ProCSuS: A Meta System for Concurrent Process Calculi based on SOS *

Atsushi Togashi, Sen Yoshida, Shigetomo Kimura, Norio Shiratori

Research Institute of Electrical Communication, Tohoku University
2-1-1, Katahira, Aoba-ku, Sendai, 980, Japan
E-mail: {togashi,yoshida,kimura,norio}@shiratori.riec.tohoku.ac.jp

1 Introduction

The studies of process calculi started from the latter half of 1970's to deal with formally the behavior of concurrent processes, or multi agents. In the literature, a variety of process calculi have been proposed. The typical systems are CSP by Hoare[8], CCS by Milner[11], τ-calculus by Milner, Parrow, Walker [10], γ calculus by Boudol[4], CHOCS by Thomsen[13] and ACP by Bergstra, Klop[2]. In Feb. 1990, ISO adopted LOTOS[5] as the international standard for OSI specification description language. Algebraic formalization techniques are utilized as the descriptive languages for communicating processes and concurrent programs. They are also applied to the verification problem, by virtue of the mathematical formality.

In general, implementation of such calculi may take much efforts, and this has disturbed developing new process calculi. So, it is promising to propose a meta system for process calculi, a general designing environment of process calculi.

The advantages of the meta system can be stated as follows:

- It works as a general interpreter of existing process calculi.
- It works as a support system in developing a new calculus.
- It gives a formal method to deal with various calculi uniformly.

In this paper, we propose a Language of Concurrent process Calculi, LCC, as a underlying meta language to describe process calculi uniformly based on structural operational semantics (SOS) by Plotkin [12]. We also describe the outline of a meta system for concurrent process calculi ProCSuS, a Process Calculus Support System consisting of the LCC interpreter and several attractive tools. Finally we conclude with a discussion about on going works.

2 Many Sorted SOS Format

In this section, we propose a many sorted SOS format, a general framework for process calculi, which is a proper extension of TSS by Groote, Vaandrager [7]

* A part of this study is supported by Grants from the Asahi Glass Foundation and Science Grant from Min. of Education
GSOS by Bloom, Istrail, Meyer[3] based on the paradigm of SOS by Plotkin [12]. We begin with the definition of a language in which processes, actions, and transitions are represented as terms of specific sorts.

Let \( \langle S, \Sigma \rangle \) be a signature. \( S \) is a (finite) set of sorts and \( \Sigma = \{ \Sigma_w, s \} \) is a family of sets \( \Sigma_w, s \) of symbols, where \( w \in S^*, s \in S \). An element of \( \sigma \in \Sigma_w, s \) is called an operator symbol of sort \( s \) with rank \( w \), denoted by \( \sigma : w \rightarrow s \). It is assumed that we have a family of denumerable sets \( X = \{ X_s \}_s \in S \) such that \( X_s \cap X_{s'} = \emptyset \) if \( s \neq s' \). An element \( x \) in \( X_s \) is called a variable of sort \( s \). A transition language is specified as a tuple \( \langle S, \Sigma, X, \rightarrow \rangle \), where \( \langle S, \Sigma \rangle \) is a signature, \( X \) is a family of variable sets, and \( \rightarrow \) is the special symbol used for transition rules. Let \( \langle S, \Sigma, X, \rightarrow \rangle \) be a transition language. A term of sort \( s \in S \) is defined inductively in the usual way. Let \( T(\Sigma, X)_s \) denote the set of all terms of sort \( s \in S \) and set \( T(\Sigma, X) = \{ T(\Sigma, X)_s \}_s \in S \). \( T(\Sigma) \) denotes the set of all ground (closed) terms of sort \( s \) and set \( T(\Sigma) = \{ T(\Sigma)_s \}_s \in S \). A substitution is a mapping \( \theta : X \rightarrow T(\Sigma, X) \) such that \( \theta(x) \in T(\Sigma, X)_s \) for all \( x \in X_s \) (\( s \in S \)). A substitution can be easily extended from variables to terms.

A transition assertion is an expression of the form \( \xi \overset{\alpha}{\rightarrow} \eta \), where \( \xi \) and \( \eta \) are terms of the same sort. A transition rule is an expression of the form \( \varphi/\varphi_1 \cdots \varphi_n \), where \( \varphi_1, \ldots, \varphi_n \) and \( \varphi \) are transition assertions. \( \varphi_1, \ldots, \varphi_n \) are called the assumptions and \( \varphi \) is called the conclusion of the rule. An abstract transition system is a pair \( \langle L, \Gamma \rangle \), where \( L = \langle S, \Sigma, X, \rightarrow \rangle \) is a transition language and \( \Gamma \) is a set of transition rules. In the usual manner, we can define the notion of provability of an transition assertion \( \varphi \) from a set \( \Gamma \) of transition rules. Let \( \Gamma \vdash \varphi \) denote that a transition assertion \( \varphi \) is provable from a set of transition rules \( \Gamma \).

In the literature, labeled transition systems have been used to give operational semantics for concurrent systems based on the idea of SOS. Its definition and its related notations are given formally as follows:

A labeled transition system (LTS) is a triple \( \langle S, A, \rightarrow \rangle \), where \( S \) is a set of states and \( \rightarrow \) is a transition relation defined as \( \rightarrow \subseteq S \times A \times S \). The transition relation defines the dynamical change of states as actions may be performed. For \( (s, a, s') \in \rightarrow \), we normally write \( s \xrightarrow{a} s' \). Thus, the transition relation can be written as \( \rightarrow = \{ a \mid a \in A \} \). \( s \xrightarrow{a} s' \) may be interpreted as "in the state \( s \) an action \( a \) can be performed and after the action the state moves to \( s' \)."

3 Language for Concurrent process Calculi: LCC

LCC (Language of Concurrent process Calculi) is a meta language designed to formally describe concurrent processes based on structural operational semantics, represented as transition rules, of various process calculi. Usually a process calculus is provided by the operators used in the calculus and the transition rules which characterize the nature of it. In LCC, an operator is represented in the style of Abstract Data Type, and a transition rule in the formal logic which regards transition relations as formulae.