On the Ambiguity, Finite-Valuedness, and Lossiness Problems in Acceptors and Transducers *

Oscar H. Ibarra

Department of Computer Science
University of California, Santa Barbara, CA 93106, USA
ibarra@cs.ucsb.edu

Abstract. We prove new decidability and undecidability results concerning the finite-ambiguity problem in acceptors, and the finite-valuedness and lossiness problems in transducers. The acceptors and transducers we study have infinite memory.

Keywords: acceptors, transducers, ambiguous, finite-valued, lossy.

1 Introduction

It is well-known that it is undecidable, given an NPDA whose stack makes only one reversal (or, equivalently, a linear CFG), whether it is unambiguous. It is also known that its unbounded ambiguity problem is undecidable [17]. Here, we show that it is undecidable, given a nondeterministic counter automaton (NCA), whether it is unambiguous (resp., has unbounded ambiguity). We also show that in the special case when the counter of the NCA makes at most \( r \) reversals (i.e., alternations between increasing and decreasing modes) for a given \( r \geq 1 \), determining whether it is unambiguous (resp., \( k \)-ambiguous for a given \( k \)) is decidable. However, deciding if an \( r \)-reversal NCA has unbounded ambiguity is open.

We then turn our attention to transducers. We study the questions of “finite-valuedness” and “finite-lossiness” of a transducer and their connections to the ambiguity of the underlying acceptor (i.e., the acceptor obtained by deleting the outputs).

A transducer \( T \) of a given type is \( k \)-valued \((k \geq 1)\) if every accepted input \( u \) is mapped into at most \( k \) distinct outputs. \( T \) is finite-valued if it is \( k \)-valued for some \( k \). Similarly, a transducer is \( k \)-lossy \((k \geq 1)\) if for every output \( v \), there are at most \( k \) distinct accepted inputs that are mapped into \( v \). \( T \) is finitely-lossy if it is \( k \)-lossy for some \( k \).

We prove decidable and undecidable results concerning the finite-valuedness and finite-lossiness problems for transducers with infinite memory. Similar problems have been investigated before, e.g., for nondeterministic finite transducers (NFTs) [1,14,16] and visibly pushdown transducers [4]. In [16], e.g., it was

* Supported in part by NSF Grant CCF-1117708.
shown that finite-valuedness of NFTs is decidable. In [4], it was shown that it is decidable if a visibly pushdown transducer is \( k \)-valued for a given \( k \). We show a similar result for 1-ambiguous pushdown transducers. (We note that as language acceptors, 1-ambiguous pushdown automata are more powerful than visibly pushdown automata.) The question of whether a transducer with infinite memory is finite-valued (i.e., \( k \)-valued for some \( k \)) has not been addressed before, as far as we know. Here, we exhibit such a class with decidable finite-valuedness problem. The result concerns linear context-free grammars (LCFGs) with outputs, where we show that the finite-valuedness problem for LCFGs with outputs (thus, a finite set of output strings is associated with the application of each rule) is decidable if the LCFG is finitely-ambiguous. If the LCFG is ambiguous, the problem becomes undecidable. The decidability of finite-valuedness generalizes to finitely-ambiguous nonterminal-bounded CFGs with outputs.

Finally, we give a strong undecidability result concerning the equivalence problem for nondeterministic 2-tape finite automata. Note that a binary relation is accepted by a 2-tape finite automaton if and only if the relation is defined by an NFT.

We will use the following notation throughout the paper: NPDA for nondeterministic pushdown automaton; DPDA for deterministic pushdown automaton; NCA for an NPDA that uses only one stack symbol in addition to the bottom of the stack, which is never altered (thus, the stack is a counter); DCA for deterministic NCA; NFA for nondeterministic finite automaton; DFA for deterministic finite automaton; 2NFA for two-way NFA (with end markers). Formal definitions can be found in the book [9].

A counter is an integer variable that can be incremented by 1, decremented by 1, left unchanged, and tested for zero. It starts at zero and cannot store negative values. Thus, a counter is a pushdown stack on unary alphabet, in addition to the bottom of the stack symbol which is never altered.

An automaton (NFA, NPDA, NCA, etc.) can be augmented with multiple counters, where the “move” of the machine also now depends on the status (zero or non-zero) of the counters, and the move can update the counters. See [10] for formal definitions. It is well known that a DFA augmented with two counters is equivalent to a Turing machine (TM) [13].

In this paper, we will restrict the augmented counter(s) to only “reverse” once, i.e., once it decrements, it can no longer increment. Thus, e.g., each counter in an NPDA with 1-reversal counters makes only one reversal. Note that a counter that makes \( r \) reversals can be simulated by \( \lceil \frac{r+1}{2} \rceil \) 1-reversal counters.

## 2 Ambiguity in Acceptors

An acceptor \( M \) of any type (e.g., NFA, NPDA, etc.) is \( k \)-ambiguous \((k \geq 1)\) if every accepted string can be accepted in at most \( k \) distinct accepting computations. Note that 1-ambiguous is the same as unambiguous. \( M \) is finitely-ambiguous if it is \( k \)-ambiguous for some \( k \); otherwise, it has unbounded ambiguity.

In this section, we look at the unambiguity and unbounded ambiguity problems concerning 1-reversal NPDAs (i.e., once the stack pops, it can no longer