Model Checking for \( \pi \)-Calculus
Using Proof Search

Alwen Tiu

INRIA Lorraine, 615 rue du Jardin Botanique,
54602 Villers-lès-Nancy, France
Alwen.Tiu@loria.fr

Abstract. Model checking for transition systems specified in \( \pi \)-calculus has been a difficult problem due to the infinite-branching nature of input prefix, name-restriction and scope extrusion. We propose here an approach to model checking for \( \pi \)-calculus by encoding it into a logic which supports reasoning about bindings and fixed points. This logic, called \( FO_{\lambda \Delta \nabla} \), is a conservative extension of Church’s Simple Theory of Types with a “generic” quantifier. By encoding judgments about transitions in \( \pi \)-calculus into this logic, various conditions on the scoping of names and restrictions on name instantiations are captured naturally by the quantification theory of the logic. Moreover, standard implementation techniques for (higher-order) logic programming are applicable for implementing proof search for this logic, as illustrated in a prototype implementation discussed in this paper. The use of logic variables and eigenvariables in the implementation allows for exploring the state space of processes in a symbolic way. Compositionality of properties of the transitions is a simple consequence of the meta theory of the logic (i.e., cut elimination). We illustrate the benefits of specifying systems in this logic by studying several specifications of modal logics for \( \pi \)-calculus. These specifications are also executable directly in the prototype implementation of \( FO_{\lambda \Delta \nabla} \).

1 Introduction

The \( \pi \)-calculus [16] provides a simple yet powerful framework for specifying communication systems with evolving communication structures. Its expressiveness derives mainly from the possibility of passing communication channels (names), restricting the scope of channels and scope extrusion. These are precisely the features that make model checking for \( \pi \)-calculus difficult. Model checking has traditionally been done with transitions which have finite state models. The name passing feature alone (input prefix) in \( \pi \)-calculus would yield infinite-branching transition systems, if implemented naively. Scope and scope extrusion add another significant layer of complexity, since in model checking the transition systems one has to take into account the exact scope and identity of various channel names. This is a problem which has been studied extensively, of course, due to the importance of \( \pi \)-calculus. A non-exhaustive list of existing

© Springer-Verlag Berlin Heidelberg 2005
works includes the work on history dependent automata [6] model of mobile processes, specific programming logics and decision procedures for model checking mobile processes [34], the spatial logic model checker [2] using Gabbay-Pitts permutation techniques [7], and implementation using logic programming [27].

The approach to model checking π-calculus (or mobile processes in general) taken in this paper is based on the proof theory of sequent calculus, by casting the problem of reasoning about scoping and name-instantiation into the more general setting of proof theory for quantifiers in formal logic. More specifically, we encode judgments about transitions in π-calculus and several modal logics for π-calculus [17] into a meta logic, and proof search is used to model the operational semantics of these judgments. This meta logic, called \( FO\lambda^{\Delta\nabla} \) [15], is an extension of Church’s Simple Theory of Types (but without quantification over propositions, so the logic is essentially first-order) with a proof theoretical notion of definitions [22] and a new “generic” quantifier, \( \nabla \). The quantifier \( \nabla \), roughly summarized, facilitates reasoning about binders (more details will be given later). We summarize our approach as follows.

**λ-tree syntax.** We use the λ-tree syntax [14] to encode syntax with bindings. It is a variant of higher-order abstract syntax, where syntax of arbitrary system is encoded as λ-terms and the λ-abstraction is used to encode bindings within expressions. One of the advantages of adopting λ-tree syntax, or higher-order abstract syntax in general, is that all the side conditions involving bindings such as scoping of variables, α-conversion, etc., are handled uniformly at the level of the abstract syntax, using the known notions in λ-calculus. Another one is that efficient implementation techniques for manipulating this abstract syntax are well-understood, e.g., algorithms for doing pattern-matching and unification of simply typed λ-terms.

**Definitional reflection.** Proof search in traditional logics, e.g., variants of Gentzen’s LJ or LK, is limited to model the may-behaviour of computation system. Must-behaviour, eg., notions like bisimulations, or in the interest of this paper, satisfiability of modal formulae, cannot be expressed directly in these logics. To encode such notions, it is necessary to move to a richer logic. Recent developments in the proof theory of definitions [10,11] have shown that must-behaviour can indeed be captured in logics extended with this proof-theoretical notion of definitions. In a logic with definitions, an atomic proposition may be “defined” by another formula (which may contain the atomic proposition itself). Thus, a definition can be seen as expressing a fixed point equation. Proof search for a defined atomic formula is done by unfolding the definition of the formula. In the logic with definitions used in this paper, a provable formula like \( \forall x.p \mathcal{R} q \), where \( p \) and \( q \) are some defined predicates, expresses the fact that for every term \( t \) and for every proof (computation) of \( pt \), there is a proof (computation) of \( qt \). If \( p \) and \( q \) are predicates encoding one-step transitions, then this formula expresses one-step simulation. If \( q \) is an encoding of some assertion in modal logics, then the formula expresses the fact that the modal assertion is true for all reachable “next states” associated with the transition relation encoded by \( p \).