is undecidable. The key to automate the weaving process in AO-PFA is providing a determined procedure for the word problem w.r.t. the semantics of AO-PFA. Same as the way to solve a word problem in general, we induce a rewriting system for our weaving process. Termination and confluence are two of the most important properties that characterise a rewriting system. A rewriting system is convergence iff it is terminating and confluent. The main contribution of this technical report is to prove that the induced rewriting system for AO-PFA is convergent. Moreover, the form of the extracted term (which is decided by the kind pointcut in AO-PFA) is also restricted by the induced rewriting system. With a certain restriction on the form of the kind pointcut, we can ensure the unambiguosity of the weaving results.

In section 2, we introduce the related background knowledge of AO-PFA, and rewriting systems. In Section 3, we formally prove the convergence of the weaving process for AO-PFA. We conclude in Section 4.

2 Background

This technical report is about the weaving process of a specification language AO-PFA. In Section 2.1, we briefly introduce the language AO-PFA. Moreover, the weaving process engenders a particular rewriting system, and we need to analyse properties of the rewriting system to verify the convergence of the weaving process. In Section 2.2, we introduce several basic notations and results of rewriting systems, which are used in the following sections of the report.

2.1 Aspect-Oriented Feature Modeling Language: AO-PFA

2.1.1 Product Family Algebra

Product family algebra extends the mathematical notations of commutative idempotent semirings to describe and manipulate product families.

Definition 1 (e.g., [7]). A product family algebra is a commutative idempotent semiring \((S, +, \cdot, 0, 1)\), where each element of the semiring is a product family. In particular,
operators of the semiring are interpreted as follows:

(a) \( + \) denotes the choice between two product families;
(b) \( \cdot \) denotes a composition of two product families;
(c) \( 0 \) represents an empty product family;
(d) \( 1 \) represents a product family consisting of only a pseudo-product which has no features.

Based on above basic notations of product family algebra, we can analyse properties of product families according to mathematical calculations. In particular, a product family \( a \) is the subfamily of a product family \( b \), denoted by \( a \leq b \), iff \( a + b = b \). A product family \( a \) is the refinement of a product family \( b \), denoted by \( a \sqsubseteq b \), iff \( \exists (c \mid a \leq b \cdot c) \).

Moreover, constraints are elicited in product families, which can be informally described as “If a member of a product family has property \( P_1 \), it must also (not) have property \( P_2 \)” [7]. A requirement relation is defined to capture constraints in the context product family algebra.

**Definition 2** (e.g., [7]). For elements \( a, b, c, d \) and a product \( p \) in product family algebra, the requirement relation \( \rightarrow \) is defined in a family-induction style as follows:

\[ a \overset{p}{\rightarrow} b \Longleftrightarrow df \quad p \sqsubseteq a \quad \text{and} \quad a \overset{c+d}{\rightarrow} b \quad \text{and} \quad a \overset{c}{\rightarrow} b \wedge a \overset{d}{\rightarrow} b. \]

### 2.1.2 Grammar of AO-PFA

The specification language **AO-PFA** (*Aspect-Oriented Product Family Algebra*) is an extension of the specification language based on product family algebra. A tool called Jory has been developed to specify and analyse product families based on product family algebra, and the current specification language used by Jory is called PFA. AO-PFA is an extension of PFA, which is proposed to specify aspects in the context of product family algebra. As mentioned in the introduction, the PFA specifications are used to specify feature models with regard to base concerns, while the aspect specifications are used to specify feature models with regard to crosscutting concerns. The grammar of the PFA specification language and the aspect specification language are given in Figure 1 and Figure 2, respectively.

Briefly, there are three types of syntactic elements in a PFA specification: basic feature declarations, product family definitions, and constraints. A basic feature label preceded by the keyword \( bf \) declares a basic feature. An equation with a product family label at the left-hand side and a product family algebra term at the right-hand side gives the definition of a product family. A triple preceded by the keyword constraint represents a constraint, which specifies a requirement relation as introduced in Definition 2.

An aspect specification includes a sequence of aspects. Join points are unified as product family terms in PFA specifications. The body of the advice is expressed by a general product family term, denoted by \( \text{Advice}(jp) \), which is either a ground term or a term
with a variable $jp$ that represents the instance of a join point. The pointcut of an aspect is to select join points, which in our context is to select product family terms satisfying certain conditions. In particular, we identify three attributes of the product family terms: the position, dynamic characteristic, and form. Correspondingly, the pointcut language in AO-PFA is represented by a triple $(scope, expression, kind)$. In summary, an aspect can be specified as follows:

$$ Aspect\ aspectId = Advice(jp) \quad \text{where} \quad jp \in (scope, expression, kind) $$

We point out that the form of those selected join points is decided by the kind pointcut. We refer the reader to our previous technique report [12] and papers [13, 14] for the specification and verification in AO-PFA.

**2.2 Rewriting Systems**

**2.2.1 Basic notations**

A rewriting system is a set of rewrite rules. A rewrite rule is used to tell how a subterm of a given term can be replaced by another term. Given terms with variable declaration $(t, Y_1)$ and $(t', Y_2)$ and a rewrite rule $(V, t_1 \rightarrow t_2)$, we say $(t', Y_2)$ is derived from $(t, Y_1)$ by application of $(V, t_1 \rightarrow t_2)$, if the two following conditions are satisfied:

1. There is $y_0 \in V$ and $t_0 \in T_f(Y_1 \cup Y_2 \cup \{y_0\})$ such that there is at most one occurrence of $y_0$ in $t_0$, and there is a mapping $\sigma : V \rightarrow T_f(Y_1 \cup Y_2)$, such that $t = \overline{h_1}(t_0)$, and
We sometimes denote the derivation of terms by the substitution of terms. The above (

\( t' = \overline{h}_2(t_0) \) for

\[
\begin{align*}
    h_1(x) &= \begin{cases} 
        \sigma(t_1) & \text{if } x = y_0 \\
        x & \text{otherwise}
    \end{cases} \\
    h_2(x) &= \begin{cases} 
        \sigma(t_2) & \text{if } x = y_0 \\
        x & \text{otherwise}
    \end{cases}
\end{align*}
\]

where, \( \sigma, \overline{h}_1, \) and \( \overline{h}_2 \) denote the extended substitutions of \( \sigma, h_1, \) and \( h_2, \) respectively.

2. \( (V \cup Y_1) \neq \emptyset \implies T_f(Y_2) \neq \emptyset. \)

We sometimes denote the derivation of terms by the substitution of terms. The above derivation \( t' \) can be represented as \( t' = t[\overline{t}_1/\overline{t}_2], \) where \( \overline{t}_1 = \sigma(t_1), \) \( \overline{t}_2 = \sigma(t_2), \) and \( \overline{t}_1 \) is subterm of \( t. \) Given a set of rewrite rules \( W, \) we say that \((t_r, Y_r)\) is derivable from \((t_1, Y_1)\) with rules \( W, \) denoted as \((t_1, Y_1) \xrightarrow{W} (t_r, Y_r)\), if there is a rewrite sequence \((t_1, Y_1), (t_2, Y_2), \ldots, (t_r, Y_r)\) for \( r > 1 \) such that for \( i = 1, 2, \ldots, r - 1, (t_{i+1}, X_{i+1}) \) is derived from \((t_i, X_i)\) by application of some rule in \( W. \)

**Theorem 1** (\[4\] p. 127). Let \( R \) and \( Q \) be an arbitrary sets of rewrite rules then \((t_1, X_1) \xrightarrow{R} (t_2, X_2), \) and \((t_2, X_2) \xrightarrow{Q} (t_3, X_3)\) imply \((t_1, X_1) \xrightarrow{R \cup Q} (t_3, X_3).\)

A rewriting system corresponds to a reduction system in general. Formally, a reduction system is a pair \((A, \rightarrow), \) where \( \rightarrow \) is a binary relation on the set of terms \( A. \) Let the
notation  \( \rightarrow^\ast \) denote the reflexive and transitive closure over \( \rightarrow \), and let  \( \leftarrow^\ast \) denote the reflexive, transitive, and symmetric closure over \( \rightarrow \). In addition to the above notations, some terminology are defined as follows:

1. \( x \) is reducible if there is a \( y \neq x \) such that \( x \rightarrow y \).
2. \( y \) is a normal form of \( x \) if \( x \rightarrow^* y \) and \( y \) is not reducible. If \( x \) has a unique normal form, the normal form is denoted as \( x \downarrow \).
3. \( x \) and \( y \) are joinable if there is a \( z \) such that \( x \rightarrow^* z \) and \( y \rightarrow^* z \), in which we write \( x \downarrow y \).

Several important properties of the reduction relation that are used in our context are given in Definition 3.

**Definition 3** ([2, p. 9]). A reduction relation \( \rightarrow \) is said to be

1. confluent iff \( x \rightarrow^* y_1 \land x \rightarrow^* y_2 \Rightarrow y_1 \downarrow y_2 \).
2. terminating iff every descending chain \( a_0 \rightarrow a_1 \rightarrow \ldots \) is finite.
3. convergent iff it is both confluent and terminating.

The rewriting system is closely related to the equational reasoning problems. We next introduce several notations regarding to the equational theory and the reduction relation.

**Lemma 1** ([3, p. 6]). Suppose \( \Delta \) is a set of formulas and \( \psi \) and \( \rho \) are formulas, then

\( (\Delta \cup \{\psi\}) \models \rho \iff \Delta \models (\rho \models \psi) \)

**Definition 4** ([2, p. 36]). Let \( s \) be a term over the signature \( \Sigma \) and the variable set \( X \).

1. The set of position of the term \( s \), denoted by \( \mathcal{P}os(s) \), is defined as follows:
   - If \( s = x \in X \), then \( \mathcal{P}os(s) = \{\epsilon\} \), where \( \{\epsilon\} \) denotes the empty string.
   - If \( s = f(s_1, \ldots, s_n) \), then
     \[
     \mathcal{P}os(s) = \{\epsilon\} \cup \bigcup_{i=1}^{n} \{ip \mid p \in \mathcal{P}os(s_i)\}
     \]
2. For \( p \in \mathcal{P}os(s) \), the subterm of \( s \) at position \( p \), denoted by \( s|_p \), is defined by induction on \( p \) as follows:
   - \( s|_\epsilon = s \),
   - \( f(s_1, \ldots, s_n)|_q = s_i|_q \).

According to the above definition, for example, the set of position for term \( \cdot(x, +(y, z)) \) is \( \{\epsilon\} \cup \{1 + \mathcal{P}os(x), 2 + \mathcal{P}os(+(y + z))\} = \{\epsilon, 1, 2 + \{\epsilon, 1 + \mathcal{P}os(y), 2 + \mathcal{P}os(z)\}\} = \{\epsilon, 1, 2, 21, 22\} \). On the other hand, if we consider the term \( \cdot(x, +(y, z)) \) then \( \cdot(x, +(y, z))|_{22} = +(y, z)|_{2} = z \).
Definition 5 ([2] p. 38]). Let $\Sigma$ be a signature and $V$ be a countably infinite set of variables. A $T_\Sigma(V)$-substitution, denoted by $\text{Sub}(T_\Sigma(V))$ or simply $\text{Sub}$, is a function $\theta : V \rightarrow T_\Sigma(V)$ such that $\theta(x) \neq x$ for only finitely many $x$s.

Definition 6 ([2] p. 39]). Let $E$ be a set of equations over $\Sigma$. The reduction relation induced by $E$, denoted by $\rightarrow_E \subseteq T_\Sigma(V) \times T_\Sigma(V)$, is defined as follows:

$s \rightarrow_E t$ iff $\exists ((l, r) \in E, p \in \text{Pos}(s), \theta \in \text{Sub} | : (s|_p = \theta(l)) \wedge (t = s(\theta(r)|_p))$)

2.2.2 Basic results related to termination

Definition 7 ([2] p. 118]). Let $>_{\Sigma}$ be a strict order over signature $\Sigma$. The lexicographic path order $>_lpo$ induced by $>$ is defined as follows:

(LOP1) $t \in \text{Var}(s)$ and $s \neq t$.

(LOP2) $s = f(s_1, \ldots, s_m), t = g(t_1, \ldots, t_n)$ and

(LOP2a) $\exists (i \mid 1 \geq i \geq m : s_i \geq t)$

(LOP2b) $f > g \wedge \forall (j \mid 1 \geq j \geq n : s >_{lpo} t_j)$

(LOP2c) $f = g \wedge \forall (j \mid 1 \geq j \geq n : s >_{lpo} t_j)$

$\wedge \exists (i \mid 1 \geq i \geq m : s_1 = t_1, \ldots, s_{i-1} = t_{i-1} \wedge s_i >_{lpo} t_i)$

Theorem 2 ([2] p. 119]). For any strict order $>$ over signature $\Sigma$, the induced lexicographic path order $>_lpo$ is a simplification order on $T_\Sigma(V)$.

Theorem 3 ([2] p. 103]). Let $\Sigma$ be a finite signature. Every simplification order $>$ on $T_\Sigma(V)$ is a reduction order.

Theorem 4 ([2] p. 103]). A term rewriting system $R$ terminates iff there exists a reduction order $>$ that satisfies $l > r$ for all $l \rightarrow r \in R$.

2.2.3 Basic results related to confluence

Given a set of equations $E$, and two terms $s$ and $t$, the process of finding a substitution $\theta$ such that $\theta(s) = \theta(t)$ is known as ”syntactic unification”. We call $\theta$ a unifier of $s$ and $t$, or a solution of the equation $s \equiv t$.

Definition 8 (Most General Unifier[2]). Let $S$ be a set of equation $\{s_1 \equiv t_1, \ldots, s_n \equiv t_n\}$. A substitution $\theta$ is a most general unifier of $S$ iff

- $\theta$ is a solution of $S$, i.e., $\forall (i \mid 1 \leq i \leq n : \theta(s_i) = \theta(t_i))$

- $\theta$ is a least element of all the solution or unifier of $S$, i.e., $\forall (\theta' \mid \theta'$ is a solution of $S: \exists (\delta : \theta' = \delta \theta)$)
Definition 9 ([2] p. 139). Assume that two rules \( l_1 \rightarrow r_1 \) and \( l_2 \rightarrow r_2 \) satisfy \( \text{Var}(l_1, r_1) \cap \text{Var}(l_2, r_2) = \emptyset \). Let \( p \in \text{Pos}(l_1) \) such that \( l_1|_p \) is not a variable and let \( \theta \) be a mgu (most general unification) such that \( \theta(l_1|_p) = \theta(l_2) \). This determines a critical pair \( \langle \theta(r_1), \theta(l_1[\theta(r_2)]|_p) \rangle \).

If two rules give rise to a critical pair, we say that they overlap. When considering the overlap of a rule with a renamed variant of itself, it is safe to ignore the case where \( p = \epsilon \).

Lemma 2 ([2] p. 76). Let \( s_1, \ldots, s_m \), and \( t_1, \ldots, t_n \) be terms over the signature \( \Sigma \). An equation \( f(s_1, \ldots, s_m) = g(t_1, \ldots t_n) \), where \( f, g \in \Sigma, f \neq g \), has no solution.

Theorem 5 ([2] p. 140). A terminating term rewriting system is confluent iff all its critical pairs are joinable.

3 Convergence of the Weaving Process

As mentioned in the introduction, the weaving of aspects in AO-PFA requires to solve a word problem w.r.t. the equational theory that is determined by the aspect specifications and the base specifications. In particular, we consider the union of two sets of equations. One is the set of equations regarding to axioms of a commutative idempotent semiring, which is denoted by \( E_f \) (see Table 1). The other set of equations, denoted by \( E_{\text{spec}} \), are derived from those family definitions given in a particular PFA specification (see Section 2.1.2 for the basic constructs in PFA specifications.).

Table 1: Equational Theory \( E_f \)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Associativity of +: ({x, y, z}, (x + y) + z, x + (y + z));</td>
<td></td>
</tr>
<tr>
<td>2. Symmetry of +: ({x, y}, x + y, y + x);</td>
<td></td>
</tr>
<tr>
<td>3. Identity of +: ({x}, x + 0, x);</td>
<td></td>
</tr>
<tr>
<td>4. Idempotent of +: ({x}, x + x, x);</td>
<td></td>
</tr>
<tr>
<td>5. Associativity of ( \cdot ) ({x, y, z}, (x \cdot y) \cdot z, x \cdot (y \cdot z));</td>
<td></td>
</tr>
<tr>
<td>6. Symmetry of ( \cdot ) ({x, y}, x \cdot y, y \cdot x);</td>
<td></td>
</tr>
<tr>
<td>7. Identity of ( \cdot ) ({x}, x \cdot 1, x);</td>
<td></td>
</tr>
<tr>
<td>8. Distributivity: ({x, y, z}, x \cdot (y + z), x \cdot y + x \cdot z);</td>
<td></td>
</tr>
<tr>
<td>9. Annihilator: ({x}, x \cdot 0, 0);</td>
<td></td>
</tr>
</tbody>
</table>

In [2], it indicates that the semantical equivalent, denoted by \( = \) from a theory coincides with the syntactical equivalence defined over the term rewire rules introduced by the theory, i.e., a set of equations \( E \). Formally, \( E \vdash t = t' \) iff \( R(E) \vdash (t = t') \)\(^1\), which can be written

\(^1\)We use \( = \) to denote both the semantical equivalence and syntactical equivalence of two terms.
as $t \rightarrow t'$. Generally, for an equation $(X, L, R)$, two rewrite rules, $(X, L \rightarrow R)$ or $(X, R \rightarrow L)$, can be induced. We call $(X, L \rightarrow R)$ the L-R-rule of $(X, L, R)$, and $(X, R \rightarrow L)$ the R-L-rule of $(X, L, R)$. We present the introduced rewriting systems w.r.t. the weaving process of AO-PFA.

Some of the rewrite rules of $R(E_f)$ are defined in Table 2. The numbering at the beginning of each line illustrates its corresponding equations in the Table 1. For equations

<table>
<thead>
<tr>
<th>#</th>
<th>Rule Type</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>r3</td>
<td>L-R</td>
<td>${x}, + (x, 0) \rightarrow x$</td>
</tr>
<tr>
<td>r4</td>
<td>L-R</td>
<td>${x}, + (x, x) \rightarrow x$</td>
</tr>
<tr>
<td>r7</td>
<td>L-R</td>
<td>${x}, \cdot (x, 1) \rightarrow x$</td>
</tr>
<tr>
<td>r8</td>
<td>L-R</td>
<td>${x, y, z}, \cdot (x, + (y, z))$ $\rightarrow + (\cdot (x, y), \cdot (x, z))$</td>
</tr>
<tr>
<td>r9</td>
<td>L-R</td>
<td>${x}, \cdot (x, 0) \rightarrow 0$</td>
</tr>
</tbody>
</table>

in $E_{\text{spec}}$ we always get L-R-rules i.e.,

$$R(E_{\text{spec}}) \overset{\text{def}}{=} \{(X, L \rightarrow R) \mid (X, L, R) \in E_{\text{spec}}\}. \quad (1)$$

According to Theorem 1, if $(t_1, Y_1) \underset{R(E_f)}{\rightarrow} (t_2, Y_2)$, and $(t_2, Y_2) \underset{R(E_{\text{spec}})}{\rightarrow} (t_3, Y_3)$, imply $(t_1, Y_1) \underset{R(E_f) \cup R(E_{\text{spec}})}{\rightarrow} (t_3, Y_3)$. In other words, $R(E_f) \cup R(E_{\text{spec}})$ is the induced rewriting system in our context.

### 3.1 Concerns Regarding to the Weaving Process of AO-PFA

Before giving the proofs, we start with a discussion of two issues that require attentions for the weaving process of AO-PFA. Firstly, it is necessary that we have a decision procedure for the word problem to implement weaving process. Then one of the main problem becomes how can we obtain a convergent AO-PFA rewriting system. Let $Th_{pfa}$ be the set of equations introduced by the base and aspect specifications. We state the above concern as follows:

**Concern 1.** To implement the weaving process, we need a decidable procedure for the word problem $Th_{pfa} \models (s = t)$ where $s, t \in T_5(V)$.

Moreover, we match the join points by extracting equivalent subterms from product family terms in the PFA specification. A join point term $e$ is matched within a given product family term $\alpha$ iff $\alpha$ can be represented as an equivalent term $\beta \cdot e + \gamma$, where $\beta$ and $\gamma$ are product family algebra terms. Additionally, $e$ cannot be further extracted from $\beta$.
and $\gamma$. Then the weaving of an aspect is processed by substituting the term of join point by the term of the advice. Obviously, terms $\beta$ and $\gamma$ should be unique w.r.t. the given $\alpha$ and $e$. Otherwise, substituting join points by an advice will cause ambiguous weaving results. However, it is easy to show that, if the term $e$ and $\alpha$ are in arbitrary forms, the term $\beta$ and $\gamma$ cannot be always uniquely decided w.r.t. the equational theory $E_f$. For example, consider a product family $\alpha$ as $f_2^1 + f_1 \cdot f_2 + f_2^2$, and a join point term $e$ as $f_1^1 + f_2$. With accordance to the equations (5), (6), and (8) in $E_f$, we possibly have two different equivalent terms of $\alpha$, either $f_2^1 + f_2 \cdot (f_1 + f_2)$, or $f_1 \cdot (f_1 + f_2) + f_2^2$. Consequently, a crosscutting concern which removes a variation point $f_1 + f_2$, will cause ambiguous weaving results. Although composing such an aspect would not always cause problems, the possible weaving ambiguity of such crosscutting concern impedes the reusability of the aspect. Therefore, instead of restricting the form of $\alpha$, the form of selected join points should be constrained to avoid ambiguous weaving results. We state the above concern as follows:

**Concern 2.** To avoid ambiguous weaving results, the join point $e$ should be restricted to a form that for an arbitrary term $\alpha$, we have unique $\beta$ and $\gamma$ up to the given theory $Th_{pfa}$ such that $\alpha = \beta \cdot e + \gamma$, where $e$ cannot be further extracted from $\beta$ and $\gamma$. We note that $\beta$ and $\gamma$ can be the constant 0. For any term $e$, we assume that it cannot be extracted from 0.

In the following sections, we discuss our solution to satisfy the above two concerns.

### 3.2 Convergence of the Rewriting System

An important theorem regarding to the decidability of the word problem is stated as follows:

**Theorem 6** ([2, p. 59]). Let $E$ be a set of equations. If $E$ is finite and the induced rewriting system $R(E)$ is convergent (i.e., terminating and confluent), then $=_{E}$ is decidable.

We have introduced that the rewriting system for the weaving process in AO-PFA is $R(E_f) \cup R(E_{spec})$. With regard to Concern 1, there is a decidable procedure to the word problem if we can prove that this term rewriting system is confluent and terminating.

#### 3.2.1 Proofs of the termination

In the definition and following related proofs, the sign "\(=\)" means the syntactical equivalence, and $\neq$ is the abbreviation for $\neg(t = s)$. With respect to a given PFA specification $S$, let $\mathcal{L}(S)$ denotes the set of labels of basic features and families given in the specification. The signature of a term rewriting system $R(E_f) \cup R(E_{spec})$ is over a finite signature $\{\{f\}, \{+, \cdot, 0, 1\} \cup \mathcal{L}(S)\}$. To show the termination of the rewriting system, we first define a strict order over its signature as follows:

**Convention 1.** Let $f_1, \ldots, f_n, F_1, \ldots, F_m$ be the declare and definition order specified in a PFA specification $S$, where $f_i(1 \leq i \leq n) \in \mathcal{L}(S)$ and $F_i(1 \leq i \leq m) \in \mathcal{L}(S)$, respectively
denote the labels of basic features and families defined by the specification. We define a strict order over \( \{\{f\},\{+,-,0,1\} \cup \mathcal{L}(S)\} \) as follows:

\[
F_m > \cdots > F_1 > f_n > \cdots > f_1 > + > + > 1 > 0.
\]

Due to the requirements we put on the syntax of PFA specifications, it is obvious that we can always obtain a strict order over the signature of any rewriting system \( R(E_f) \cup R(E_{\text{spec}}) \) according to the above convention.

**Lemma 3.** We have \((l >_{\text{lop}} r)\) for the rules \((\{x\},+(x,0) \rightarrow x), (\{x\},+(x,x) \rightarrow x), \) and \((\{x\},\cdot(x,1) \rightarrow x)\).

**Proof.** For the sake of clarity, we divide our proof into 3 parts as follows:

1. We first prove \((\cdot(x,+(y,z))) >_{\text{lop}} x, y, z, \) and \((+(y,z)) >_{\text{lop}} y, z.\)

\[
(+(y,z)) >_{\text{lop}} y, z, \) and \((\cdot(x,+(y,z))) >_{\text{lop}} x, y, z
\]

\[
\iff \langle \text{Definition of } >_{\text{lop}}: \text{LOP1. Particularly, } x \in \text{Var}(+(x,0)) \land x \neq +(x,0),
\]

\[
x \in \text{Var}(+(x,x)) \land x \neq +(x,x), \) and \(x \in \text{Var}(\cdot(x,1)) \land x \neq \cdot(x,1) \rangle
\]

2. We prove \((\cdot(x,+(y,z))) >_{\text{lop}} \cdot(x,y)\) and \((\cdot(x,+(y,z))) >_{\text{lop}} \cdot(x, z).\)

**Proof (1):** \((\cdot(x,+(y,z))) >_{\text{lop}} x) \land (\cdot(x,+(y,z))) >_{\text{lop}} y) \land (+(y,z)) >_{\text{lop}} y)

\[
\iff \langle \text{Definition of } >_{\text{lop}}: \text{LOP2c. In particular, let } f = g = \cdot, s = (x,+(y,z)),
\]

\[
s_1 = x, s_2 = +(y,z), t = \cdot(x,y), t_1 = x, t_2 = y, \) and \(n = m = 2. \}
\]

Since the proof for \((\cdot(x,+(y,z))) >_{\text{lop}} \cdot(x,z)\) is quite similar with the above proof, we omit the detail here.

3. Finally, we prove \((\cdot(x,+(y,z))) >_{\text{lop}} +(\cdot(x,y),\cdot(x,z)).\)

**Proof (2):** \((\cdot(x,+(y,z))) >_{\text{lop}} \cdot(x,y)) \land (\cdot(x,+(y,z))) >_{\text{lop}} \cdot(x,z))

\[
\iff \langle \text{Definition of } >_{\text{lop}}: \text{LOP2b In particular, let } f = \cdot > g = +(x,+(y,z)) = t_1 = \cdot(x,y), t_2 = \cdot(x,z), \) and \(n = 2. \}
\]

\((x,+(y,z)) >_{\text{lop}} +(\cdot(x,y),\cdot(x,z)).\)
Lemma 5. We have \((l >_{\text{lop}} r)\) for the rewriting rule \((\{x\}, (x, 0) \rightarrow 0)\).

Proof.

\[ (+ > 0) \land (0 \geq 0) \]
\[ \iff \langle \text{Definition of } >_{t_{\text{tpo}}} \text{: LOP2a. In particularly, } f = +, g = 0, s = +(x, 0), s_1 = x, s_2 = 0, t = 0, \text{ and } m = 2. \rangle \]
\[ +(x, 0) >_{\text{lop}} 0 \]

Lemma 6. We have \((l >_{\text{lop}} r)\) for any rule in \(R(E_{\text{spec}})\).

Proof.

\[ \text{true} \iff \langle \text{The syntactical requirements of PFA specification} \rangle \]
\[ \forall ((X, L, R) \in Eq(S) \mid \exists (0 \leq i \leq m \mid L = F_i \land \text{Val}(R) = X - \{F_i\}) \land \text{Val}(r) \subseteq \{f_1, \ldots, f_n, F_1, \ldots, F_{i-1}\}) \rangle \]
\[ \iff \langle \text{Definition of } R(E_{\text{spec}}) \text{: Expression (1)} \rangle \]
\[ \forall ((X, l \rightarrow r) \in R(E_{\text{spec}}) \mid \exists (0 \leq i \leq m \mid l = F_i \land \text{Val}(r) = X - \{F_i\}) \land \text{Val}(r) \subseteq \{f_1, \ldots, f_n, F_1, \ldots, F_{i-1}\}) \rangle \]
\[ \iff \langle \text{Use the Definition of } >_{t_{\text{tpo}}} \text{: LOP2b recursively. In particular, } s = F_i >^*, +, \text{ and } (F_i > f_1) \land \cdots \land (F_i > f_n) \land (F_i > F_1) \land \cdots \land (F_i > F_{i-1}) \land (F_i > 1) \land (F_i > 0) \rangle \]
\[ \forall ((X, L \rightarrow R) \in R(E_{\text{spec}}) \mid l >_{\text{lop}} r) \]

Theorem 7. For any syntactically correct base specification \(S\) and aspect specification \(A\), the rewriting system \(R(E_f) \cup R(E_{\text{spec}})\) is terminating.

Proof.

The rewriting system \(R(E_f) \cup R(E_{\text{spec}})\) is given by Table 2 and Expression 1.
\[ \iff \langle \text{Lemmas 3, 4} \rangle \]
\[ \forall ((X, l \rightarrow r) \mid (X, l \rightarrow r) \in R(E_f) \cup R(E_{\text{spec}}) : l >_{\text{lop}} r) \]
\[ \iff \langle \text{Theorem 2 and Theorem 3 which indicate } >_{\text{lop}} \text{ is a reduction order} \rangle \]

The rewriting system \(R(E_f) \cup R(E_{\text{spec}})\) is terminating.

\[ \square \]
3.2.2 Proofs of the confluence

According to the definition of critical pairs, we rename the variable to ensure that \( \text{Var}(l_1, r_1) \cap \text{Var}(l_2, r_2) = \emptyset \) in the following proofs. In particular, variables in the rewriting rule that corresponds to \( l_1 \to r_1 \) keep the same, while variables in the rewriting rule that corresponds to \( l_2 \to r_2 \) are renamed by adding the prime symbol. For example, \( x \) becomes \( x' \).

**Lemma 7.** Let \( U = \{ (a, b) \mid a, b \in R(E_f) \cup R(E_{\text{spec}}) \} \). The union of the following subset of \( U \) is \( U \).

\[
\begin{align*}
S_1 & : \text{Any two distinct rewriting rules from } R(E_{\text{spec}}) \\
S_2 & : \text{Any one rewriting rule from } R(E_{\text{spec}}), \text{ and any one rule in } R(E_f) \\
S_3 & : \text{Any one rewriting rule of } r_3 \text{ and } r_4, \text{ and any rule of } r_7 \text{ and } r_9. \\
S_4 & : \text{The rule } r_3 \text{ and the rule } r_4. \\
S_5 & : \text{The rule } r_7 \text{ and the rule } r_9. \\
S_6 & : \text{The rule } r_8 \text{ and any one rule of } r_7 \text{ and } r_9. \\
S_7 & : \text{The rule } r_8 \text{ and the rule } r_3. \\
S_8 & : \text{The rule } r_8 \text{ and the rule } r_4. \\
S_9 & : \text{Any one rule of } R(E_f) \cup R(E_{\text{spec}}) \text{ with itself.}
\end{align*}
\]

**Proof.**

\[
\begin{align*}
& S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 \cup S_6 \cup S_7 \cup S_8 \cup S_9 \\
\iff & \langle \text{Definition of } S_6, S_7, S_8, \text{ and } E_f \rangle \quad \& \quad \text{idempotent, commutativity and associativity of } \cup \\
& (S_1 \cup S_9) \cup S_2 \cup (S_3 \cup S_4 \cup S_5 \cup S_9) \\
& \cup \{ (a, b) \mid (a \in \{r_8\} \land b \in E_f \setminus \{r_8\}) \lor (a \in E_f \setminus \{r_8\} \land b \in \{r_8\}) \} \\
\iff & \langle \text{Definition of } S_3, S_4, S_5, S_9, \text{ and } E_f \rangle \\
& (S_1 \cup S_9) \cup S_2 \cup \{ (a, b) \mid a \in E_f \setminus \{r_8\} \land b \in E_f \setminus \{r_8\} \} \\
& \cup \{ (a, b) \mid (a \in \{r_8\} \land b \in E_f \setminus \{r_8\}) \lor (a \in E_f \setminus \{r_8\} \land b \in \{r_8\}) \} \\
& \cup \{ (a, b) \mid (a \in \{r_8\} \land b \in \{r_8\}) \} \\
\iff & \langle \text{Definition of set union} \quad \& \quad \text{Definition of } E_f \rangle \\
& (S_1 \cup S_9) \cup S_2 \cup \{ (a, b) \mid a, b \in E_f \} \\
\iff & \langle \text{Definitions of } S_1, S_2 \text{ and } S_9 \rangle \\
& \{ (a, b) \mid a, b \in Eq \} \cup \{ (a, b) \mid (a \in Eq \land b \in E_f) \lor (a \in E_f \land b \in Eq) \} \\
& \cup \{ (a, b) \mid a, b \in E_f \} \\
\iff & \langle \text{Definition of set union} \rangle \\
& \{ (a, b) \mid a, b \in Eq \cup E_f \}
\end{align*}
\]
Lemma 8. Any two distinct rewriting rules in $R(E_{\text{spec}})$ cannot give raise of a critical pair.

Proof. For the case of redundant specification, (i.e., two rewrite rules generated by equations of $E_{\text{spec}}$ are the same), the possibility of giving raise of critical pair is the same as the case for the set of $S_0$. We will discuss this case later. In the following proof, we assume that all rewrite rules in $R(E_{\text{spec}})$ are different. According to the definition of $R(E_{\text{spec}})$, we have:

\[
\forall (l_1 \rightarrow r_1, l_2 \rightarrow r_2 \in R(E_{\text{spec}}) \mid l_1 \neq l_2 : l_1, l_2 \in L(S))
\]

\[\iff \quad \langle \text{Lemma 2, In particular, there is no mgu for two distinct constants} \rangle\]

\[
\forall (l_1 \rightarrow r_1, l_2 \rightarrow r_2 \in R(E_{\text{spec}}) \mid l_1 \neq l_2 : \neg \exists (\theta, p : \theta(l_1|_p) = \theta(l_2))
\]

\[\iff \quad \langle \text{Definition 9 of critical pair} \rangle\]

\[
\forall (l_1 \rightarrow r_1, l_2 \rightarrow r_2 \in R(E_{\text{spec}}) \mid l_1 \neq l_2 : l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2
\]

cannot give raise of a critical pair\)

Lemma 9. Any one rewriting rule in $R(E_{\text{spec}})$ with any one rule in $R(E_f)$ cannot give raise of a critical pair.

Proof. To show that any one rewriting rule in $R(E_{\text{spec}})$ with any one rule in $R(E_f)$ cannot give raise a critical pair, we divide proof into two parts.

(1) In the first part, we assume $l_1 \rightarrow r_1 \in R(E_{\text{spec}})$ and $l_2 \rightarrow r_2 \in R(E_f)$. According to the definition of $R(E_{\text{spec}})$ and $R(E_f)$, we have:

\[
\forall (l_1, l_2 \mid l_1 \rightarrow r_1 \in R(E_{\text{spec}}), l_2 \rightarrow r_2 \in R(E_f)
\]

\[
: l_1 \in L(S) \land l_2 \text{ is a non-variable and non-constant term}
\]

\[\iff \quad \langle \text{Lemma 2, In particular, there is no mgu for a constant with a non-variable and non-constant term} \rangle\]

\[
\forall (l_1, l_2 \mid l_1 \rightarrow r_1 \in R(E_{\text{spec}}) \land l_2 \rightarrow r_2 \in R(E_f) : \neg \exists (\theta, p : \theta(l_1|_p) = \theta(l_2))
\]

\[\iff \quad \langle \text{Definition 9 of critical pair} \rangle\]

\[
\forall (l_1, l_2 \mid l_1 \rightarrow r_1 \in R(E_{\text{spec}}) \land l_2 \rightarrow r_2 \in R(E_f) : l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2
\]

cannot give raise of critical pair\)

(2) In the second part, we assume $l_1 \rightarrow r_1 \in R(E_f)$ and $l_2 \rightarrow r_2 \in R(E_{\text{spec}})$. According to the definition of $R(E_{\text{spec}})$ and $R(E_f)$, we have:

\[
\forall (l_1, l_2 \mid l_1 \rightarrow r_1 \in R(E_f) \land l_2 \rightarrow r_2 \in R(E_{\text{spec}}) : l_2 \in L(S) \land \text{ the non-variable subterm of } l_1 \text{ is either a constant } 0 \text{ or } 1, \text{ or } l_1 \text{ itself}
\]

\[\iff \quad \langle \text{Lemma 2, In particular, there is no mgu for either two distinct constants, or a constant with a non-variable and non-constant term} \rangle\]

\[
\forall (l_1, l_2 \mid l_1 \rightarrow r_1 \in R(E_f) \land l_2 \rightarrow r_2 \in R(E_{\text{spec}}) : \neg \exists (\theta, p : \theta(l_1|_p))
\]

\[\]
Lemma 10. Any rewriting rule among $r_3$ and $r_4$, and any rewriting rule among $r_7$ and $r_9$ cannot give raise of a critical pair.

Proof. To show that any one rewriting rule of $r_3$ and $r_4$, and any one rewriting rule of $r_7$ and $r_9$ cannot give raise of a critical pair, we divide the proof into two parts.

1. In the first part, we assume that $l_1 \rightarrow r_1$ is either $r_3$ or $r_4$, while $l_2 \rightarrow r_2$ is either $r_7$ or $r_9$. According to the definitions of $r_3$, $r_4$, $r_7$, and $r_9$, we have

$$
\forall(l_1, l_2) \mid l_1 \rightarrow r_1 \in \{r_3, r_4\} \land l_2 \rightarrow r_2 \in \{r_7, r_9\} : \text{the non-variable subterm of } l_1 \text{ is either a constant 0 or 1, or } l_1 \text{ itself, which is in the form of } +(t_1, t_2) \land l_2 \text{ is a term in the form of } -(t'_1, t'_2) \\
\iff \langle \text{Lemma 2} \rangle \text{ In particular, there is no mgu for a constant with a non-variable and non-constant term, and there is no mgu for two terms in the form of } +(t_1, t_2) \land -(t'_1, t'_2) \\
\forall(l_1, l_2) \mid l_1 \rightarrow r_1 \in \{r_3, r_4\} \land l_2 \rightarrow r_2 \in \{r_7, r_9\} : \neg \exists(\theta, p) \mid \theta(l_1|_p) \text{ is not a variable : } \theta(l_1|_p) = \theta(l_2) \\
\iff \langle \text{Definition 9 of critical pair} \rangle \\
\forall(l_1, l_2) \mid l_1 \rightarrow r_1 \in \{r_3, r_4\} \land l_2 \rightarrow r_2 \in \{r_7, r_9\} : l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ cannot give raise of a critical pair} \]

2. In the second part, we assume that $l_1 \rightarrow r_1$ is either $r_7$ or $r_9$, while $l_2 \rightarrow r_2$ is either $r_3$ or $r_4$. According to the definitions of $r_3$, $r_4$, $r_7$, and $r_9$, we have

$$
\forall(l_1, l_2) \mid l_1 \rightarrow r_1 \in \{r_7, r_9\} \land l_2 \rightarrow r_2 \in \{r_3, r_4\} : \text{the non-variable subterm of } l_1 \text{ is either a constant 0 or 1, or } l_1 \text{ itself, which is in the form of } -(t_1, t_2) \land l_2 \text{ is a term in the form of } +(t'_1, t'_2) \\
\iff \langle \text{Lemma 2} \rangle \text{ In particular, there is no mgu for a constant with a non-variable and non-constant term, and there is no mgu for two terms in the form of } +(t_1, t_2) \land -(t'_1, t'_2) \\
\forall(l_1, l_2) \mid l_1 \rightarrow r_1 \in \{r_7, r_9\} \land l_2 \rightarrow r_2 \in \{r_3, r_4\} : \neg \exists(\theta, p) \mid \theta(l_1|_p) \text{ is not a variable : } \theta(l_1|_p) = \theta(l_2) \\
\iff \langle \text{Definition 9 of critical pair} \rangle \\
\forall(l_1, l_2) \mid l_1 \rightarrow r_1 \in \{r_7, r_9\} \land l_2 \rightarrow r_2 \in \{r_3, r_4\} : l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ cannot give raise of a critical pair} \]

\qed
Lemma 11. The critical pairs determined by the rewriting rule $r_3$ and $r_4$ are joinable.

Proof. The rules $r_3$ and $r_4$ can give raise of two critical pairs. We respectively show that both critical pairs are joinable.

(1) Assume that $r_3$ is $l_1 \rightarrow r_1$ and $r_4$ is $l_2 \rightarrow r_2$. According definitions of $r_3$ and $r_4$, we have:

\[
\begin{align*}
  l_1 \rightarrow r_1 & \in \{r_3\} \land l_2 \rightarrow r_2 \in \{r_4\} \implies \text{the non-variable subterm of } l_1 \\
  & \text{is either } 0, \text{ or } +(x, 0) \land l_2 = +(x', 0) \\
\end{align*}
\]

\[
\begin{align*}
  \langle \text{Definition of mgu. In particular, } \theta = \{x \mapsto x', x \mapsto 0\} \rangle \\
  l_1 \rightarrow r_1 & \in \{r_3\} \land l_2 \rightarrow r_2 \in \{r_4\} \implies \exists(\theta, p \mid \theta(l_1)|_p = \theta(l_2) = 0 + 0) \\
\end{align*}
\]

\[
\begin{align*}
  \iff \langle \text{Definition } \theta \rangle \text{ of critical pair } & \land \theta = \{y \mapsto x', z \mapsto 0\} \\
  l_1 \rightarrow r_1 & \in \{r_3\} \land l_2 \rightarrow r_2 \in \{r_4\} \implies l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ give raise of a critical pair } (x', x') \\
\end{align*}
\]

\[
\begin{align*}
  l_1 \rightarrow r_1 & \in \{r_3\} \land l_2 \rightarrow r_2 \in \{r_4\} \implies l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ give raise of a joinable critical pair.}
\end{align*}
\]

(2) Assume that $r_4$ is $l_1 \rightarrow r_1$ and $r_3$ is $l_2 \rightarrow r_2$. According definitions of $r_3$ and $r_4$, we have:

\[
\begin{align*}
  l_1 \rightarrow r_1 & \in \{r_4\} \land l_2 \rightarrow r_2 \in \{r_3\} \implies \text{the non-variable subterm of } l_1 \\
  & \text{is } +(x, x) \land l_2 = +(x', 0) \\
\end{align*}
\]

\[
\begin{align*}
  \langle \text{Definition of mgu. In particular, } \theta = \{x' \mapsto x, x \mapsto 0\} \rangle \\
  l_1 \rightarrow r_1 & \in \{r_4\} \land l_2 \rightarrow r_2 \in \{r_3\} \implies \exists(\theta, p \mid \theta(l_1)|_p = \theta(l_2) = +(0, 0)) \\
\end{align*}
\]

\[
\begin{align*}
  \iff \langle \text{Definition of critical pair } \theta \rangle & \land \theta = \{x \mapsto x, x \mapsto 0\} \\
  l_1 \rightarrow r_1 & \in \{r_4\} \land l_2 \rightarrow r_2 \in \{r_3\} \implies l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ give raise of a critical pair } (0, 0) \\
\end{align*}
\]

\[
\begin{align*}
  l_1 \rightarrow r_1 & \in \{r_4\} \land l_2 \rightarrow r_2 \in \{r_3\} \implies l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ give raise of a joinable critical pair.}
\end{align*}
\]

Lemma 12. The rewriting rule $r_7$ and $r_9$ cannot give raise of a critical pair.

Proof. We also consider the proof in two cases.

(1) Assume that $r_7$ is $l_1 \rightarrow r_1$ and $r_9$ is $l_2 \rightarrow r_2$. According definitions of $r_7$ and $r_9$, we have:

\[
\begin{align*}
  l_1 \rightarrow r_1 & \in \{r_7\} \land l_2 \rightarrow r_2 \in \{r_9\} \implies \text{the non-variable subterm of } l_1 \\
  & \text{is either } 1, \text{ or } \cdot(x, 1) \land l_2 = \cdot(x', 0)
\end{align*}
\]
Lemma 13. The rewriting rule \( r_8 \) cannot give raise of critical pairs with \( r_7 \) and \( r_9 \).

Proof. We also consider the proof in two cases.

(1) Assume that either \( r_7 \) or \( r_9 \) is \( l_1 \to r_1 \), and \( r_8 \) is \( l_2 \to r_2 \). According definitions of \( r_7 \), \( r_8 \), and \( r_9 \), we have:

\[
\forall (l_1, l_2) \mid l_1 \to r_1 \in \{r_7, r_9\} \land l_2 \to r_2 \in \{r_8\} : \quad \text{the non-variable strict subterm of } l_1 \text{ is a constant } \land l_2 \text{ is a non-variable term}
\]

\[
\iff \quad \langle \text{Lemma 2} \rangle \quad \text{In particular, there is no unification for a constant and a non-variable term.}
\]

\[
\forall (l_1, l_2) \mid l_1 \to r_1 \in \{r_7, r_9\} \land l_2 \to r_2 \in \{r_8\} : \quad \bar{\exists}(\theta, p) : \theta(l_1)|_p = \theta(l_2)
\]

\[
\iff \quad \langle \text{Definition 9 of critical pair} \rangle
\]

\[
\forall (l_1, l_2) \mid l_1 \to r_1 \in \{r_7, r_9\} \land l_2 \to r_2 \in \{r_8\} : \quad l_1 \to r_1 \land l_2 \to r_2 \text{ cannot give raise of a critical pair}
\]

(2) Assume that \( r_8 \) is \( l_1 \to r_1 \), and either \( r_7 \) or \( r_9 \) is \( l_2 \to r_2 \). According definitions of \( r_7 \), \( r_8 \), and \( r_9 \), we have:

\[
\forall (l_1, l_2) \mid l_1 \to r_1 \in \{r_8\} \land l_2 \to r_2 \in \{r_7, r_9\} : \quad \text{the non-variable strict subterm of } l_1 \text{ is a term in the form of } + (t_1, t_2) \land l_2 \text{ is a term}
\]

\[
\iff \quad \langle \text{Lemma 2} \rangle \quad \text{In particular, there is no unifier for two distinct constants, and there is no unifier for a constants with a non-variable and non-constant term.}
\]

\[
l_1 \to r_1 \in \{r_7\} \land l_2 \to r_2 \in \{r_9\} \implies \bar{\exists}(\theta, p) : \theta(l_1)|_p = \theta(l_2)
\]

\[
\iff \quad \langle \text{Definition 9 of critical pair} \rangle
\]

\[
l_1 \to r_1 \in \{r_7\} \land l_2 \to r_2 \in \{r_9\} \implies l_1 \to r_1 \land l_2 \to r_2 \text{ cannot give raise of a critical pair}
\]
in the form of \((t'_1, t'_2)\)

\[ \iff \quad \langle \text{Lemma 2} \rangle \quad \text{in particular, there is no unification for two term } + (t_1, t_2) \text{ and } + (t'_1, t'_2) \]

\[ \forall (l_1, l_2) \mid l_1 \rightarrow r_1 \in \{r8\} \land l_2 \rightarrow r_2 \in \{r7, r9\} : \exists (\theta, p \mid \theta(l_1)p = \theta(l_2)) \]

\[ \iff \quad \langle \text{Definition 9 of critical pair} \rangle \]

\[ \forall (l_1, l_2) \mid l_1 \rightarrow r_1 \in \{r8\} \land l_2 \rightarrow r_2 \in \{r7, r9\} : l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ cannot give raise of a critical pair} \]

\[ \square \]

**Lemma 14.** The rewriting rule \(r8\) and \(r3\) only give raise of one joinable critical pair.

**Proof.** We give the proof by two cases.

1. Assume that \(r3\) is \(l_1 \rightarrow r_1\), and \(r8\) is \(l_2 \rightarrow r_2\). According to definitions of \(r8\) and \(r3\), we have:

\[ l_1 \rightarrow r_1 \in \{r3\} \land l_2 \rightarrow r_2 \in \{r8\} \iff \text{there is no non-variable strict subterm for } l_1 \]

\[ \iff \quad \langle \text{Definition of critical pair} 9 \rangle \]

\[ l_1 \rightarrow r_1 \in \{r3\} \land l_2 \rightarrow r_2 \in \{r8\} \iff l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ cannot give raise of a critical pair} \]

2. Assume that \(r8\) is \(l_1 \rightarrow r_1\), and \(r3\) is \(l_2 \rightarrow r_2\). According to definitions of \(r8\) and \(r3\), we have:

\[ l_1 \rightarrow r_1 \in \{r8\} \land l_2 \rightarrow r_2 \in \{r3\} \iff \text{the non-variable strict subterm of } l_1 \text{ is } + (y, z) \land l_2 = + (x', 0) \]

\[ \langle \text{Definition of mgu. In particular, } \theta = \{y \mapsto x', z \mapsto 0\} \rangle \]

\[ l_1 \rightarrow r_1 \in \{r8\} \land l_2 \rightarrow r_2 \in \{r4\} \iff \exists (\theta, p \mid \theta(l_1)p = \theta(l_2) = x' + 0) \]

\[ \iff \quad \langle \text{Definition of critical pair} 9 \rangle \quad \& \quad \theta = \{y \mapsto x', z \mapsto 0\} \]

\[ l_1 \rightarrow r_1 \in \{r8\} \land l_2 \rightarrow r_2 \in \{r3\} \iff l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ give raise of a critical pair } + ((x, x'), (x, 0)), (x, x') \]

\[ \iff \quad \langle + ((x, x'), (x, 0)) \rightarrow +(x, x'), 0 \rightarrow (x, x') \rangle \quad \rightarrow \quad \langle + (x, x'), 0 \rightarrow (x, x') \rangle \]

\[ l_1 \rightarrow r_1 \in \{r8\} \land l_2 \rightarrow r_2 \in \{r3\} \iff l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ give raise of a joinable critical pair in the rewriting system } R(E_f) \cup R(E_{spec}). \]

\[ \square \]

**Lemma 15.** The rewriting rule \(r8\) and \(r4\) only give raise of one joinable critical pair.

**Proof.** We give the proof by two cases.

1. Assume that \(r4\) is \(l_1 \rightarrow r_1\), and \(r8\) is \(l_2 \rightarrow r_2\). According to definitions of \(r8\) and \(r4\), we have:
\[ l_1 \rightarrow r_1 \in \{ r_4 \} \land l_2 \rightarrow r_2 \in \{ r_8 \} \implies \text{there is no non-variable strict subterm for } l_1 \]
\[ \iff \langle \text{Definition 9 of critical pair} \rangle \]
\[ l_1 \rightarrow r_1 \in \{ r_4 \} \land l_2 \rightarrow r_2 \in \{ r_8 \} \implies l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ cannot give raise of a critical pair.} \]

(2) Assume that \( r_8 \) is \( l_1 \rightarrow r_1 \), and \( r_4 \) is \( l_2 \rightarrow r_2 \). According to definitions of \( r_8 \) and \( r_4 \), we have:

\[ l_1 \rightarrow r_1 \in \{ r_8 \} \land l_2 \rightarrow r_2 \in \{ r_4 \} \implies \text{the non-variable strict subterm of } l_1 \text{ is } + (y, z) \land l_2 = + (x', x') \]
\[ \langle \text{Definition of mgu. In particular, } \theta = \{ y \mapsto \cdot x', z \mapsto \cdot x' \} \rangle \]
\[ l_1 \rightarrow r_1 \in \{ r_8 \} \land l_2 \rightarrow r_2 \in \{ r_4 \} \implies \exists (\theta, p : \theta(l_1)\vert_p = \theta(l_2) = x' + x') \]
\[ \iff \langle \text{Definition 9 of critical pair } \& \theta = \{ y \mapsto \cdot x', z \mapsto \cdot x' \} \rangle \]
\[ l_1 \rightarrow r_1 \in \{ r_8 \} \land l_2 \rightarrow r_2 \in \{ r_4 \} \implies l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ give raise of a critical pair} \langle + (\cdot (x, x'), \cdot (x, x')), \cdot (x, x')) \rangle \]
\[ \iff \langle + (\cdot (x, x'), \cdot (x, x')) \rightarrow r_4 \cdot (x, x') \rangle \]
\[ l_1 \rightarrow r_1 \in \{ r_8 \} \land l_2 \rightarrow r_2 \in \{ r_4 \} \implies l_1 \rightarrow r_1 \text{ and } l_2 \rightarrow r_2 \text{ give raise of a joinable critical pair in the rewriting system } R(E_f) \cup R(E_{spec}). \]

Lemma 16. Any rewriting rules in \( R(E_f) \cup R(Eq) \) cannot give raise of a critical pair with itself.

Proof. To show that any rewriting rules in \( R(E_f) \cup R(Eq) \) cannot give raise of a critical pair with itself, we divide the proof into three cases.

1. Consider rules in \( R(E_{spec}) \) and the rule \( r_4 \). According to their definitions, we have:

\[ \forall (l \mid l \rightarrow r \in R(E_{spec}) \cup \{ r_4 \} : \text{there is no proper subterm for } l \]
\[ \iff \langle \text{Definition of critical pair } \rangle \text{ It is safe to ignore } p = \epsilon \text{ when considering the overlap of one rule itself. } \]
\[ \forall (l \mid l \rightarrow r \in R(E_{spec}) \cup \{ r_4 \} : l \rightarrow r \text{ cannot give raise of a critical pair with itself } \]

2. Consider the rules \( r_3, r_7 \) and \( r_9 \). According to their definitions, we have:

\[ \forall (l \mid l \rightarrow r \in \{ r_3, r_7, r_9 \} : \text{the proper non-variable subterm of } l \]
\[ \text{is a constant } \]
\[ \iff \langle \text{Lemma 2 } \rangle \text{ In particular, there is no unifier for a constant and a non-variable and non-constant term } \]
∀(l | l → r ∈ \{r3, r7, r9\}) : ¬∃(θ, p | p ≠ ϵ : θ(l)|p = θ(l))

⇐⇒ ( Definition of critical pair 9. It is safe to ignore \(p = ϵ\) when considering the overlap of one rule itself. )

∀(l | l → r ∈ \{r3, r7, r9\}) : l → r cannot give raise of a critical pair with itself

3. Consider the rule r8. According to its definition, we have:

\[ l \rightarrow r \in \{r8\} \implies \text{the proper non-variable subterm of } l \text{ is in a form of } + (t_1, t_2) \]

⇐⇒ ( Lemma 2 In particular, there is no unification for two term \(+ (t_1, t_2)\) and \(+(t_1', t_2')\) )

\[ l \rightarrow r \in \{r8\} \implies ¬∃(θ, p | p ≠ ϵ : θ(l)|p = θ(l)) \]

⇐⇒ ( Definition of critical pair 9. It is safe to ignore \(p = ϵ\) when considering the overlap of one rule itself. )

\[ l \rightarrow r \in \{r8\} \implies l \rightarrow r \text{ cannot give raise of a critical pair with itself} \]

\[ \square \]

**Theorem 8.** For any syntactically correct specification \(S\) and aspect specification \(A\), the rewriting systems \(R(E_f) \cup R(E_{spec})\) is confluent.

**Proof.** Give the base specification \(S\) and aspect specification \(A\). According to the definitions of the rewriting system \(R(E_f) \cup R(E_{spec})\), we have:

The rewriting system \(R(E_f) \cup R(E_{spec})\) is confluent

⇐⇒ ( Theorem 7 & Theorem 5 )

All critical pairs of \(R(E_f) \cup R(E_{spec})\) are joinable

⇐⇒ ( Lemmas 7–16 )

true

\[ \square \]

In the Table 1, the four equations 1, 2, 5, and 6 axiomatize associativity and commutativity of the function symbols + and ·. The associativity and commutativity appearing in our equational theory, called AC-theory, are quite common properties for binary operations. We cannot introduce rewriting rules for the commutativity equation as the other equations in \(E_f\), since it will lead to a nonterminating rewriting system. Consequently, rewriting rules for AC-theory will be nonterminating. On the other hand, equation unification can be used for term rewriting, which considers semantic properties of function symbols. Therefore, instead of introducing rewriting rules for the AC-theory, we use AC-unification for the corresponding term rewriting with respect to the equations 1, 2, 5, and 6 in our equational theory. Algorithms of AC-Unification can be found in [2], and for many rewriting tools, we can use build-in AC-Unification modulo to implement the AC Unification directly.
Definition 10 \((\downarrow_{\text{AC}})\). Given a set of rewrite rules \(R\) on terms in \(T_f(V)\), we call a term an AC normal term iff the term and any equivalent term of it (up to the AC theory of \(+\) and \(\cdot\)) cannot be further rewritten by any rule in \(R\). For any term \(\beta\), we denote its AC normal form as \(\beta \downarrow_{\text{AC}}\).

Theorem 9. Let \(\text{Th}_{\text{pfa}}\) be the equational theory obtained according to the given base and aspect specifications. Then we have

\[
(\text{Th}_{\text{pfa}} \models (s = t)) \iff (R(E_f) \cup R(E_{\text{spec}}) \vdash (s \downarrow_{\text{AC}} = t \downarrow_{\text{AC}})),
\]

where \(\downarrow_{\text{AC}}\) denotes the equivalent of two terms up to the AC-theory of \(+\) and \(\cdot\).

Proof.

\[
(\text{Th}_{\text{pfa}} \models (s = t))
\]
\[
\iff \{\text{Definition of } \text{Th}_{\text{pfa}}\}
\]
\[
(\{e_3, e_4, e_7, e_8, e_9\} \cup E_{\text{spec}} \cup \{e_1, e_2, e_5, e_6\}) \models (s = t)
\]
\[
\iff \{\text{Lemma 1}\}
\]
\[
(\{e_3, e_4, e_7, e_8, e_9\} \cup E_{\text{spec}}) \models (\{e_1, e_2, e_5, e_6\} \implies (s = t))
\]
\[
\iff \{\text{Definition 6 & Definition 10}\}
\]
\[
(R(E_f) \cup R(E_{\text{spec}})) \vdash (s \downarrow_{\text{AC}} = t \downarrow_{\text{AC}})
\]

\(\blacksquare\)

The corollary below ensure that the Concern 1 is satisfied.

Corollary 1. Given a syntactically correct PFA base specification and an aspect specification, we have a decidable procedure for the weaving process.

Proof. Based on Propositions 7, 8, we show that the term rewriting system \(R(E_f) \cup R(E_{\text{spec}})\) for a syntactically correct PFA base specification and an aspect specification is convergent (i.e., terminating and confluent). According to Theorem 9, it is obvious that we have a decidable procedure for the word problem with regard to \(\text{Th}_{\text{pfa}}\), where \(\text{Th}_{\text{pfa}}\) is decided by the given PFA and aspect specifications. Therefore, a decidable procedure for solving the word problem w.r.t. \(\text{Th}_{\text{pfa}}\) indicates a decidable procedure for the weaving process. \(\blacksquare\)

3.3 Unambiguousness of the Weaving Results

As mentioned earlier, the weaving of an aspect in AO-PFA is to substitute a particular subterm in product family terms with another term. Concern 2 indicates that the form of the selected joint points, which corresponds to the form of kind pointcut in AO-PFA, should be restricted to ensure the uniqueness of the substituted terms. Since the equivalence of two terms is decided according to a rewriting system as discussed in the previous subsection, the weaving of aspects should be restricted by the introduced rewriting system.
3.3.1 The restriction on the form of the kind pointcut

**Lemma 17.** Given a PFA specification $S$, let $\beta$ be a non-ground term in $T_f(\Sigma(S))$. We have:

$$\left( R(E_f) \cup R(E_{\text{spec}}) \right) \vdash (\beta \downarrow_{AC} = \sum_{i=1}^{m} t_i),$$

where all $t_i$ are not equivalent (up to AC theory of $\cdot$), and each $t_i (1 \leq i \leq m)$ is either 1 or $\prod x_{ij}$ that $x_{ij} \in \Sigma(S)$.

**Proof.** Let $\beta \xrightarrow{R(E_{\text{spec}})} \beta'$, where $\beta'$ cannot be further rewritten by any rule from $R(E_{\text{spec}})$. In other words, no label from $\Sigma(S)$ in $\beta'$ matches the left-hand side of any rule in $R(E_{\text{spec}})$. Consequently, any term derived from $\beta'$ by applying rules in $R(E_f)$ cannot be further rewritten by any rule from $R(E_{\text{spec}})$, since applying any rule in $R(E_f)$ will not create new labels. Hence, we can obtain the AC normal form of $\beta$ by only applying the AC theory and rules of $R(E_f)$ on $\beta'$. In particular, the given the proof based on the inductive definition of $\beta'$, which is a term in the set $T_f(\Sigma(S)) - T_f(\emptyset)$.

The strict subterm relationship $\prec$ give a well-founded order over the set of terms $T_f(\Sigma(S)) - T_f(\emptyset)$. Let us define the structural inductive property formally as follows:

$$P(\beta') \iff \left( R(E_f) \right) \vdash (\beta' \downarrow_{AC} = \sum_{i=1}^{m} t_i),$$

where all $t_i$ are not equivalent (up to AC theory of $\cdot$), and each $t_i$ is either 1 or $\prod x_{ij}$ that $x_{ij} \in \Sigma(S)$.

1. **Base cases:** The minimal elements in $T_f(\Sigma(S)) - T_f(\emptyset)$ are all labels in $\Sigma(S)$. We need to prove $\forall(x \mid x \in \Sigma(S)) : P(x) \iff \left( R(E_f) \right) \vdash (x \downarrow_{AC} = \sum_{i=1}^{m} t_i)$.

   The proof is trivial. Any label $x$ from $\Sigma(S)$ can be considered as a case of $\sum_{i=1}^{m} t_i$, where $m = 1$ and $t_1 = x$. Furthermore, no axioms of AC theory and rules of $R(E_f)$ can be applied to $x$, since a label from $\Sigma(S)$ cannot be an instance of any term at the left-hand sides of the rewriting rules of $R(E_f)$.

2. **Inductive cases:** All constructs over terms are $+$ and $\cdot$. We have $t_1, t_2 \prec t_1 + t_2$, and $t_1, t_2 \prec t_1 \cdot t_2$. Let assume that:

   - $P(t_1) \iff \left( R(E_f) \right) \vdash t_1 \downarrow_{AC} = \sum_{i=1}^{m_1} p_i$, where all $p_i$ are not equivalent (up to AC theory of $\cdot$), and each $p_i (1 \leq i \leq m_1)$ is either 1 or $\prod x_{ij}$ that $x_{ij} \in \Sigma(S)$.
   - $P(t_2) \iff \left( R(E_f) \right) \vdash t_2 \downarrow_{AC} = \sum_{i=1}^{m_2} q_i$, where all $q_i$ are not equivalent (up to AC theory of $\cdot$), and each $q_i (1 \leq i \leq m_2)$ is either 1 or $\prod x_{ij}$ that $x_{ij} \in \Sigma(S)$.

(a) We need to prove $P(t_1) \land P(t_2) \implies P(t_1 + t_2)$. The idea is to obtain the AC normal form of $t_1 + t_2$ according to the AC normal form of $t_1$ and $t_2$. 
We also prove
\[ P = m \cdot \text{equivalent up to the AC theory of } \sum \text{ and } R \]
\[ \text{AC} \implies \langle \text{The commutativity of + implies that the sum of a sequence of terms is equivalent (up to the AC theory of +) with the sum of a permutation of those terms. Therefore, let } \{p'_1, \ldots, p'_m\} \text{ be a permutation of } \{p_1, \ldots, p_m\}, \text{ and } \{q'_1, \ldots, q'_m\} \text{ be a permutation of } \{q_1, \ldots, q_m\} \text{ such that } \forall(i \mid 1 \leq i \leq k) : p'_{m_1-k+i} = q'_{i}. \text{ Assume that } k \text{ is the number of equivalent (up to the AC theory of } \cdot \text{) elements of } \{p_1, \ldots, p_m\} \text{ and } \{q_1, \ldots, q_m\}. \rangle \]
\[ \sum_{i=1}^{m_1-k} p'_i + (\sum_{i=m_1-k+1}^{m_1} p'_i + \sum_{i=1}^{k} q'_i) + \sum_{i=k+1}^{m_2} q'_i \]
\[ \text{AC} \implies \langle \text{(Apply } k \text{ times the rewrite rule } r4 \text{ for } p'_{m_1-k+i} = q'_{i}. \rangle \]
\[ \sum_{i=1}^{m_1-k} p'_i + (\sum_{i=m_1-k+1}^{m_1} p'_i + \sum_{i=1}^{k} q'_i) + \sum_{i=k+1}^{m_2} q'_i \]
\[ \sum_{i=1}^{m_1} p'_i + \sum_{i=k+1}^{m_2} q''_i \]
\[ \langle \text{No rules of } R(E_f) \text{ can be further applied to } \sum_{i=1}^{m_1} p'_i + \sum_{i=k+1}^{m_2} q''_i \text{ or to its equivalent terms (up to the AC theory of + an } \cdot \text{).} \rangle \]

According to the above steps, we obtain the AC normal form of \( t_1 + t_2 \) as \( \sum_{i=1}^{m_1} p'_i + \sum_{i=k+1}^{m_2} q''_i \). Moreover, the assumption of \( P(t_1) \) and \( P(t_2) \) indicates that each \( p'_i \) and \( q''_i \) is either 1 or \( \prod x_{ij} \) such that \( x_{ij} \in \mathcal{L}(S) \), and all \( p'_i \) and \( q''_i \) are not equivalent up to the AC theory of \( \cdot \). In other words, we prove \( P(t_1 + t_2) \), and particularly, \( m = m_1 + m_2 - k \), and \( t_i = p'_i \) for \( 1 \leq i \leq m_1 \) and \( t_i = q''_i \) for \( m_1 + 1 \leq i \leq m_1 + m_2 - k \).

(b) We also prove \( P(t_1) \land P(t_2) \implies P(t_1 \cdot t_2) \) in the same way as the case (a).
\[ \xrightarrow{\ast \text{AC}}_{R(E_f)} \langle \text{Same as the proof in case (a), we consider a permutation of } s_1, \ldots, s_{m_1m_2} \text{ in order to apply the rewrite rules } r4. \text{ In particular, } \text{let } s'_1, \ldots, s'_k \text{ be all non-equivalent (up to the AC theory of } \cdot \text{) terms from } s_1, \ldots, s_{m_1m_2}. \rangle \]

\[ \sum_i s'_i \]

According to the above steps, we obtain the AC normal form of \( t_1 + t_2 \) as \( \sum_{i=1}^{k} s'_i \), where all \( s'_i \) are not equivalent up to the AC theory of + and \( \cdot \), and each \( s'_i \) is either 1 or \( \prod x_{ij} \) such that \( x_{ij} \in \mathcal{E}(S) \). In other word, we prove \( P(t_1 \cdot t_2) \), and particularly, \( t_i = s'_i \) and \( m = k \) such that \( 1 \leq k \leq m_{1m_2} \).

In summary, we prove the \( P(\beta') \) for all \( \beta' \in T_f(\mathcal{E}(S)) - T_f(\emptyset) \) based on the above proofs of base cases and inductive cases for \( \beta' \).

\[ \square \]

**Lemma 18.** Given a product family term \( s \), let \( (R(E_f) \cup R(E_{spec})) \vdash (s \upharpoonright_{AC} \sum_{i=1}^{m} s_i) \). A term \( e \) of a product of labels cannot be extracted from \( s \) iff \( \forall(i \mid 1 \leq i \leq m : e \upharpoonright s_i) \), where \( x \upharpoonright y \iff -\forall(x \mid y) \leftrightarrow \exists(z \mid y = x \cdot z) \).

**Proof.** We prove by contradiction.

\[ \neg \forall(i \mid 1 \leq i \leq m : e \upharpoonright s_i) \]

\[ \iff \langle \text{Generalized De Morgan} \rangle \]

\[ \exists(i \mid 1 \leq i \leq m : e|s_i) \]

\[ \iff \langle \text{Lemma 17} \rangle \& \ (e \text{ is a product of labels} \& \text{ Instance of } \exists. \text{ In particular, let } s_k = e \cdot t \text{ where } 1 \leq k \leq m \rangle \]

\[ R(E_f) \cup R(E_{spec}) \vdash s \upharpoonright_{AC} \sum_{i=1}^{k-1} s_i + e \cdot t + \sum_{i=k+1}^{m} s_i \]

\[ \iff \langle \text{Associativity and Community of } + \rangle \]

\[ R(E_f) \cup R(E_{spec}) \vdash s \upharpoonright_{AC} e \cdot t + \sum_{i=1}^{k-1} s_i + \sum_{i=k+1}^{m} s_i \]

\[ \iff \langle \text{Corollary 9} \rangle \]

\[ Th_{pfa} = (s = e \cdot t + (\sum_{i=1}^{k-1} s_i + \sum_{i=k+1}^{m} s_i)) \]

\[ \implies \langle \text{Assumption: } e \text{ cannot be extracted from } s \rangle \]

false

\[ \square \]

**Lemma 19.** Give two product family terms \( \beta \) and \( \gamma \). Let \( e \) be another term in the form of a product of labels that cannot be extracted from either \( \beta \) or \( \gamma \). Assume \( (R(E_f) \cup R(E_{spec})) \vdash (\beta \upharpoonright_{AC} \sum_{i=1}^{m} p_i) \), \( (\gamma \upharpoonright_{AC} \sum_{i=1}^{l} s_i) \), then we have

\[ (R(E_f) \cup R(E_{spec})) \vdash ((\beta \cdot e + \gamma) \upharpoonright_{AC} \sum_{i=1}^{m} (p_i \cdot e) + \sum_{i=1}^{l} s_i) \].
Proof. $\beta \cdot e + \gamma$

$$\xrightarrow{\ast R_{AC}} \frac{\beta}{\gamma} \quad \langle \text{Assumption of } \beta \mid_{AC} \text{ and } \gamma \mid_{AC} \rangle$$

$$\sum_{i=1}^{m} p_i \cdot e + \sum_{i=1}^{l} s_i \quad \xrightarrow{\ast R_{(E_f)}} \quad \langle \text{Apply the rewrite rule } r8 \text{ in } E_f \text{ } m \text{ times} \rangle$$

$$\sum_{i=1}^{m} (p_i \cdot e) + \sum_{i=1}^{l} s_i$$

Since $e$ and all $p_i$ are in the form of products of labels, and all $p_i$ are distinct, we have all $p_i \cdot e$ are distinct. Moreover, Lemma 18 implies any $p_i \cdot e$ and any $q_i$ cannot be equivalent. Therefore, it is obvious that no further rules can be applied to any equivalent terms of $\sum_{i=1}^{m} (p_i \cdot e) + \sum_{i=1}^{l} s_i$. \hfill \square

Lemma 20. Give two product family terms $s$ and $t$. Let $(R(E_f) \cup R(E_{spec})) \vdash (s \mid_{AC} = \sum_{i=1}^{m} s_i)$, and $(R(E_f) \cup R(E_{spec})) \vdash (t \mid_{AC} = \sum_{i=1}^{n} t_i)$. We have $Th_{pfa} \models s = t \iff \exists (f \mid f \in \mathcal{P}\{1, \ldots, m\} \times \mathcal{P}\{1, \ldots, n\} \wedge f \text{ is a bijection : } f(i) = j \iff s_i \doteq t_j)$, where $\doteq$ means two terms are equivalent up to the community and associativity of $\cdot$.

Proof.

$$Th_{pfa} \models (s = t)$$

$$\iff \langle \text{Corollary } [9] \rangle$$

$$(R(E_f) \cup R(E_{spec})) \vdash (s \mid_{AC} \overset{AC}{=} t \mid_{AC})$$

$$\iff \langle \text{The AC normal form of } s \text{ and } t. \rangle$$

$$(R(E_f) \cup R(E_{spec})) \vdash \sum_{i=1}^{m} s_i \overset{AC}{=} \sum_{i=1}^{n} t_i$$

$$\iff \langle \text{Regarding to associativity and community of } + \rangle$$

$$\exists (f \mid f \in \mathcal{P}\{1, \ldots, m\} \times \mathcal{P}\{1, \ldots, n\} \wedge f \text{ is a bijection : } f(i) = j \iff s_i \overset{AC}{=} t_j)$$

$$\iff \langle \text{Lemma } [17] \text{. In particular, all } s_i \text{ and all } t_i \text{ are products of labels \& The AC theory of + cannot apply to a term that is a product of labels} \rangle$$

$$\exists (f \mid f \in \mathcal{P}\{1, \ldots, m\} \times \mathcal{P}\{1, \ldots, n\} \wedge f \text{ is a bijection : } f(i) = j \iff s_i \doteq t_j)$$

\hfill \square

Theorem 10. Give two product family terms $e$ and $\alpha$. Assume that w.r.t. the set of equations $Th_{pfa}$, $\alpha = \beta_1 \cdot e + \gamma_1 = \beta_2 \cdot e + \gamma_2$, where $e$ cannot be further extracted from $\beta_1$, $\beta_2$, $\gamma_1$ and $\gamma_2$. Then if $e$ is a product of labels, we have $\beta_1 = \beta_2 \wedge \gamma_1 = \gamma_2$ w.r.t. to the set of equations $Th_{pfa}$.

Proof. According to assumption in the proposition, we have:

$$Th_{pfa} \models \beta_1 \cdot e + \gamma_1 = \beta_2 \cdot e + \gamma_2$$

$$\iff \langle \text{Theorem } [9] \rangle$$
\((R(E_f) \cup R(E_{eq})) \vdash (\beta_1 \cdot e + \gamma_1) \downarrow_{AC} \equiv (\beta_2 \cdot e + \gamma_2) \downarrow_{AC}\)

\[\iff \langle \text{Lemma 19} \rangle \text{ In particular, let } \beta_1 \downarrow_{AC} = \sum_{i=1}^{m} p_i, \beta_2 \downarrow_{AC} = \sum_{i=1}^{n} q_i, \gamma_1 \downarrow_{AC} = \sum_{i=1}^{m} s_i, \text{ and } \gamma_2 \downarrow_{AC} = \sum_{i=1}^{r} t_i. \rangle \]

\((R(E_f) \cup R(E_{eq})) \vdash \sum_{i=1}^{m} (p_i \cdot e) + \sum_{i=1}^{l} s_i \downarrow_{AC} = \sum_{i=1}^{n} (q_i \cdot e) + \sum_{i=1}^{r} t_i \)

\[\iff \langle \text{Lemma 18} \ & \text{ Lemma 20} \rangle \]

\[\exists (f \mid f \in \mathcal{P}\{1, \ldots, m\} \times \mathcal{P}\{1, \ldots, n\}) \wedge f \text{ is a bijection : } f(i) = j \]

\[\iff p_i \cdot e = q_j \cdot e \]

\[\wedge \exists (f' \mid f' \in \mathcal{P}\{1, \ldots, l\} \times \mathcal{P}\{1, \ldots, r\}) \wedge f \text{ is a bijection : } f(i) = j \]

\[\iff s_i \cdot t_j \]

\[\iff \langle \text{Axiom of } \sum \ & \text{ Definition of } \downarrow_{AC} \rangle \]

\[\iff (\sum_{i=1}^{m} p_i \downarrow_{AC} = \sum_{i=1}^{n} q_i) \wedge (\sum_{i=1}^{l} s_i \downarrow_{AC} = \sum_{i=1}^{r} t_i) \]

\[\iff \langle \text{Definitions of } \beta_1 \downarrow_{AC}, \beta_2 \downarrow_{AC}, \gamma_1 \downarrow_{AC}, \text{ and } \gamma_2 \downarrow_{AC} \rangle \]

\[\iff \langle \text{Theorem 9} \rangle \]

\[Th_{pfa} \models \beta_1 = \beta_2 \land \gamma_1 = \gamma_2 \]

Corollary 2. Give a syntax correct base specification \(S\), and an aspect \(A\). Using the decidable procedure given in Corollary 7 for the weaving process, the weaving result is unambiguous if the kind of pointcut in \(A\) is a product of labels from \(\Sigma(S)\).

Proof. We use the decidable procedure as given in Corollary 1 for the weaving process. According to Theorem 10, it is straightforward to see that if the kind of join-point \(e\) is in the form of a product of labels, we can have unique \(\beta\) and \(\alpha\) for an arbitrary term \(\alpha\).

The above corollary addresses the Concern 2 of the weaving process in AO-PFA.

4 Conclusion and Future Work

In this technical report, we give the detail of proofs related to the weaving process in AO-PFA. We briefly discuss the problems introduced to the weaving process in the context of AO-PFA, and show how those problems are tackled according to formal proofs. In particular, the rewriting system for the aspect-oriented specification language AO-PFA is proved to be convergent, and the proposed restriction on the form of kind pointcut is proved to avoid ambiguous weaving results.

The short term future work is to implement the proposed weaving process upon the existing tool Jory. With the formalism of the weaving process, an extensional module to Jory for AO-PFA can be constructed easily. On the other hand, the proofs presented in this report have shown that such implementation of the weaving process is reasonable (i.e., convergence and unambiguity). The long term future work is to get closer to the
automatic code generation of product families from the base specification, the aspect specification, and the specifications for each basic feature. With the introduction of aspects at the feature modeling level, the specification of a basic feature should be modified to deal with aspects from a finer granularity. The current specification of a basic feature in [9] is considered as the basis for the outgoing work.

References


