Compositional Process Semantics of Petri Boxes\(^1\)

Eike Best\(^2\) and Hans-Günther Linde-Göers\(^2\)

Abstract
The Petri Box algebra defines a linear notation to express a structured class of Petri nets which can be seen as a modification and generalisation of Milner's CCS. The calculus has been designed as an intermediate stage in the compositional translation of higher level concurrent programming notations into Petri nets. This paper defines the notion of a 'Box process' intended to capture the (Petri net) partial order semantics of the Box algebra. The main result is the equivalence of the direct compositional semantics so defined, and the indirect non-compositional semantics which uses processes of Petri nets, for a class of expressions.

1 Introduction

The Petri Box Calculus (PBC [5]), which has been developed in the Esprit Basic Research Action DEMON, is a blend which is partially derived from existing calculi (notably, Milner's CCS [23]) and is partially novel. It was designed to satisfy two requirements. Firstly, it should be firmly based on a Petri net semantics, and secondly, it should be oriented towards easing the compositional definition of the semantics of various concurrent programming languages such as occam [21], including all data aspects; it has been discussed in [6, 18] how this can be achieved.

Compared with CCS, the PBC features a different synchronisation operator and a refinement operator. Moreover, the PBC is not prefix-driven but, on the contrary, treats entry and exit points of processes symmetrically. As a consequence, the sequence operator is basic and the recursion operator is much more general and not limited to tail-end recursion.

Up to the present time, there have been various developments concerning the Box calculus; they have been chiefly oriented towards its static aspects. In [5], a number of static equivalences that can be derived as a consequence of the static semantics have been established; in [4], static (denotational) definitions have been given for refinement and recursion; in [12], the S-invariant covering of Boxes has been investigated.

This present paper and its companion papers [7, 20], by contrast, address the dynamic aspects of Box expressions. The aim of the present paper is to make a domain out of the set of (Petri net) processes and to define a compositional

\(^1\)Work done within the Esprit Basic Research Action 3148 DEMON (Design Methods Based on Nets) and the Working Group 6067 CALIBAN (Causal Calculi Based on Nets).

\(^2\)Institut für Informatik, Universität Hildesheim, Marienburger Platz 22, D-31141 Hildesheim, {e.best,linde}@informatik.uni-hildesheim.de
process semantics of Box expressions on this domain. The ground set is defined as the sets of (equivalence classes of) processes, denoting all possible concurrent runs of an expression. The operations on the domain mirror the Box expression operators. The main result establishes the consistency of the Box process semantics (the 'direct' semantics) and the set of processes that can be obtained indirectly by first deriving the Box of an expression and then the processes of this Box using the standard net theoretical notion [3, 17] (the 'indirect' semantics).

The operations we shall define on Box processes have been inspired by prior work such as that of Cherkasova and Kotov [10] and Pratt [25]. The specific form of our operations, which significantly differ from the ones used in the cited papers, has been motivated by [5] and other work on the Box Calculus. This work is also pertinent to a large body of recent work on giving Petri net semantics of existing process calculi such as CCS, CSP [19] or ACP [1] (for instance, [2, 9, 10, 11, 14, 15, 22, 24, 26]).

The organisation is as follows. Section 2 explains the syntactic domain of Box expressions. Section 3 defines the basic elements of the semantic domains we are going to consider: labelled nets, labelled causal nets, Petri Boxes and Box processes. Section 4 defines the first semantic domain, namely the domain of Box processes. Section 5 describes the second semantic domain, the domain of Petri Boxes. Section 6 deals with consistency between the direct semantics and the indirect semantics. The main result of section 6 establishes the equivalence of these two notions. Section 7 contains concluding remarks.

2 The Syntactic Domain: Box Expressions

Action names and variable names are the basic constituents of the Box expression algebra. We assume a set of action names, $A$, to be given. On $A$, we assume a conjugation bijection to be defined: $\hat{\cdot}: A \rightarrow A$ with $\hat{a} \neq a$ and $\hat{\hat{a}} = a$ for all $a \in A$. The set $\mathcal{L}$ of finite multisets over $A$ is called the set of communication labels. Elements $\alpha$ of the set $\mathcal{L}$ may serve as the labels of transitions and events. The function $\hat{\cdot}$ can be extended to any multiset $\mu$ over $A$ by element-wise application. When $\alpha$ is a singleton set $\{a\}$, we omit the enclosing set brackets if unambiguity is ensured. We use capital letters $X, Y$ etc. to denote Box expression variables which are used for refinement and recursion. Let $\mathcal{V}$ denote the set of such variable names. Elements of $\mathcal{V}$ may also serve as transition or event labels.

Using these conventions, Table 1 defines the Box expression syntax which we consider in this paper - which is the full syntax considered in [5] except for scoping and relabelling. Scoping is a derived operator from synchronisation and restriction (and its semantics follows accordingly). Relabelling is omitted here since its treatment complicates the formalism but presents no specific difficulties.