High Level Expressions with their SOS Semantics*
– Extended Abstract –

Hanna Klaudel¹ and Robert-C. Riemann²

¹ LRI, Université Paris-Sud, and IUT Paris XII, France
² LRI, Université Paris-Sud, France, and Universität Hildesheim, Germany

Abstract. A process algebra, called *M-expressions*, based on parameterised (multi-)actions is introduced. In the context of the *Petri Box Calculus*, it serves as a kind of high level counterpart to *Box expressions*. An operational semantics based on step sequences, similar to that given by Best, Esparza and Koutny for *Box expressions* is defined. The consistency and completeness of the semantics is proved with respect to the elementary case. The process algebra is applied as the semantic domain for a concurrent programming language. The consistency and completeness of the operational semantics is proved for this application with respect to an existing high level Petri net semantics.

1 Introduction

When modelling concurrent systems, the communication can be achieved in various ways. One way is the *handshake* communication, i.e., the direct transmission of values (as the rendezvous of Ada, input/output guards of (T)CSP [9] or synchronisations in CCS [12]). Another way is the common access to variables shared by two or more subprograms. One can also consider FIFO or LIFO buffers which are asynchronous versions of handshake communications. All these modes of communication are expressible in the *Petri Box Calculus* (PBC) [2], which can be seen for this respect as the common denominator for other process algebras.

The PBC is a basic process algebra which combines a number of ideas taken from CCS, CSP, ACP [1], etc., and provides a compositional Petri net semantics. Although the PBC has not specifically been defined as a common generalisation of other formalisms, it turns out that they can essentially be encoded in the PBC.

Originally, the PBC comprised a process algebra, called *Box expressions*, with a semantic domain of place/transition nets, called *Petri Boxes*. In the meantime a calculus based on high level Petri nets, called *M-nets* [5], has been introduced in order to cope with the net size problem encountered for Petri Boxes. The operations of M-nets are provably consistent in the strictest possible sense (through

* Supported by the French-German project POEM.
net unfoldings) [5]. Also, the introduction of M-nets (high level version of Petri Boxes) solved successfully the size explosion problem by giving "abbreviated" and more readable nets. In particular, it became possible to give a compositional (finite) net semantics to concurrent programming languages comprising various basic data types and process communication structures.

In this paper we propose a process algebra, called \textit{M-expressions}, which can be seen as the high level counterpart of Box expressions. Its basic constituents are parameterised guarded actions and operators such as parallel composition (\(|\|\)), sequence (;), choice (\(\triangleright\)), synchronisation (sy), restriction (rs), and iteration ([t\textgreater t\textgreater]). Each of them has basically its intuitive semantics known from PBC (and other process algebras). Like in the PBC, the synchronisation operator plays a crucial rôle in the formalism. Furthermore, a step sequential operational semantics in Plotkin style is given for the M-expressions algebra. On one hand, it allows to apply verification techniques for transition systems. On the other hand, it permits the comparisons of M-expressions with other models. Here, we characterise the relationship to M-nets and Box expressions.

Section 2 gives the syntax of M-expressions, defines their operational semantics, and concludes with results concerning the synchronisation operator and a comparison of Box expressions and M-expressions. Section 3 applies M-expressions as the semantic domain for a parallel programming language, which is also equipped with a high level Petri net semantics. The main results is the coherence of the M-expression and the existing Petri net based program semantics. This article is the extended abstract of [10]. Proofs, formal definitions concerning the M-net model, and examples can be found there.

2 Algebra of M-expressions

2.1 Notations and Preliminaries

Let \textit{Val} be a fixed but suitable large set of values, and \(S\) and \(F\) disjoint sets of storing, respectively free variables; \(S \cup F\) will be abbreviated by \textit{Var}. In the set \textit{Op} of operators we have, e.g., = to denote the equality of values, and operators like not, and, or, +, −,... The set of all well-formed terms over \textit{Val}, \textit{Var}, and \textit{Op} is denoted by \textit{T}.

We assume the existence of a fixed but sufficiently large set \(A\) of actions. Each action \(A \in A\) is assumed to have an arity \(ar(A)\) which is a natural number describing the number of its parameters. The set \(A\) is, by definition, the carrier of a bijection:\(\sim: A \to A\), called \textit{conjugation}, satisfying \(\forall A \in A : \hat{A} \neq A \land \hat{\hat{A}} = A\). That is, the mapping \(\sim\) groups the elements of \(A\) into pairwise conjugates. Conjugated actions will be used like in CCS to express synchronisation capabilities. It is assumed that \(\forall A \in A : ar(A) = ar(\hat{A})\).

A construct \(A(\tau_1,...,\tau_{ar(A)})\), where \(A\) is an action and \(\forall j : 1 \leq j \leq ar(A) : \tau_j \in \textit{Var} \cup \textit{Val}\), is a parameterised action. The set of all parameterised actions is denoted by \(P\). A parameterised action \(A(\tau_1,...,\tau_{ar(A)})\), where \(\forall j : 1 \leq j \leq ar(A) : \tau_j \in \textit{Val}\), is called elementary. In that case we will abbreviate it as \(A(\overline{\tau})\).