Bisimulation Equivalence is Decidable for all Context-Free Processes

Søren Christensen* Hans Hüttel† Colin Stirling*

1 Introduction

Over the past decade much attention has been devoted to the study of process calculi such as CCS, ACP and CSP [11]. Of particular interest has been the study of the behavioural semantics of these calculi as given by labelled transition graphs. One important question is when processes can be said to exhibit the same behaviour, and a plethora of behavioural equivalences exists today. Their main rationale has been to capture behavioural aspects that language or trace equivalences do not take into account.

The theory of finite-state systems and their equivalences can now be said to be well-established. There are many automatic verification tools for their analysis which incorporate equivalence checking. Sound and complete equational theories exist for the various known equivalences, an elegant example is [16].

One may be led to wonder what the results will look like for infinite-state systems. Although language equivalence is decidable for finite-state processes, it is undecidable when one moves beyond finite automata to context-free languages. For finite-state processes all known behavioural equivalences can be seen to be decidable. In the setting of process algebra, an example of infinite-state systems is that of the transition graphs of processes in the calculus BPA (Basic Process Algebra) [3]. These are recursively defined processes with nondeterministic choice and sequential composition.

A special case is that of normed BPA processes. A process is said to be normed if it can terminate in finitely many steps at any point during the execution. Even though normed BPA does not incorporate all regular processes, systems defined in this calculus can in general have infinitely many states. A recent result shows that strong bisimulation equivalence for normed BPA processes is decidable. A number of proofs of this result exists, the original proof by Baeten, Bergstra and Klop [1], another due to Caucacl [7] (see [9] too), and a third due to Hüttel and Stirling [14] (with more details in [13]) which appeals to tableaux. The tableau based approach supports a sound and complete equational theory for normed BPA. On the other hand, Huynh and Tian

*Laboratory for Foundations of Computer Science, James Clerk Maxwell Building, University of Edinburgh, Edinburgh EH9 3JZ, Scotland.
†Department of Mathematics and Computer Science, Aalborg University Centre, Fredrik Bajersvej 7E, 9220 Aalborg Ø, Denmark.
[15] and Groote and Hüttel [10] have proved that all other standard equivalences are undecidable for normed BPA and thus for BPA in general.

One remaining question to be answered is whether bisimulation equivalence is decidable for the full BPA language. In this paper we answer this question in the affirmative, using a technique inspired by Cancal's proof of the decidability of language equivalence for simple algebraic grammars [5]. Section 2 contains preliminary definitions. Section 3 describes the main result, namely that the maximal bisimulation of any BPA transition graph is generable from a finite self-bisimulation relation.

2 BPA processes

The class of guarded recursive BPA (Basic Process Algebra) processes [1, 3] is defined by the following abstract syntax

\[ E ::= a \mid X \mid E_1 + E_2 \mid E_1 \cdot E_2 \]

Here \( a \) ranges over a set of atomic actions \( Act \), and \( X \) over a family of variables. The operator + is nondeterministic choice while \( E_1 \cdot E_2 \) is the sequential composition of \( E_1 \) and \( E_2 \) – we usually omit the \( \cdot \). A BPA process is defined by a finite system of recursive process equations

\[ \Delta = \{X_i \overset{def}{=} E_i \mid 1 \leq i \leq k\} \]

where the \( X_i \) are distinct, and the \( E_i \) are BPA expressions with free variables in \( \text{Var}_\Delta = \{X_1, \ldots, X_k\} \). One variable (generally \( X_1 \)) is singled out as the root. Usually one considers relations within the transition graph for a single \( \Delta \). This can be done without loss of generality, since we can let \( \Delta \) be the disjoint union of any pair \( \Delta_1 \) and \( \Delta_2 \) that we wish to compare (with suitable renamings of variables, if required).

We restrict our attention to guarded systems of recursive equations.

**Definition 2.1** A BPA expression is guarded if every variable occurrence is within the scope of an atomic action. The system \( \Delta = \{X_i \overset{def}{=} E_i \mid 1 \leq i \leq k\} \) is guarded if each \( E_i \) is guarded for \( 1 \leq i \leq k \).

We use \( X, Y, \ldots \) to range over variables in \( \text{Var}_\Delta \) and Greek letters \( \alpha, \beta, \ldots \) to range over elements in \( \text{Var}_\Delta^* \). In particular, \( \epsilon \) denotes the empty variable sequence.

**Definition 2.2** Any system of process equations \( \Delta \) defines a labelled transition graph. The transition relations are given as the least relations satisfying the following rules:

\[
\begin{align*}
a & \xrightarrow{a} \epsilon, \ a \in Act \\
E & \xrightarrow{a} E' & X & \overset{def}{=} E \in \Delta \\
E \xrightarrow{a} E' & X \xrightarrow{a} E' & \text{if } E' \neq \epsilon \\
E + F & \xrightarrow{a} E' & E + F & \xrightarrow{a} F' \\
E & \xrightarrow{a} E' & E + F & \xrightarrow{a} F' \\
EF & \xrightarrow{a} E'F & E'F & \xrightarrow{a} F
\end{align*}
\]

**Example 2.1** Consider the system \( \Delta = \{X \overset{def}{=} a + bXY; \ Y \overset{def}{=} c\} \). By the transition rules in Definition 2.2 \( X \) generates the transition graph in Figure 1. 

\[ \qed \]