Exploiting Algebraic Laws to Improve Mechanized Axiomatizations

Luca Aceto\textsuperscript{1}, Eugen-Ioan Goriac\textsuperscript{1}, Anna Ingolfsdottir\textsuperscript{1}, Mohammad Reza Mousavi\textsuperscript{2}, and Michel A. Reniers\textsuperscript{3}

\textsuperscript{1} ICE-TCS, School of Computer Science, Reykjavik University, Menntavegur 1, IS-101, Reykjavik, Iceland
\textsuperscript{2} Center for Research on Embedded Systems (CERES), Halmstad University, Sweden
\textsuperscript{3} Department of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

Abstract. In the field of structural operational semantics (SOS), there have been several proposals both for syntactic rule formats guaranteeing the validity of algebraic laws, and for algorithms for automatically generating ground-complete axiomatizations. However, there has been no synergy between these two types of results. This paper takes the first steps in marrying these two areas of research in the meta-theory of SOS and shows that taking algebraic laws into account in the mechanical generation of axiomatizations results in simpler axiomatizations. The proposed theory is applied to a paradigmatic example from the literature, showing that, in this case, the generated axiomatization coincides with a classic hand-crafted one.

1 Introduction

Algebraic properties, such as commutativity, associativity and idempotence of binary operators, specify some natural properties of programming and specification constructs. These properties can either be validated using the semantics of the language with respect to a suitable notion of program equivalence, or they can be guaranteed a priori ‘by design’. In particular, for languages equipped with a Structural Operational Semantics (SOS) \cite{19}, there are two closely related lines of work to achieve this goal: firstly, there is a rich body of syntactic rule formats that can guarantee the validity of certain algebraic properties; see \cite{5,17} for recent surveys. Secondly, there are numerous results regarding the mechanical generation of ground-complete axiomatizations of various behavioral equivalences and preorders for SOS language specifications in certain formats—see, e.g., \cite{17,20}. However, these two lines of research have evolved separately and no link has been established between the two types of results so far. In this paper, we take the first steps in marrying these two research areas and in using rule formats for algebraic properties (specifically, for commutativity) to enhance the...

\* The first three authors have been partially supported by the project ‘Meta-theory of Algebraic Process Theories’ (nr. 100014021) of the Icelandic Research Fund. Eugen-Ioan Goriac is also funded by the project ‘Extending and Axiomatizing Structural Operational Semantics: Theory and Tools’ (nr. 1102940061) of the Icelandic Research Fund.
process of automatic generation of axiomatizations for strong bisimilarity from GSOS language specifications [10]. In particular, we show that linking these two areas results in axiomatizations that look like hand-crafted ones.

Contribution and Related Work. Many ground-completeness results have been presented in the literature on process calculi. (See, for instance, the survey paper [3] for pointers to the literature.) A common proof strategy for establishing such ground-completeness results is to reduce the problem of axiomatizing the notion of behavioural equivalence under consideration over arbitrary closed terms to that of axiomatizing it over ‘synchronization-tree terms’. This approach is also at the heart of the algorithm proposed in [1] for the automatic generation of finite, equational, ground-complete axiomatizations for bisimilarity over language specifications in the GSOS format. A variation on that algorithm for GSOS language specifications with termination has been presented in [7]. In [20], Ulidowski has instead offered algorithms for the automatic generation of finite axiom systems for the testing preorder over de Simone process languages. In Section 4 of this paper, we present a refinement of the algorithm from [1] that uses a rule format guaranteeing commutativity of certain operators to obtain ground-complete axiomatizations of bisimilarity that are closer to the hand-crafted ones than those produced by existing algorithms. (See Section 5 where we apply the algorithm to axiomatize the classic parallel composition operator and compare the generated axiomatization to earlier ones.)

Our rule format for commutativity (presented in Section 3) is a generalization of the rule format for commutativity from [16], which allows operators to have various sets of commutative arguments. Apart from being natural, such a generalization is useful in the automatic generation of ground-complete axiomatizations, as the developments in this study show.

2 Preliminaries

In this section we review, for the sake of completeness, some standard definitions from process theory and the meta-theory of SOS that will be used in the remainder of the paper. We refer the interested reader to [4,17] for further details.

Transition System Specifications in GSOS Format. We let \( V \) denote an infinite set of variables with typical members \( x, x', x_1, y, y', y_i, \ldots \). A signature \( \Sigma \) is a set of function symbols, each with a fixed arity. We call these symbols operators and usually represent them by \( f, g, \ldots \). An operator with arity zero is called a constant. We define the set \( T(\Sigma) \) of terms over \( \Sigma \) (sometimes referred to as \( \Sigma \)-terms) as the smallest set satisfying the following constraints: (1) A variable \( x \in V \) is a term. (2) If \( f \in \Sigma \) has arity \( n \) and \( t_1, \ldots, t_n \) are terms, then \( f(t_1, \ldots, t_n) \) is a term. We use \( s, t, t', t_i, u, \ldots \) to range over terms. We write \( t_1 \equiv t_2 \) if \( t_1 \) and \( t_2 \) are syntactically equal. The function \( \text{vars} : T(\Sigma) \to 2^V \) gives the set of variables appearing in a term. The set \( C(\Sigma) \) is the set of closed terms, i.e., the set of all terms \( t \) such that \( \text{vars}(t) = \emptyset \). We use \( p, p', p_i, q, r, \ldots \) to range over closed terms. A substitution \( \sigma \) is a function of type \( V \to T(\Sigma) \). We extend the domain of substitutions to terms homomorphically. If the range of a substitution lies in \( C(\Sigma) \), we say that it is a closed substitution.