Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 700 – 704

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

Using Failures and Category Theory to Verify Process Communications between Design and Implementation of Concurrent Systems

Ming Zhu[a], Peter Grogono[a], Olga Ormandjieva[a], Heng Kuang[b]

[a]*Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada*
[b]*IBM Canada, Markham, Canada*

Abstract

The process-oriented design and implementation of concurrent systems have important advantages. However, it is challenging to verify the consistency of process communications between the design and the implementation. To deal with such a challenge, we construct a formal framework for designing, implementing and verifying the consistency of process communications. In this framework, we use Failures in Communicating Sequential Processes (CSP), Erasmus and Category Theory as the foundation. The framework is illustrated by using a running example.

Keywords: concurrent system; verification; category theory; Failures; CSP; process-oriented programming

1. Introduction

Process-oriented approach is a necessary concept for designing and implementing concurrent systems[1]. However, design and implementation are usually at different levels of abstraction in software development process. It is challenging to incorporate knowledge and experience to control the consistency between those phases[2]. To deal with this, verification plays a critical role in checking the consistency between design and implementation of a concurrent system[3]. Research[4,5] used category theory, dataflow and traces of processes to explore approaches that may address that challenge. As a continuation, this paper uses failures to verify the consistency of process communications between design and implementation of concurrent systems.

The rest of this paper is organized as follows: Section 2 provides background knowledge and related work on the Communicating Sequential Processes (CSP), the process-oriented programming language Erasmus, and category

* Olga Ormandjieva. Tel.: +1-514-848-2424 (ext.7810) ; fax: +1-514-848-2830.
E-mail address: ormandj@cse.concordia.ca

doi:10.1016/j.procs.2016.04.155

theory. In Section 3, the categorical framework is introduced for formally designing, implementing, and verifying concurrent systems. In section 4, each step in that framework is applied to a running example. Section 5 concludes our paper and suggests directions for future research.

2. Background and Related Work

In this section, the background and related work on our research are introduced.

2.1. Communicating Sequential Processes

Process algebra has been developed to model concurrent systems by describing algebras of communicating processes. CSP is a process algebra that formally models concurrent systems by events[6,7]. It has been widely used to specify, design and implement concurrent systems. In CSP, a process is defined as (*alphabet*, *failures*, *divergences*)[6,7]. If a process is assumed not to become chaos, (*alphabet*, *failures*) is enough to describe safety and liveness of the process[1]. Several operators are defined to describe the relationships between processes. Given two processes P and Q, CSP can calculate sequence $P\,;Q$, deterministic choice $P \,\Box\, Q$, non-deterministic choice $P \sqcap Q$, parallel execution $P \parallel Q$, and iteration, using the recursion operator $\mu P : A \cdot F(P)$.

Traces, failures and divergences with operations on processes are used to analyze the liveness and correctness of processes[1]. Besides, traces are used to verify process communications between design and implementation[5].

2.2. Erasmus

Process-oriented programming is predicted to be the next programming paradigm[1,8]. The foundation of process-oriented programming is process algebra[9]. Erasmus is a process-oriented programming language based on CSP but with some differences[8]. An Erasmus program consists of *cells*, *processes*, *ports*, *protocols* and *channels* . A cell, containing a collection of one or more processes or cells, provides the structuring mechanism for an Erasmus program. A process is a self-contained entity which performs computations, and communicates with other processes through its ports. A port, which is of a type of protocol, serves as an interface of a process for sending and receiving messages. A protocol specifies the type and the orderings of messages that can be sent and received by the ports of the type of this protocol. A channel, which is of a type of protocol, links two ports and so enables two processes to communicate.

Some research is proposed to study communications in Erasmus, which includes constructing a fair protocol that allows arbitrary, nondeterministic communication between processes [10], and building a static analyzer to detect communication errors between processes[11].

2.3. Category Theory

Due to its abstractness and generality, category theory has led to its use as a conceptual framework in many areas of computer science[12] and software engineering[13]. It is suggested that category theory can be helpful towards discovering and verifying connections in different areas, while preserving structures in those areas[14]. In software engineering, category theory is proposed as an approach to formalizing refinement from design to implementation[15]. Specifically, for modeling concurrency, category theory is used to model, analyze, and compare *Transition Systems*, *Trace Languages*, *Event Structures*, *Petri nets*, and other classical models of concurrency[16]. However, to the best of our knowledge, there is no such kind of categorical framework for verifying the consistency between process-oriented design and implementation. The aim of this paper is to work on the categorical framework based on research[4,5]. In this paper, we use the constructs *Category* and *Functor* from category theory for the verification.

3. The Categorical Framework

Based on research[4,5], the categorical framework for verification consists of the following steps (See Fig. 1).

(1). Designing: design concurrent systems in CSP, and analyze failures of processes together with communications,

(2). Implementing: implement concurrent systems in Erasmus with the design refinement,