Journal of Logical and Algebraic Methods in Programming 85 (2016) 200–226

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

# Product line process theory

Fatemeh Ghassemi [a,*], Mohammad Reza Mousavi [b,1]

[a] School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
[b] Centre for Research on Embedded Systems (CERES), School of IT, Halmstad University, Sweden

ARTICLE INFO

*Article history:*
Received 4 August 2014
Received in revised form 3 August 2015
Accepted 17 September 2015
Available online 4 November 2015

*Keywords:*
Software product line
Process theory
Product line bisimulation
Strict strong bisimulation
$\mu$-Calculus
Axiomatization

ABSTRACT

Software product lines (SPLs) facilitate reuse and customization in software development by genuinely addressing the concept of variability. Product Line Calculus of Communicating Systems (PL-CCS) is a process calculus for behavioral modeling of SPLs, in which variability can be explicitly modeled by a binary variant operator. In this paper, we study different notions of behavioral equivalence for PL-CCS, based on Park and Milner's strong bisimilarity. These notions enable reasoning about the behavior of SPLs at different levels of abstraction. We study the compositionality property of these notions and the mutual relationship among them. We further show how the strengths of these notions can be consolidated in an equational reasoning method. Finally, we designate the notions of behavioral equivalence that are characterized by the property specification language for PL-CCS, called multi-valued modal $\mu$-calculus.


## 1. Introduction

*Software product line (SPL)* engineering has become an established trend in software development, where a family of similar software products with minor differences are developed in tandem, instead of developing each specific software product separately [1]. SPL engineering benefits from systematic reuse throughout the system life cycle and enables mass development and customization of numerous products. Hence, the development cost and the time to market for an SPL is substantially decreased, compared to the cumulative development cost and time of the isolated products [2]. To this aim, various software engineering activities have to be adapted to cope with the differences among the artifacts for different products, called *variability*. Variability introduces a new complexity dimension and hence, this calls for a genuine treatment of variability in different artifacts (such as requirement specification, architectural design, detailed design, and implementation artifacts). Such a treatment should also allow for a collective analysis of product line behavior (e.g., in testing and verification [3,4]) to deal with the inherent complexity of SPLs.

At the highest level of abstraction, an SPL can be specified by a set of features that satisfy the specific needs of a particular market segment or mission [5]. A feature identifies "a prominent or distinctive unit of requirement which can be either a user-visible behavior, aspect, quality, or characteristic of a software system" [6]. Hence, a product can be specified by a subset of features. To specify an SPL, the features are organized in a hierarchical model, called a *feature model*.

* Corresponding author.
*E-mail address:* fghassemi@ut.ac.ir (F. Ghassemi).
[1] The work of M.R. Mousavi has been partially supported by the Swedish Research Council (Vetenskapsrådet) award number: 621-2014-5057 (Effective Model-Based Testing of Concurrent Systems) and the Swedish Knowledge Foundation (Stiftelsen för Kunskaps- och Kompetensutveckling) in the context of the AUTO-CAAS HöG project (number: 20140312).

http://dx.doi.org/10.1016/j.jlamp.2015.09.008
2352-2208/

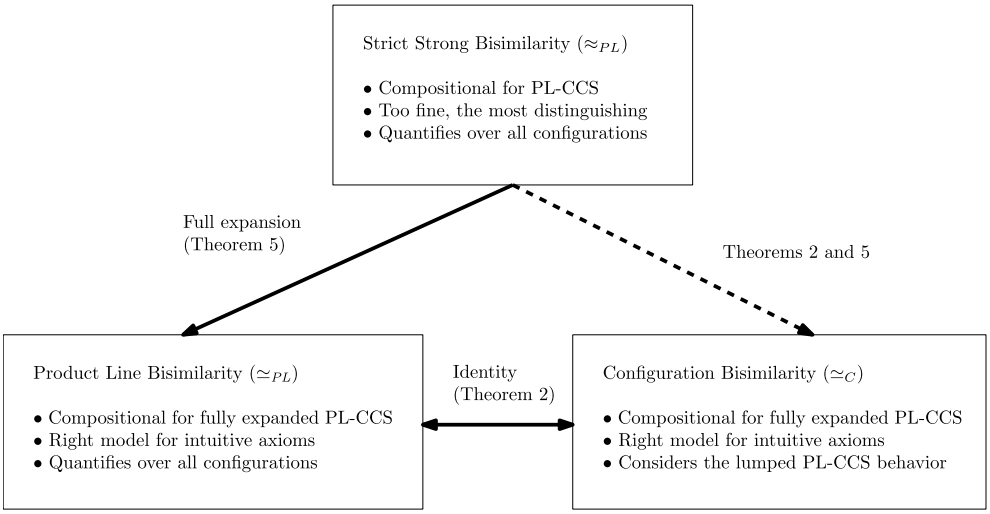


**Fig. 1.** Our notions of behavioral equivalence for PL-CCS specifications.

It identifies commonalities and differences among the products of the SPL in terms of their features and it identifies suitable relations among features, such as optional, mandatory, or mutually exclusive. More concrete specifications capture structural and behavioral aspects of an SPL. For instance, the architecture description languages Koala [7] and xADL [8] concentrate on structural modeling of SPLs. Modal transition systems (MTSs) [9] and Featured transition systems (FTSs) [10], however, concentrate on behavioral modeling (we refer to [11] for an overview of such behavioral models). MTSs capture the behavior of SPLs by defining state transitions as optional or mandatory, while FTSs annotate transitions with a set of features. Behavioral models typically come equipped with a product derivation method; e.g., a product, derived from a feature model, can project an FTS into a labeled transition system (LTS).

Formal verification techniques provide strong tools to analyze complex systems to guarantee their correctness. Process algebra is a formal approach to describe the behavior of communicating concurrent systems in a compositional manner. Product Line Calculus of Communicating Systems (PL-CCS) [12,13] is an extension of Milner's Calculus of Communicating Systems (CCS) [14]. PL-CCS extends CCS by adding the binary variant operator $\oplus_i$ to model behavioral variability in SPLs. More specifically, process term $p_1 \oplus_1 p_2$, where $p_1$ and $p_2$ are CCS process terms, specifies a family of two alternative products, namely $p_1$ or $p_2$ (the index $i$ in $\oplus_i$ is used to designate repeated choices that have to be made in the same way; when no repetition of indices is present, the indices can be safely ignored). The semantics of a PL-CCS specification is given in terms of three different models: the *flat semantics*, the *unfolded semantics*, and the *configured-transition semantics* [12]. A PL-CCS specification can be turned into a product, specified by a CCS term, by resolving the variability points, i.e., the variant operators, by deciding on whether their right or left process is chosen. The flat semantics of a PL-CCS term is given in terms of the semantics of all derivable products, denoted by CCS terms. A product family LTS (PF-LTS) is an extension of an LTS, where labels and states are paired with configuration vectors that maintain the configuration of variants. PF-LTSs provide the unfolded semantics of PL-CCS terms and are derived through a set of structural rules in a systematic way. The structural rules given in [12] work on a restricted set of PL-CCS terms in order to be compositional. The configured-transition semantics is defined over the unfolded semantics by merging all states that only differ in their configuration parts. This provides the most succinct model of PL-CCS terms. Hence, in the developments to come, we mostly focus on the configured-transition semantics of PL-CCS. In particular, we provide a set of structural rules that derive a configured-transition semantics for PL-CCS terms directly.

Equational reasoning is the cornerstone of the algebraic approach to process theory. To furnish PL-CCS with a proper equational theory, we study a number of notions of behavioral equivalence, based on strong bisimilarity [15]. A summary of these notions, their properties and the results establishing their relationship is depicted in Fig. 1. We start with a set of axioms that we expect to be sound for a model of PL-CCS and define the notion of strict strong bisimilarity, which is a natural extension of strong bisimilarity in the SPL setting. Namely, strict strong bisimilarity requires bisimilar product lines to behave bisimilarly for all common configurations. This turns out to be a fully compositional notion for PL-CCS, but too strong of a notion for some of our intuitive axioms. For example, strict strong bisimilarity rejects $p \oplus_i q = q \oplus_i p$, which is an intuitive axiom. Subsequently, we introduce a strictly coarser notion, called product line bisimilarity, which does satisfy the axioms we defined for PL-CCS. However, this notion is shown to satisfy a weaker compositionality property. Namely, it is compositional for a subset of PL-CCS terms, called fully expanded terms. To remedy the latter issue, we show that all PL-CCS term can be rewritten into this subset using a sound transformation, thanks to the strong compositional notion of strict strong bisimilarity. Since strict strong bisimilarity implies product line bisimilarity this transformation is also sound for the latter notion and hence, resolves its compositionality issue. Finally, we introduce configuration bisimilarity, which is an alternative yet equivalent notion for product line bisimilarity. The main motivation for introducing configuration bisimilarity