Mechanizing a $\pi$-calculus equivalence in HOL

Otmane AÏT MOHAMED*

CRIN-CNRS & INRIA-Lorraine, BP 239, Vandœuvre-lès-Nancy Cedex, France

Abstract. The $\pi$-calculus is a relatively simple framework in which the semantics of dynamic creation and transmission of channels can be described nicely. In this paper we consider the issue of verifying mechanically the equivalence of $\pi$-terms in the context of bisimulation based semantics while relying on the general purpose theorem prover HOL. Our main contribution is the presentation of a proof method to check early equivalence between $\pi$-terms. The method is based on $\pi$-terms rewriting and an operational definition of bisimulation. The soundness of the rewriting steps relies on standard algebraic laws which are formally proved in HOL. The resulting method is implemented in HOL as an automatic tactic.

1 Introduction

The $\pi$-calculus [12] is an extension of CCS [10] based on the idea that processes can communicate channel names. This possibility dramatically increases the expressive power of the calculus, for instance it allows one to model networks with a dynamically changing topology [14], and reasonable encodings of the $\lambda$-calculus and of higher-order process calculi have been proposed [11, 2, 15].

This paper reports on work concerning the mechanical verification of equivalence between $\pi$-terms in the context of bisimulation based semantics within the general purpose theorem prover HOL. This work is based on the one described in [3], and is constructed on top of our mechanization of the $\pi$-calculus theory in the HOL system [1].

The embedding of the $\pi$-calculus in HOL is inspired by previous works on the mechanization of process algebra in HOL, namely, the mechanization of CSP by Camilleri [6], the mechanization of CCS by Nesi [13], and, recently, the mechanization of the $\pi$-calculus by Melham [9].

Our general goal is two fold: firstly, we want to develop formally the theory of the $\pi$-calculus and secondly, we want to apply this framework to the verification of applications specified in the $\pi$-calculus.

In the proof construction process it is of the utmost importance to have tactics that carry out simple parts of the proof automatically. In particular our goal here is to define a tactic that can solve automatically the equivalence problem for finite terms (a term is said to be finite if it contains only finite summations, and no recursion). More precisely, the main contribution of this

* email:amohamed@loria.fr
paper is to describe a method of checking early equivalence between finite \( \pi \)-
terms, and its implementation in HOL (the method can be applied to general
\( \pi \)-terms as well but in this case termination is not guaranteed). This method is
composed of two basic parts:

- In the first part a \( \pi \)-term is rewritten into a \textit{prefixed form} where it is possible
to read directly one of its next actions (if any). More precisely, a process \( P \) is
said to be in a prefixed form if it is \( \text{Nil} \), or it is in one of the two forms: \( \alpha . P \)
or \( \alpha . P + Q \), where \( \text{Nil} \) is the terminated process, \( \alpha \) is a suitable prefix, and
\( + \) is the non-deterministic sum. The rewriting rules are obtained by forcing
a suitable orientation of basic algebraic laws of the \( \pi \)-calculus, typically we
apply a suitable form of the expansion theorem and a certain number of
rules concerning the commutation of restriction with the other operators.
Since the algebraic laws have been formally derived in our HOL’s formalisa-
tion of the \( \pi \)-calculus [1] we are able to derive easily the soundness of the
rewriting process. Let us anticipate that, in order to represent certain inter-
mediate states of the computation we employ a few new operators. These
auxiliary operators give an equational characterization of parallel composi-
tion. They were introduced for the first time by Bergstra and Klop in their
finite axiomatization of strong bisimulation equivalence over ACP [4, 5]. This
in turn leads to transformational proof techniques for showing that a pro-
cess implementation meets its specification. These operators are introduced
in HOL’s \( \pi \)-calculus theory by following the \textit{definitional} principle, in order to
ensure the consistency of this extension. Related algebraic laws are derived
formally and applied to the rewriting process.

- In the second part two prefixed forms are compared according to a suit-
able set of rules. These rules are based on the definition of the bisimulation
relation. Their soundness is shown within the HOL system.

In general, our tactic alternates the computation of prefixed forms (part 1)
and their comparison according to the rules of bisimulation (part 2).

The structure of the paper is as follows: In section 2 we give a brief presenta-
tion of the \( \pi \)-calculus: its syntax, its semantics and the definition of the strong
early equivalence. In section 3 we recall some aspects of the formalisation of the
\( \pi \)-calculus in HOL. In section 4 we give the definition of our proof method, fol-
lowed by its representation in the HOL system. We conclude with some remarks
and some directions for future work.

2 \( \pi \)-calculus

In this section, we present the syntax and the semantic of the \( \pi \)-calculus, as
well as the definition of the strong early equivalence. For further details on these
topics we refer to [12].

The syntax of the \( \pi \)-calculus is given by the following BNF grammar: