An Abstract Machine Theory for Formal Language Parsers*

David B. Benson

Received January 22, 1973

Summary. The usual data necessary for any abstract machine theory is given in categorical terminology. In these terms, an abstract machine theory for formal language parsers is developed, exposing the essential nature of any left-to-right parsing scheme. A weak classification of all parsers for a given language is developed and the usual notions of initial machine, reachable machine and minimal machine apply. Minimality is an extremely weak notion in this theory, although it is equivalent to a simple form of immediate error detection for parsers. Remarks on the construction of parsing procedures are given.

0. Introduction

We abstract the essentials of formal language parsing into an algebraic setting. The abstraction provides a machine theoretic view of the computations performed by a parsing procedure. The abstraction is successful in establishing a machine theoretic development of parsers which faithfully follows the standard sequential machine development and exposes the essential nature of any left-to-right parsing scheme. While the technical content of this note can be read without knowledge of actual parsing procedures, the examples are rather more meaningful in comparison with the exposition in [1].

The usual notion of minimality is abstracted for this setting, and it is shown that there is a minimal abstract parser for each formal language. Since parsers for non-regular languages must have an infinite state set, this notion of minimality is closely associated with the notions of immediate error detection in a parser. This latter property is important for practical parsing algorithms.

Given two parsers for the same grammar, one is "less minimal" than the other if, in essence, there is an surjective function from the state set of the first to the state set of the second. The classification of parsers this notion provides is again quite weak. Nonetheless, it is currently the only known classification of the set of all left-to-right parsers for a given grammar.

The setting is categorical and the reader will find portions of MacLane [14], or the first chapter of Pareigis [15], a desirable background. We include certain definitions from category theory since our notation differs slightly from that in either of the above references.

Categories are denoted by boldface capital letters, \( I, X \), etc., and by script capital letters, \( \mathcal{P}, \mathcal{X} \). The object class of a category \( C \) is denoted \( \text{Obj}(C) \). The set of morphisms in category \( C \) with domain \( A \) and codomain \( B \) is denoted \( \text{Mor}_C(A, B) \). The identity morphism at the object \( A \) is denoted \( 1_A \), although it is extremely convenient to identify \( A \) with its identity morphism. Then we write \( A \) for the object \( A \) or the morphism \( 1_A \), context determining which is meant. Composition is denoted by \( \circ \) or by juxtaposition and is always in the order of following arrows. That is, if \( f: A \to B \) and \( g: B \to C \) are morphisms of \( C \), the

* Research supported in part by grant GJ-1171 from the National Science Foundation.
composition is denoted \( f \circ g \) or \( fg \). For each category \( C \), the opposite category to \( C \) is denoted \( C^0 \). \( f: A \to B \) is a morphism of \( C \) if \( f^0: B \to A \) is a morphism of \( C^0 \). The identity functor at the category \( C \) is denoted \( 1_C \).

\( \mathcal{S} \) is a category of sets and functions. \( \emptyset \) denotes both the null set and the null function.

1. Abstract Machine Theories

An abstract machine theory consists of the following data.

1. A category of input objects, \( I \). Each input object has semigroup structure in the following way. First, a method of forming “product objects” is given, the products being associative, at least up to isomorphism. This amounts to giving an associative bifunctor, \( \otimes: I \times I \to I \). Typically this bifunctor is Cartesian product. For notational ease and to avoid details which would detract from the main development, we assume strict associativity of all products in all that follows. It is convenient to insist on an identity \( e \) for \( \otimes \) so that \( I \) equipped with \( \otimes \) is a strict monoidal category, \( (I, \otimes, e) \). (Benabou [2], MacLane [13, 14].) Strict monoidal categories are called \( \mathcal{X} \)-categories by Hotz [10].

For sequential machines, each object of \( I \) is a free monoid \( \Sigma^* \) over a set of generators \( \Sigma \). The morphisms of \( I \) are the homomorphisms between the free monoids.

Second, for each input object \( I \in \text{Obj}(I) \), a morphism \( \mu: I \otimes I \to I \) is given. \( (I, \mu) \) is a semigroup as follows. \( \mu \) is required to satisfy the associativity axiom, 

\[
(1_I \otimes \mu) \mu = (\mu \otimes 1_I) \mu.
\]

The commuting diagram for the axiom is

\[
\begin{array}{ccc}
I \otimes I \otimes I & \xrightarrow{1_I \otimes \mu} & I \otimes I \\
\downarrow{\mu \otimes 1_I} & & \downarrow{\mu} \\
I \otimes I & \xrightarrow{\mu} & I
\end{array}
\]

Each input object \( I \in \text{Obj}(I) \) is to be thought of as a “set of inputs” for a machine, with \( \mu \) the usual concatenation of input sequences in the case of sequential machines.

2. A category of configuration objects, \( C \). Usually certain of these objects are identified as “state sets”. The category of input objects is a subcategory of the configuration objects, \( I \subseteq C \). Further, the associative bifunctor \( \otimes \) on the category of input objects must extend to an associative bifunctor on the category of configuration objects, \( \otimes: C \times C \to C \). Again, \( (C, \otimes, e) \) is a strict monoidal category and \( (I, \otimes, e) \) is a sub-strict monoidal category of \( (C, \otimes, e) \).

3. For some configuration objects \( C \in \text{Obj}(C) \), the “state sets”, and some semigroups \( (I, \mu) \) in \( I \), a right action with identity exists. A right action with identity for \( C \) and \( (I, \mu) \) is a pair of morphisms in \( C \), \( a: C \otimes I \to C \) and \( e: C \to C \otimes I \), such that

\[
\begin{array}{ccc}
C \otimes I \otimes I & \xrightarrow{1_C \otimes \mu} & C \otimes I \\
\downarrow{a \otimes 1_I} & & \downarrow{a} \\
C \otimes I & \xrightarrow{e} & C
\end{array}
\]

\[
\begin{array}{ccc}
C \otimes I & \xrightarrow{1_C} & C \otimes I \\
\downarrow{e} & & \downarrow{a} \\
C & \xrightarrow{e} & C
\end{array}
\]