AUTOMATA, REPRESENTATIONS, AND PROOFS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

James Michael Worthington

August 2009

AUTOMATA, REPRESENTATIONS, AND PROOFS

James Michael Worthington, Ph.D.

Cornell University 2009

In this dissertation, we study automata, languages, and functions between automata which preserve the language accepted. We examine them from the perspectives of representation theory, category theory, and proof theory. A central theme is that these perspectives interact in many useful, interesting ways.

Let M be a monoid in a monoidal category. We view automata as objects of a category of representations of M, equipped with a start state and an observation function. If M is a monoid in **Set**, this view yields a generalization of the standard notion of a deterministic automaton. In this generalization, the inputs to an automaton are elements of an arbitrary monoid. Omitting the requirement that automata have start states yields categories with final objects. These final objects can be used to minimize deterministic automata.

Let K be a commutative semiring. To express nondeterminism in our framework, we must first take an algebraic detour and show that the category of K-semimodules and K-linear maps is a monoidal category. We then define K-algebras as monoids in this category. We call the corresponding automata K-linear automata. These automata are related to, but distinct from, weighted automata. A K-linear automaton with input K-algebra A defines an element of $\mathrm{Lin}(A, K)$. In certain cases, elements of $\mathrm{Lin}(A, K)$ correspond to K-linear extensions of formal power series.

In the K-linear case, there is an addition defined on both the states and the inputs. Addition of states can be used to represent nondeterminism. Addi-

tion of inputs is needed to define *comultiplication*. Comultiplication defines a K-algebra structure on $\mathrm{Lin}(A, K)$. That is, comultiplication of inputs corresponds to multiplication of "languages". If multiplication and comultiplication satisfy certain "compatibility conditions", the input elements form a structure known as a K-*bialgebra*. Part of this dissertation is the development of the numerous parallels between the theory of bialgebras and the theory of automata and formal languages. These parallels demonstrate that comultiplication is a "hidden parameter" in many standard constructions involving automata and formal languages.

In certain cases, a category of K-linear automata is related to a category of (generalized) deterministic automata by an adjunction. We show that the standard subset construction used to determinize automata can be generalized as a forgetful functor; determinization is essentially forgetting the K-semimodule structure on the states of a K-linear automaton.

Using this adjunction, we can construct a sequence of K-linear automata and morphisms thereof which witnesses the fact that two K-linear automata define the same element of $\mathrm{Lin}(A, K)$. With appropriate restrictions, this witness can be efficiently verified. Furthermore, we can use this witnessing sequence as the basis for a proof system for the equational theory of *Kleene algebra*. We also show that such proofs can be generated by a $PSPACE$ transducer.

Finally, we discuss alternating automata in relation to our framework, and provide a determinizing functor.

BIOGRAPHICAL SKETCH

James was born on June 5$^{\text{th}}$, 1979 and grew up in Chenango Bridge, which is a small suburb of Binghamton, NY. He attended Chenango Valley High School and then earned a Bachelor of Science in Mathematics at SUNY Binghamton (which was later renamed "Binghamton University"). The next stop was Winterthur, Switzerland, to teach English at the Kantonsschule Büelrain. In the following year, he returned to upstate New York and enrolled in Cornell University to pursue a Ph.D in mathematics. After two years, he decided to study computer science as well, for good measure.

For Jess and Beth.

ACKNOWLEDGEMENTS

I would like to thank:

- My advisor, Dexter Kozen, for his support, direction, introducing me to the problem of proving automata equivalent, and the many valuable suggestions he made for improving this dissertation. I would also like to acknowledge his truly excellent textbooks, from which I have learned a great deal.

- My "second advisor", Anil Nerode, for his encouragement, guidance, many informative conversations, and his fruitful suggestion that I look into Hopf Algebras.

- My third committee member, Richard Shore, for many fascinating classes, organizing top-notch logic seminars, and toleration of my non-standard interpretation of "homework due date".

- Maria Terrell, for her advice and encouragement, to me and to all the graduate students in the math department.

- John Hopcroft, for his support and for being a first-class teaching role model.

- John Meier, for running an REU program which led me to study mathematics at the graduate level.

- My undergraduate professors Matt Brin, Benjamin Brewster, Ross Geoghegan, Fernando Guzman, and Luise-Charlotte Kappe, for introducing me to theoretical mathematics.

- Mia Minnes, for her encouragement.

- My family, in particular my sisters Jessica and Bethany, for their support during my graduate studies.

- Krystal, for her love and downright superhuman patience.

- Mauricio and Florencia, for their friendship and hospitality.

- The friends I made while at Cornell, in particular, Becky, Jay, Mike, Paul, Sarah, and Tim.

- Friends from before, in particular, Cory, Matt, Nick, and Scott.

- The anonymous reviewers of RelMiCS/AKA '08 and LFCS '09, who made many valuable suggestions on various precursors of this dissertation.

- The "delivry" staff of Jade Garden restaurant, who in a very real, biochemical sense, made this dissertation possible.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

In this dissertation, we study various types of automata and their morphisms. We examine them from the perspectives of representation theory, category theory, and proof theory. These perspectives are not mutually exclusive; in fact, they interact in many interesting ways.

1.1 Automata as Pointed, Observable Representations

The algebraic perspective can be summarized as follows: an automaton is a pointed, observable representation of an algebraic structure. The elements of this structure are the inputs to the automaton and the elements of the representation are the states of the automaton. "Pointed" refers to the fact that the representation has a distinguished element, which we treat as the start state of the automaton. "Observable" means that there is a function from the representation to some (possibly structured) set, for example, the set {accept, reject}. We are interested in the relationships between automata, languages, and various operations on the set of inputs. Additionally, we study maps between automata which respect the relevant structures.

The simplest algebraic structures we consider are monoids, which act on sets. We define a deterministic automaton as a set S (the state set) along with a distinguished $s \in S$ (the start state), an action of a monoid M on S, and an observation function $S \to O$, where O is an arbitrary set. In the literature, it is common to assume that M is a free, finitely generated monoid.

A deterministic automaton defines a function $M \to O$, which we call the *language accepted* by the automaton. Normally, "language" refers to a set of words. Such a set can be identified with its characteristic function; we generalize and use the term "language" for any function defined by an automaton. Some authors use the word *behavior* in this case. Morphisms of deterministic automata are functions between the underlying state sets which commute with the actions of M, subject to conditions on the start states and observation functions. These conditions guarantee that the domain automaton and the codomain automaton accept the same language.

Transitions in an automaton represent multiplication in a monoid. Otherwise, deterministic automata are somewhat limited algebraically, because the states and observations are just elements of unstructured sets. This leads us to K-linear automata, whose states are elements of a semimodule and whose observations are semiring-valued (see Section 1.4 below for a discussion of the relation between K-linear automata and weighted automata). Many constructions involving K-linear automata rely on these semiring and semimodule structures.

One goal of this dissertation is to examine the similarities between the theory of K-linear automata and the theory of *bialgebras*. To express K-linear automata bialgebraically, we must define the tensor product of semimodules over a commutative semiring. This is the subject of Chapter 4. Using this tensor product, we define K-algebras, K-coalgebras, and K-bialgebras for an arbitrary commutative semiring K. A K-algebra is a monoid in the category of K-semimodules; an example of a K-algebra is the set of polynomials in noncommuting variables $\{x_1, x_2, \cdots, x_n\}$ with coefficients in K. A K-coalgebra C is the categorical dual

of a K-algebra. It has a coassociative operation $\Delta : C \to C \otimes C$, called *comulti-plication*. Comultiplication can be thought of as a rule for splitting elements of C into two parts. A standard result is that the K-coalgebra structure on C defines a K-algebra structure on the dual semimodule of C. Finally, a K-bialgebra is both a K-algebra and a K-coalgebra, satisfying compatibility conditions between the two structures.

We can now give a rough definition of a K-linear automaton as a pointed, observable representation of a K-algebra. States of a K-linear automaton are elements of a K-semimodule. Allowing K-linear combinations of states is one way of expressing nondeterminism; K-linear combinations of inputs are a natural counterpart to this. Multiplication of the inputs corresponds to transition structures of automata. Maps between automata which commute with the transitions correspond to certain maps between representations of a K-algebra. When K is a commutative ring, these maps are known in the literature as *linear intertwiners*.

A K-coalgebra defines a K-algebra on the dual of the underlying K-semimodule; elements of the dual semimodule can be thought of (in certain cases) as K-linear extensions of formal power series. Intuitively, if we know how to split inputs (using Δ), we know how to multiply languages. For example, for an appropriate K-semimodule, there is a comultiplication which corresponds to language intersection, and a comultiplication which corresponds to the shuffle product of languages. Comultiplication requires the inputs to be elements of a K-semimodule; this is another justification for endowing the set of inputs with a K-semimodule structure.

If the K-algebra and K-coalgebra structure on inputs interact nicely, then we have a K-bialgebra. In this case, a construction from the theory of bialge-

bras describes how to run two K-linear automata in parallel. Intuitively, if you can split inputs to K-linear automata, you can multiply K-linear automata: construct a machine which splits the inputs and then feeds the resulting parts to the respective K-linear automata.

While the relation between K-linear automata and K-bialgebras requires the underlying semiring to be commutative, this restriction is not needed for the completeness proof in Chapter 7.

Finally, in Chapter 9, we discuss alternating automata, which are in some sense a combination of deterministic and K-linear automata.

1.2   The Categorical Perspective

We often consider automata as objects in a category. The arrows are maps between automata which preserve the language accepted.

One advantage of this perspective is that it allows succinct expression of similarities between different types of automata. In Chapter 2, we recall the definition of a monoidal category and show how to interpret deterministic automata as objects in the category of representations of a monoid in **Set**. After developing the theory of K-algebras, we can apply these same categorical definitions, since K-algebras are monoids in the monoidal category of K-semimodules.

We also show that the construction to run two automata in parallel is essentially a proof that the representations of a K-bialgebra are themselves objects of a monoidal category. In other words, a K-bialgebra structure defines a multiplication of K-linear automata. Furthermore, the relation between certain cat-

egories of K-linear automata and certain categories of deterministic automata is given by an adjunction. The functor from K-linear automata to deterministic automata is a generalization of the standard determinization procedure via the subset construction.

In addition to automata, we study categories of "observable representations", which are just automata without specified start states. The existence of final objects in these categories yields a minimization procedure for automata.

1.3   Proofs and Complexity

The original motivation for the work in this dissertation was the proof-theoretic properties of automata and their morphisms. Here we use the definition of a proof from proof complexity theory, which defines a proof as an efficiently verifiable witness to the truth of a statement.

With appropriate restrictions on the input K-algebra, we can use an adjunction between the category of K-linear automata and a category of deterministic automata to produce a witness to the equivalence of two equivalent K-linear automata A and B. This witness consists of a sequence of K-linear automata and morphisms of K-linear automata. The sequence starts with A and ends with B. With appropriate restrictions on K and the K-semimodules involved, this witness can be verified in polynomial time. Note that we are using all three of our perspectives: representation-theoretic, categorical, and proof-theoretic.

In particular, we can produce proofs of equivalence of nondeterministic finite automata in this system. We can also augment the system to yield a com-

plete proof system for the equational theory of *Kleene algebra*. This means that the system may be of use for applications involving verification of properties of programs. This is discussed in Chapter 8.

1.4 Related and Motivating Work

There is a well-established tradition of using semirings when studying automata; see, for example, [8], [2], or [20]. There are also many categorical approaches to the theory of automata, see [1] for a classical account, or [26] for a more recent version. The novelty of our approach is a combination of the two ideas, namely, viewing automata as objects in a category of representations of a (categorical) monoid. Our approach is influenced by the theory of weighted automata, but differs in that we view the inputs and the states of an automaton as objects in the same monoidal category. This provides a modern generalization of the concept of an automaton, and introduces a uniform automata-theoretic intuition for certain situations involving both algebraic and coalgebraic components. It is our hope that this intuition may suggest new and worthwhile problems in other areas of mathematics that are only apparent from the viewpoint of automata theory.

We were also influenced by the work of Grossman and Larson, who used representations of a bialgebra to define automata [12], [11]. Their work is also our source for the idea of using an automaton to define an element of a dual vector space; we provide a generalization to commutative semirings, and also consider constructions and morphisms of automata from a bialgebraic perspective. Duchamp et al. used the idea of relating standard operations on formal

languages, such as intersection and shuffle, to various comultiplications [5], [6]. Similar constructions have appeared in combinatorics, e.g., the shuffle product bialgebra [28].

The proof system we use is based on Kozen's proof of the completeness of regular sets for the equational theory of KA, which appeared in [14]. Equivalence proofs utilizing final objects are a major motivation for the use of "universal coalgebra" in theoretical computer science [27]. Universal coalgebra and automata are studied by Rutten in [26], which is our source for the connection between final objects and minimal automata.

Some of the results in this dissertation have already been published in [29] and [30].

1.5   Notes on Terminology

In mathematical literature, the word "algebra" most likely refers to some kind of module enriched with an associative multiplication. The word "coalgebra" most likely refers to a module with operations satisfying the diagrams dual to those defining an algebra.

In computer science literature, these words often appear in the context of "Universal Algebra" and "Universal Coalgebra". In this case, an "algebraic" operation is an operation into the carrier of a structure, and a "coalgebraic" operation is an operation out of the carrier of a structure. For example, "pointing" is an algebraic operation, and "observation" is a coalgebraic operation. Of course, the algebras and coalgebras defined on modules are special cases of uni-

7

versal algebras and coalgebras, respectively. Note that a transition function, when thought of as a function whose domain and codomain is the state set of an automaton, can be thought of either algebraically or coalgebraically.

In this dissertation, we resolve the ambiguity with the use of prefixes. Algebras and coalgebras defined on K-semimodules are called K-algebras/K-coalgebras, where K is a commutative semiring.

The phrase "representation objects" means "class of structures upon which a given class of structures acts". For example, the representation objects of monoids are sets, the representation objects of rings are modules, and the representation objects of fields are vector spaces. Note that a representation is a representation object along with an action.

Finally, by "nondeterministic automaton", we mean the standard notion of a nondeterministic finite automaton as defined in [15]. These are just the nondeterministic automata that one encounters in an introductory course. We are concerned mainly with algebraic encodings (and generalizations) of nondeterminism, in particular, K-linear automata and (to a lesser extent) weighted automata. Nonetheless, we occasionally refer to nondeterministic automata, especially when demonstrating how to encode them algebraically.

BACKGROUND AND DEFINITIONS

This chapter contains definitions and background material necessary for the later chapters. It also serves as an introduction to the main ideas of this dissertation. We define monoids and deterministic automata and express them in the language of category theory. In subsequent chapters, we apply these ideas to other categories.

2.1 Monoids and Deterministic Automata

We give the (non-categorical) definitions of monoids, deterministic automata, and morphisms of deterministic automata. We also define the language accepted by a deterministic automaton.

Definition 2.1.1. A *monoid* is a nonempty set M along with an associative binary operation μ (called multiplication) and a distinguished element $1 \in M$, which is a two-sided identity for μ.

The operation μ is also denoted $\cdot$, or simply by concatenation of inputs. I.e., $\mu(x, y) = x \cdot y = xy$.

Definition 2.1.2. Let M and N be monoids. A function $f : M \to N$ is a *monoid homomorphism* if $f(xy) = f(x)f(y)$ for all $x, y \in M$ and $f(1_M) = 1_N$.

Monoids are ubiquitous in mathematics. For any set S, there is a free monoid on S. Furthermore, given any set S, two monoids can be defined on the set of all endofunctions of S.

Definition 2.1.3. Let S be a set. The *left endomorphism monoid* of S, denoted $\mathrm{End}^l(S)$, is the set of all endofunctions of S with multiplication given by

$$(f \cdot g)(s) = f(g(s))$$

for all $f, g : S \to S$. The multiplicative identity is the identity function on S.

Definition 2.1.4. Let S be a set. The *right endomorphism monoid* of S, denoted $\mathrm{End}^r(S)$, is the set of all endofunctions of S with multiplication given by

$$(f \cdot g)(s) = g(f(s))$$

for all $f, g : S \to S$. The identity function on S is the multiplicative identity.

The distinction between $\mathrm{End}^l(S)$ and $\mathrm{End}^r(S)$ allows us to define automata that read their inputs from left to right and automata that read their inputs from right to left. The monoids $\mathrm{End}^l(S)$ and $\mathrm{End}^r(S)$ are related via the following.

Definition 2.1.5. Let M be a monoid. The *opposite monoid* of M, denoted M^{op}, has the same underlying set and multiplicative identity as M. Multiplication in M^{op} is denoted μ^{op} and defined as

$$\mu^{\mathrm{op}}(x, y) = \mu(y, x).$$

The representation objects of monoids are sets. Monoids can act on sets from the left or from the right.

Definition 2.1.6. Let M be a monoid and S be a set. A *left action* of M on S is a map

$$\triangleright : M \times S \to S$$

written in infix notation, satisfying

$$1 \triangleright s = s$$

10

$$(m \cdot n) \triangleright s = m \triangleright (n \triangleright s)$$

for all $m, n \in M$, $s \in S$.

Equivalently, a left action of M on S is a monoid homomorphism from M to $\mathrm{End}^l(S)$. A *right action* $\triangleleft : S \times M \to S$ is defined analogously and is equivalent to a monoid homomorphism $M \to \mathrm{End}^r(S)$.

A monoid action on a set is also called a *transition structure*. To turn a transition structure into an automaton, we add a start state and an observation function. Let the symbol $\star$ denote a fixed one-element set.

Definition 2.1.7. A *right deterministic automaton $D = (S, M, \alpha, \triangleleft, \Omega, O)$* consists of:

1. A nonempty set S of *states*,

2. An *input monoid M*,

3. A *start function $\alpha : \star \to S$*,

4. A right action $\triangleleft : S \times M \to S$

5. A nonempty set O of *observations* and an *observation function $\Omega : S \to O$*.

Remark 2.1.1. We occasionally refer to "right deterministic M, O-automata" when we wish to emphasize the input monoid and set of observations.

Left deterministic automata are defined similarly. In the sequel, we will give only one side of a definition or theorem, the other follows mutatis mutandis. When it is clear from context, we omit the words "left" and "right".

A deterministic automaton is frequently defined with an input alphabet Σ and a transition function $\delta : S \times \Sigma \to S$. One then defines the extended transition

function $\hat{\delta} : S \times \Sigma^* \to S$. This is a special case of the definition above, since Σ^* is the free monoid on Σ. By freeness of Σ^*, δ can be uniquely extended to a monoid homomorphism

$$\lhd : \Sigma^* \to \mathrm{End}^r(S).$$

Automata define functions from the input monoid to the set of observations. We call such functions *languages*; they are also known as *behaviors*.

Definition 2.1.8. Let $D = (S, M, \alpha, \lhd, \Omega, O)$ be a right deterministic automaton. The *language accepted* by D is the function

$$\rho_D : M \to O$$

$$\rho_D(m) = \Omega(\alpha(\star) \lhd m).$$

Definition 2.1.9. Let $D = (S, M, \alpha_D, \lhd_D, \Omega_D, O)$ and $E = (T, M, \alpha_E, \lhd_E, \Omega_E, O)$ be deterministic automata. The automata D and E are said to be *equivalent* precisely when $\rho_D = \rho_E$.

Of special importance are maps between automata which preserve the language accepted. We require the automata to have the same input monoid and set of observations.

Definition 2.1.10. Let $D = (S, M, \alpha_D, \lhd_D, \Omega_D, O)$ and $E = (T, M, \alpha_E, \lhd_E, \Omega_E, O)$ be deterministic automata. A *morphism of deterministic automata* is a function

$$f : S \to T$$

such that the following diagrams commute:

$$
\begin{array}{ccc}
\star \xrightarrow{\alpha_D} S & S \xrightarrow{\lhd_D} S & S \xrightarrow{\Omega_D} O \\
\alpha_E \searrow \downarrow f & f \downarrow \quad \downarrow f & f \downarrow \nearrow \Omega_E \\
T & T \xrightarrow{\lhd_E} T & T.
\end{array}
$$

Remark 2.1.2. In the literature, functions formally similar to morphisms of deterministic automata have been called *linear sequential morphisms* [1], *relational simulations* [3], *boolean bisimulations* [9], and *disimulations* [29]. Disimulations are based on the *bisimulation lemma* of Kleene algebra [14].

A variant of the following theorem is proved in most of the references mentioned in the above remark.

Theorem 2.1.1. *Let $D = (S, M, \alpha_D, \lhd_D, \Omega_D, O)$ and $E = (T, M, \alpha_E, \lhd_E, \Omega_E, O)$ be deterministic automata. If there is an automaton morphism $f : D \to E$, then D and E are equivalent.*

Proof. For any $m \in M$,

$$\Omega_D(\alpha_D(\star) \lhd_D m) = \Omega_E(f(\alpha_D(\star) \lhd_D m))$$
$$= \Omega_E(f(\alpha_D(\star)) \lhd_E m)$$
$$= \Omega_E(\alpha_E(\star) \lhd_E m)$$

$\square$

2.2 Monoidal Categories

We express monoids, actions, and transition structures categorically, to facilitate generalization in later chapters. We use the standard terminology, which is a bit confusing due to overloading. The main thing to remember is that in a monoidal category C, there is a bifunctor $\otimes : C \times C \to C$. This bifunctor is best thought

of simply as an "associative" binary operation on C with a two-sided identity. It does not have to be a categorical product or coproduct (although it could be).

Monoids can be defined diagrammatically in any monoidal category. However, given a monoid M in a monoidal category C, multiplication in M is not given by the bifunctor $\otimes$. Instead, multiplication in M is given by an arrow $\mu : M \otimes M \to M$. This arrow must satisfy certain diagrams expressing associativity and a multiplicative identity.

The material in this section is from [23].

Definition 2.2.1. A *monoidal category* $B = \langle B, \otimes, e, a, l, r \rangle$ is a category B, a bifunctor $\otimes : B \times B \to B$, an object e of B known as the *unit object*, and three natural isomorphisms a, r, l (*a*ssociator, *l*eft unit, and *r*ight unit). The associator a is a natural isomorphism between the functors

$$\otimes \, (\ \otimes \ ) : B \times B \times B \to B$$

and

$$(\ \otimes \ ) \otimes : B \times B \times B \to B$$

i.e.,

$$a = a_{x,y,z} : x \otimes (y \otimes z) \cong (x \otimes y) \otimes z$$

(note that we identify $(B \times B) \times B$ and $B \times (B \times B)$). Furthermore, the associator must satisfy the following diagram, for all objects w, x, y, z of B:

$$
\begin{array}{ccccc}
w \otimes (x \otimes (y \otimes z)) & \xrightarrow{\;a\;} & (w \otimes x) \otimes (y \otimes z) & \xrightarrow{\;a\;} & ((w \otimes x) \otimes y) \otimes z \\
{\scriptstyle 1 \otimes a}\big\downarrow & & & & \big\uparrow{\scriptstyle a \otimes 1} \\
w \otimes ((x \otimes y) \otimes z) & & \xrightarrow{\qquad\quad a \qquad\quad} & & (w \otimes (x \otimes y)) \otimes z.
\end{array}
$$

This diagram expresses the *pentagonal condition*.

The natural transformations

$$l_x : e \otimes x \cong x$$

and

$$r_x : x \otimes e \cong x$$

must satisfy the following diagrams, which ensure that e acts as an identity for $\otimes$:

$$
\begin{array}{ccc}
x \otimes (e \otimes y) & \xrightarrow{\ \ a\ \ } & (x \otimes e) \otimes y \\
& \searrow^{1 \otimes l} \quad \swarrow_{r \otimes 1} & \\
& x \otimes y &
\end{array}
$$

We also require $l_e = r_e : e \otimes e \to e$.

There are two ways to transform $w \otimes (x \otimes (y \otimes z))$ into $((w \otimes x) \otimes y) \otimes z$ using a; the pentagonal condition ensures their equivalence. The following example of an isomorphism that does not satisfy the pentagonal condition is given in [23]. Let **Ab** be the category of abelian groups and homomorphisms and $\otimes$ be the tensor product over $\mathbb{Z}$. Consider the map

$$a' : X \otimes (Y \otimes Z) \to (X \otimes Y) \otimes Z$$

$$a' : x \otimes (y \otimes z) \to -(x \otimes y) \otimes z.$$

In the pentagonal diagram, one path from $w \otimes (x \otimes (y \otimes z))$ to $((w \otimes x) \otimes y) \otimes z$ contains two instances of the associator and the other contains three, so a difference in sign prevents a' from satisfying the diagram.

The pentagonal condition can be used as the base case in a proof of the *coherence theorem* ([23], VII.2). The coherence theorem allows us to ignore parentheses in a monoidal category and simply write $X \otimes Y \otimes Z$.

If we want $\otimes$ to be commutative, we work with symmetric monoidal categories.

Definition 2.2.2. Let B be a monoidal category. The category B is said to be a *symmetric monoidal category* when it is equipped with a natural family of isomorphisms

$$\sigma_{x,y} : x \otimes y \cong y \otimes x$$

such that

$$\sigma_{x,y} \circ \sigma_{y,x} = 1_x, \quad r_y = l_y \circ \sigma_{y,e} : y \otimes e \cong y$$

and the diagram

$$
\begin{array}{ccccc}
x \otimes (y \otimes z) & \xrightarrow{a} & (x \otimes y) \otimes z & \xrightarrow{\sigma} & z \otimes (x \otimes y) \\
\downarrow{\scriptstyle 1\otimes\sigma} & & & & \downarrow{\scriptstyle a} \\
x \otimes (z \otimes y) & \xrightarrow{a} & (x \otimes z) \otimes y & \xrightarrow{\sigma\otimes1} & (z \otimes x) \otimes y
\end{array}
$$

commutes for all objects x, y, z of B.

There is a coherence theorem for symmetric monoidal categories, which allows us to ignore parentheses. See [23] and the references therein.

We now define monoids in monoidal categories.

Definition 2.2.3. Let $B = \langle B, \otimes, e, a, l, r \rangle$ be a monoidal category. A *monoid* $\langle m, \mu, \eta \rangle$ in B is an object m of B along with an arrow $\mu : m \otimes m \to m$ and an arrow $\eta : e \to m$. The arrows μ and η must satisfy the following diagrams expressing associativity of μ and a multiplicative identity:

$$
\begin{array}{ccccc}
m \otimes (m \otimes m) & \xrightarrow{a} & (m \otimes m) \otimes m & \xrightarrow{\mu\otimes1} & m \otimes m \\
\downarrow{\scriptstyle 1\otimes\mu} & & & & \downarrow{\scriptstyle \mu} \\
m \otimes m & & \xrightarrow{\mu} & & m,
\end{array}
$$

$$e \otimes m \xrightarrow{\eta \otimes 1} m \otimes m \xleftarrow{1 \otimes \eta} m \otimes e$$

with l, μ, r arrows down to $m.$

They are known as the associativity diagrams and unit diagrams, respectively.

Definition 2.2.4. Let $\langle m, \mu, \eta \rangle$ and $\langle m', \mu', \eta' \rangle$ be monoids in a monoidal category B. A *morphism of monoids* is an arrow $f : m \to m'$ of B satisfying the following diagrams:

$$
\begin{array}{ccc}
m \otimes m & \xrightarrow{f \otimes f} & m' \otimes m' \\
\mu \downarrow & & \downarrow \mu' \\
m & \xrightarrow{\quad f \quad} & m'
\end{array}
\qquad
\begin{array}{ccc}
m & \xrightarrow{\quad f \quad} & m' \\
& \eta \nwarrow \quad \nearrow \eta' & \\
& e. &
\end{array}
$$

Example 2.2.1. **Set** is a monoidal category with $\otimes$ interpreted as the cartesian product and $\star$ interpreted as e. Morphisms of monoids are simply monoid homomorphisms.

Example 2.2.2. Another well-known monoidal category is $\langle \mathbf{Ab}, \otimes_{\mathbb{Z}}, \mathbb{Z} \rangle$, where $\otimes_{\mathbb{Z}}$ is the tensor product over the integers. The monoids therein are known as rings.

The coherence theorem for monoidal categories can be used to prove the *General Associative Law* ([23], VII.3.1), which states that any two n-fold multiplications in a monoid are equal. This allows us to write expressions involving μ without parentheses.

Definition 2.2.5. Let B be a monoidal category. The collection of monoids and morphisms of monoids in B forms a category, called the *category of monoids* of B.

Transition structures can also be generalized to monoids in a monoidal category.

Definition 2.2.6. Let B be a monoidal category and $\langle m, \mu, \eta \rangle$ be a monoid in B. A *right action* of $\langle m, \mu, \eta \rangle$ on an object x of B is an arrow $\triangleleft : x \otimes m \to x$ such that the following diagram commutes:

$$
\begin{array}{ccccccc}
(x \otimes m) \otimes m & \xrightarrow{a^{-1}} & x \otimes (m \otimes m) & \xrightarrow{1 \otimes \mu} & x \otimes m & \xleftarrow{1 \otimes \eta} & x \otimes e \\
{\scriptstyle \triangleleft \otimes 1} \downarrow & & & & {\scriptstyle \triangleleft} \downarrow & & \downarrow {\scriptstyle r} \\
x \otimes m & & \xrightarrow{\hspace{3em} \triangleleft \hspace{3em}} & & x & \xleftarrow{1_x} & x
\end{array}
$$

Left actions are defined similarly, using l.

Definition 2.2.7. Let B be a monoidal category, $\langle m, \mu, \eta \rangle$ a monoid in B, and x, x' objects of B. Let $\triangleleft$ and $\triangleleft'$ be right actions of $\langle m, \mu, \eta \rangle$ on x, x', respectively. A *morphism of right actions* is an arrow $f : x \to x'$ in B such that

$$
\begin{array}{ccc}
x \otimes m & \xrightarrow{f \otimes 1} & x' \otimes m \\
{\scriptstyle \triangleleft} \downarrow & & \downarrow {\scriptstyle \triangleleft'} \\
x & \xrightarrow{\hspace{1.5em} f \hspace{1.5em}} & x'.
\end{array}
$$

Remark 2.2.1. For a given monoid m in a monoidal category B, the collection of right actions of m and morphisms of right actions forms a category known as the *category of right representations of m*. In [23], this category is denoted $\mathbf{Ract}_m$. In [28], this category is known as the *category of right m-modules*.

Example 2.2.3. Given a monoid $\langle m, \mu, e \rangle$ in $\langle \mathbf{Ab}, \otimes_{\mathbb{Z}}, \mathbb{Z} \rangle$, m is a ring and $\mathbf{Ract}_m$ is the category of right m-modules. The arrows of $\mathbf{Ract}_m$ are homomorphisms of right m-modules.

We also require the concepts of comonoids and bimonoids in a monoidal category. We postpone their introduction until Chapter 5, since our main examples require the tensor product of semimodules.

2.3 Categories and Deterministic Automata

Given a monoid M in **Set** and a set of observations O, it is easy to see that the collection of right deterministic M, O-automata are the objects of a category C. The arrows of C are morphisms of deterministic automata. Every arrow in C is also an arrow in the category of right representations of M. Note that C cannot have an initial or a final object (except in trivial cases) since the morphisms preserve the language accepted.

To minimize automata, and to show completeness of our proof system, we work with related categories of *observable representations*.

Definition 2.3.1. Let M be a monoid and O a nonempty set. A *right observable representation* consists of a nonempty set S, a right action of M on S, and a function $\Omega : S \to O$.

Remark 2.3.1. Left observable representations are defined similarly. In this section, we only work with right observable representations. We therefore omit the word "right" to avoid excessive wordiness. To emphasize M and O, we sometimes refer to observable M, O-representations.

Definition 2.3.2. Let $D = (S, M, \lhd_D, \Omega_D, O)$ and $E = (T, M, \lhd_E, \Omega_E, O)$ be observable representations. An *morphism of observable representations* is a function $f : S \to T$ such that

$$f(s \lhd_D m) = f(s) \lhd_E m$$

$$\Omega_D(s) = \Omega_E(f(s)).$$

For a given monoid M and nonempty set O, the collection of observable representations and their morphisms forms a category. This category has a final

object. To define this final object, we must exhibit a right action of M on the set of all functions $M \to O$.

Lemma 2.3.1. *Let M be a monoid and O a nonempty set. Left multiplication in M defines a right action of M on the set O^M via*

$$(f \triangleleft m)(n) = f(mn)$$

for all $f \in O^M$, $m, n \in M$.

Proof. We must show

$$((f \triangleleft m) \triangleleft m')(n) = (f \triangleleft mm')(n).$$

This is clear; $((f \triangleleft m) \triangleleft m')(n) = (f \triangleleft m)(m'n) = f(mm'n)$. Since $f \triangleleft 1 = f$, $\triangleleft$ is a right action. $\qquad\square$

Definition 2.3.3. Let M be a monoid and O a nonempty set. Let $F = (S, M, \triangleleft, \Omega, O)$ be the observable representation defined as follows:

$$S = \{f \mid f \in O^M\}$$

$$(f \triangleleft m)(n) = f(mn)$$

$$\Omega(f) = f(1).$$

Given any observable M, O-representation D, there is a morphism $L : D \to F$, which maps every state s of D to the language accepted by s.

Definition 2.3.4. Let $D = (S, M, \triangleleft_D, \Omega_D, O)$ be an observable representation and $s \in S$. The *language accepted* by s is the function $L_s : M \to O$ given by

$$L_s(m) = \Omega(s \triangleleft m)$$

for all $m \in M$.

Lemma 2.3.2. *Let $D = (S, M, \lhd_D, \Omega_D, O)$ be an observable M, O-representation and let $F = (O^M, M, \lhd_F, \Omega_F, O)$ be defined as in Definition 2.3.3. Then there is a unique morphism $l : D \to F$ given by*

$$l(s) = L_s$$

for all $s \in S$.

Proof. We must show that l satisfies $\Omega_D(s) = \Omega_F(l(s))$ and $l(s \lhd_D m) = l(s) \lhd_F m$. The condition on the observation functions is satisfied because

$$\Omega_D(s) = \Omega_D(s \lhd 1) = \Omega_F(l(s)).$$

To see that l commutes with the actions of M, note that $l(s \lhd_D m)$ is the function which, on input m', returns $\Omega_D((s \lhd_D m) \lhd_D m') = \Omega_D(s \lhd_D mm')$. In F, we have that $l(s) \lhd_F m$ is the function which, on input m', returns

$$L_s(mm') = \Omega_D(s \lhd_D mm').$$

We now claim that l is the only morphism $D \to F$. Suppose for the sake of contradiction that $g : D \to F$ is a morphism such that there exists an $s \in S$ with $g(s) \neq L_s$. For notational convenience, let $G_s = g(s)$. Since $G_s \neq L_s$, there is an $m \in M$ such that $G_s(m) \neq L_s(m)$. Since g is a morphism, we must have $\Omega_D(s \lhd m) = \Omega_F(g(s) \lhd_F m) = G_s(m)$. The same argument shows that $\Omega_D(s \lhd m) = L_s(m)$, a contradiction. Therefore l is unique. $\square$

Combining the above, we have the following theorem.

Theorem 2.3.1. *Let M be a monoid, O a nonempty set, and F the observable representation given in Definition 2.3.3. Then F is a final object in the category of observable M, O-representations.*

Example 2.3.1. Let M be the free monoid on generators $\{a, b\}$ and $O = \{0, 1\}$. Then F is the set of all formal languages on $\{a, b\}$ (here we are identifying a language and its characteristic function). The observation function Ω_F returns 1 if the empty word is in the language, and 0 otherwise. For $f \in \{0, 1\}^M$,

$$(f \triangleleft a)(w) = 1 \leftrightarrow f(aw) = 1.$$

In other words, $f \triangleleft a$ is the Brzozowski derivative of f by a.

We now use this final object to minimize automata. We must first eliminate inaccessible states.

Definition 2.3.5. Let $D = (S, M, \alpha, \triangleleft, \Omega, O)$ be a deterministic M, O-automaton. A state $s \in S$ is said to be *accessible* if there is an $m \in M$ such that $\alpha(\star) \triangleleft m = s$. A state s that is not accessible is said to be *inaccessible*.

Definition 2.3.6. Let $D = (S, M, \alpha, \triangleleft, \Omega, O)$ be a deterministic M, O-automaton. Let $S' \subseteq S$ be the set of accessible states of D, and let $D' = (S', M, \alpha, \triangleleft', \Omega', O)$ be the deterministic M, O-automaton where $\triangleleft'$ and Ω' are the restrictions of $\triangleleft$ and Ω to S'. The automaton D is known as the *accessible subautomaton* of D.

Lemma 2.3.3. *Let $D = (S, M, \alpha, \triangleleft, \Omega, O)$ be a deterministic M, O-automaton and let D' be its accessible subautomaton. Then $\rho_D = \rho_{D'}$, i.e., D and D' are equivalent.*

Proof. The inclusion map $\iota : S' \to S$ is a deterministic automaton morphism. $\square$

Theorem 2.3.2. *Let $D = (S, M, \alpha, \triangleleft, \Omega, O)$ be a deterministic M, O-automaton, all of whose states are accessible. Let M be the deterministic M, O-automaton constructed according to the following procedure:*

- *Forget the start state of D, yielding an observable representation D'.*

- *Let M' be the image of D' in F under the unique morphism g.*

- *Let M be the deterministic M,O-automaton equivalent to D obtained by declaring the image of the start state of D to be the start state of M'. Note that g is a deterministic automaton morphism $D \to M$.*

Then M is a minimal in the following sense: for any equivalent deterministic automaton E, the number of states of E is greater than or equal to the number of states of M. This is true even if the automata involved have infinitely many states.

Proof. There is a surjective map from the (accessible) states of E to the states of M. Furthermore, viewing M as an observable representation, no two states of M accept the same language. $\qquad\square$

2.4 Proofs and Complexity

We now consider the proof-theoretic applications of deterministic automata and their morphisms. A sequence

$$D_1, f_1, D_2, \ldots, D_n, f_n, D_{n+1}$$

of deterministic automata D_i and morphisms of deterministic automata $f_i : D_i \to D_{i+1}$ is readily seen to be a witness to the equivalence of D_1 and D_{n+1}, since the f_i's preserve the language accepted. Moreover, given any two equivalent deterministic automata, we can find a sequence witnessing their equivalence.

Theorem 2.4.1. *Let D and E be equivalent deterministic M,O-automata. Let D' and E' be their respective accessible subautomata, and let M be the minimization of D'.*

Then the sequence

$$D \xleftarrow{\iota_1} D' \xrightarrow{f} M \xleftarrow{g} E' \xrightarrow{\iota_2} E$$

witnesses the equivalence of D and E. Here ι_1 is the inclusion of D' into D, ι_2 is the inclusion of E' into E, f is the unique morphism from D' to M, and g is the unique morphism from E' to M.

Proof. This is a consequence of Lemma 2.3.3 and Theorem 2.3.2. The automata D' and E' map to the same M because the map to the final observable representation sends a state to the language it accepts. □

If we restrict the complexity of the automata and morphisms, we can get a proof system, similar to the proof systems defined in [4]. Here we assume a binary encoding of the theory in question.

Definition 2.4.1. Let $L \subseteq \{0,1\}^*$. A *proof system* for L is a binary relation $P(x,y)$ satisfying the following three conditions:

- Completeness: $x \in L \rightarrow \exists y\, P(x,y)$.

- Soundness: $\exists y\, P(x,y) \rightarrow x \in L$.

- $P(x,y)$ is an efficiently verifiable relation.

Example 2.4.1. Let M be the free monoid on generators $\{a,b\}$ and $O = \{0,1\}$. Using any reasonable binary encoding of automata and morphisms, the witnessing sequences for equivalent deterministic *finite state* automata can be verified efficiently.

SEMIRINGS, SEMIMODULES, AND WEIGHTED AUTOMATA

In order to introduce other types of automata, we must define semirings and semimodules and establish some of their basic properties. Semirings and semimodules underlie weighted automata, whose observation functions are semiring-valued and whose states are elements of finitely generated free semimodules. Furthermore, weighted automata accept formal power series, which have coefficients from a semiring. See [2] and [20].

In later chapters, we endow the set of inputs to an automaton with a semimodule structure. This allows us to work with K-linear automata, which are a representation-theoretic generalization of deterministic automata.

3.1 Basic Definitions

The material in this section is from [10].

Definition 3.1.1. A *semiring* is a structure $(K, +, \cdot, 0, 1)$ such that $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, and the following laws hold:

$$j(k + l) = jk + jl$$

$$(k + l)j = kj + lj$$

$$0k = k0 = 0$$

for all $j, k, l \in K$. If $(K, \cdot, 1)$ is a commutative monoid, then K is said to be a *commutative semiring*. If $(K, +, 0)$ is an idempotent monoid, then K is said to be an *idempotent semiring*.

The representation objects of semirings are known as *semimodules*.

Definition 3.1.2. Let K be a semiring. A *left K-semimodule* is a commutative monoid $(M, +, 0)$ along with a left action of K on M. The action satisfies the following axioms:

$$(j + k)m = jm + km$$

$$j(m + n) = jm + jn$$

$$(jk)m = j(km)$$

$$1m = m$$

$$k0_M = 0_M = 0_K m$$

for all $j, k \in K$ and $m, n \in M$. If addition in M is idempotent, M is said to be an *idempotent left K-semimodule*.

Right K-semimodules are defined analogously. In the sequel, we give only "one side" of a definition. If K is commutative, then every left K-semimodule can be regarded as a right K-semimodule, and vice versa. In this case, we omit the words "left" and "right".

Example 3.1.1. Let K be a semiring and m, n be positive integers. The set of $m \times n$ matrices over K is a left K-semimodule, and the set of $m \times m$ matrices over K is a semiring, using the standard definitions of matrix addition, multiplication, and left scalar multiplication.

Semimodules can be combined using the operations of direct sum and direct product.

Definition 3.1.3. Let K be a semiring and $\{M_i \mid i \in I\}$ be a collection of left K-semimodules for some index set I. Let M be the cartesian product of the

underlying sets of the M_i's. The *direct product* of the M_i's, denoted $\prod M_i$, is the set M endowed with pointwise addition and scalar multiplication. The *direct sum* of the M_i's, denoted $\bigoplus M_i$, is the subsemimodule of $\prod M_i$ in which all but finitely many of the coordinates are 0.

Remark 3.1.1. As usual, direct products and direct sums coincide when I is finite.

Standard definitions from the theory of modules can be generalized to the theory of semimodules.

Definition 3.1.4. Let K be a semiring and M, N be left K-semimodules. A function $\phi : M \to N$ is a *left K-semimodule homomorphism* if

$$\phi(m + m') = \phi(m) + \phi(m') \text{ for all } m, m' \in M$$

$$\phi(km) = k\phi(m) \text{ for all } m \in M, k \in K.$$

Such ϕ are also called K-*linear maps*.

Definition 3.1.5. Let K be a semiring and M a left K-semimodule. Let $\{M_i \mid i \in I\}$ be an indexed family of subsemimodules of M. For each i, we have an injection

$$\iota_i : M_i \to M$$

and so there is an induced K-linear map

$$\iota^* : \bigoplus M_i \to M$$

$$\iota^*((x_i)) = \sum_{i \in I} x_i.$$

This sum exists, since only finitely many of the entries in (x_i) are non-zero. If ι^* is an isomorphism, we say that the family $\{M_i \mid i \in I\}$ is a *direct sum decomposition* of M and write $M = \bigoplus_i M_i$.

Definition 3.1.6. Let K be a semiring, M a left K-semimodule, and $\equiv$ an equivalence relation on M. Then $\equiv$ is a *congruence relation* if and only if

$$m \equiv m' \text{ and } n \equiv n' \text{ implies } m + n \equiv m' + n'$$

$$m \equiv m' \text{ implies } km \equiv km'$$

for all $k \in K$, $m, m', n, n' \in M$.

Definition 3.1.7. Let K be a semiring, M a left K-semimodule, and $\equiv$ a congruence relation on M. For each $m \in M$, let $[m]$ be the equivalence class of m with respect to $\equiv$. Let $M/\equiv$ be the set of all such equivalence classes. Then $M/\equiv$ is a left K-semimodule with the following operations:

$$[m] + [n] = [m + n]$$

$$k[m] = [km]$$

for all $m, n \in M, k \in K$. This semimodule is known as the *factor semimodule* of M by $\equiv$. It is also sometimes called a *quotient semimodule*.

In the sequel, we use elementary facts about factor semimodules, free semimodules, congruence relations, and homomorphisms without comment. See [10] for proofs.

The collection of left K-semimodules and K-linear maps form a category, which we denote by K-**Mod**.

3.2 Free K-semimodules and Weighted Automata

Unlike vector spaces, not all K-semimodules are free K-semimodules (neither are all modules over a ring free modules). However, free K-semimodules exist,

and have all the nice properties that freeness entails.

Definition 3.2.1. Let K be a semiring and X a nonempty set. The *free left K-semimodule on X* is the set of all finite formal sums of the form

$$k_1 x_1 + k_2 x_2 + \cdots + k_n x_n$$

with $k_i \in K$ and $x_i \in X$. Alternatively, it is the set of all $f \in K^X$ with finite support. Addition and the action of K are defined pointwise.

Equivalently, one can define a left K-semimodule M to be free if and only if M has a basis [10].

Definition 3.2.2. Let M be a left K-semimodule and X a nonempty subset of M. Then there is a unique K-linear map ϕ from the free left K-semimodule on X to M given by

$$\phi(f) = \sum_{x \in X} f(x) x.$$

If ϕ is surjective, then X is said to be a *set of generators* of M. If ϕ is injective, then X is said to be *linearly independent*. If ϕ is a bijection, then X is said to be a *basis* of M.

Remark 3.2.1. Using Definition 3.1.5, it is easy to see that the free left K-semimodule on a nonempty set X is isomorphic to $\bigoplus_{x \in X} K$.

Remark 3.2.2. If M is a left K-semimodule with a basis of size m, and N is a left K-semimodule with a basis of size n, when m, n are positive integers, then a K-linear map from M to N can be represented by an $n \times m$ matrix over K.

The states of a weighted automaton are elements of a free K-semimodule on a finite set.

Definition 3.2.3. Let S be a finite nonempty set with n elements, Σ a finite alphabet, and K a semiring. A *right K-weighted automaton* $W = (K^S, s, T_x, \Sigma, t)$ consists of the following:

1. A $1 \times n$ *start vector* $s \in K^S$,

2. An $n \times n$ *transition matrix* over K for each $x \in \Sigma$, denoted T_x,

3. An $n \times 1$ *observation vector* t over K.

Remark 3.2.3. This is essentially the definition of a *linear representation* given in [2].

Definition 3.2.4. Let $W = (K^S, s, T_x, \Sigma, t)$ be a right K-weighted automaton. The *language accepted* by W is the formal power series $\rho_W \in K^{\Sigma^*}$ given by

$$\rho_W(w) = s \cdot T_{x_1} \cdot T_{x_2} \cdots T_{x_n} \cdot t$$

for each $w = x_1 x_2 \cdots x_n \in \Sigma^*$.

Remark 3.2.4. Cf. *recognizable formal series* in [2].

Remark 3.2.5. Note that the transition matrices act on the start vector by multiplication on the right. In other words, these automata read their inputs from left to right. Left K-weighted automata are defined similarly.

This definition requires many assumptions: the heart of the definition is the action of a free monoid on a free K-semimodule on a finite set. This is a "low-level" generalization of deterministic automata. A "high-level" generalization, given in Chapter 6, uses the fact that K-semimodules are the representation objects of K-algebras. However, the definition of a weighted automaton does allow for the semiring K to be noncommutative, whereas the definition of a K-algebra requires K be be commutative.

3.3 Dual Semimodules

Dual semimodules allow us to express "observation/state space duality" for weighted automata. In this section, we require K to be a commutative semiring.

Definition 3.3.1. Let K be a commutative semiring and M a K-semimodule. The set of all K-linear maps $M \to K$ is denoted $\mathrm{Lin}(M, K)$.

The following two lemmas are simple generalizations of standard facts about dual R-modules, where R is a commutative ring.

Lemma 3.3.1. *Let K be a commutative semiring and M a K-semimodule. The set $\mathrm{Lin}(M, K)$ can be endowed with a K-semimodule structure.*

Proof. The set $\mathrm{Lin}(M, K)$ is a commutative monoid under pointwise addition. Let $f \in \mathrm{Lin}(M, K)$. The action of K on $\mathrm{Lin}(M, K)$, denoted $\triangleright$, is defined by $k \triangleright f(m) = kf(m)$. Commutativity of K is needed to show that the resulting functions are K-linear. $\square$

Lemma 3.3.2. *Let K be a commutative semiring, X a finite nonempty set, and F the free K-semimodule on X. Then $\mathrm{Lin}(F, K)$ is also a free K-semimodule on a set of size $|X|$.*

Proof. Let $x_1, x_2, \ldots, x_n$ be a basis of F. Let $f_i \in \mathrm{Lin}(F, K)$ be such that

$$f_i(x_j) = 1 \text{ if } i = j, \text{ and } 0 \text{ otherwise.}$$

We claim that the f_i's are a basis of $\mathrm{Lin}(F, K)$. Let $g \in \mathrm{Lin}(F, K)$. We must express g as a K-linear combination of the $f_i's$. Set $a_i = g(x_i)$. Given an arbitrary

$$k_1 x_1 + k_2 x_2 + \cdots k_n x_n \in F,$$

$$g(k_1 x_1 + k_2 x_2 + \cdots + k_n x_n)$$
$$= g(k_1 x_1) + g(k_2 x_2) + \cdots + g(k_n x_n)$$
$$= k_1 g(x_1) + k_2 g(x_2) + \cdots + k_n g(x_n)$$
$$= k_1 a_1 + k_2 a_2 + \cdots + k_n a_n$$
$$= a_1 f_1(k_1 x_1 + \cdots + k_n x_n) + \cdots + a_n f_n(k_1 x_1 + \cdots + k_n x_n)$$

and so $g = a_1 f_1 + a_2 f_2 + \cdots + a_n f_n$. Hence the $f_i's$ generate $\mathrm{Lin}(F, K)$. Moreover, the f_i's are linearly independent; if

$$j_1 f_1 + j_2 f_2 + \cdots + j_n f_n = j_1' f_1 + j_2' f_2 + \cdots + j_n' f_n,$$

then evaluating each side on x_i yields $j_i = j_i'$. $\qquad\square$

Remark 3.3.1. The f_i's in the proof of Lemma 3.3.2 are known as the *dual basis* of the basis $\{x_1, x_2, \ldots, x_n\}$. The dual basis is also denoted $\{x_1^*, x_2^*, \ldots, x_n^*\}$.

We note that reversal of weighted automata can be expressed using dual modules. We require the following lemma.

Lemma 3.3.3. *Let A, B be matrices over a commutative semiring K, where the sizes of A and B are such that AB is defined. Then*

$$(AB)^{\mathrm{T}} = B^{\mathrm{T}} A^{\mathrm{T}}.$$

Proof. The standard proof from linear algebra is valid in this case. $\qquad\square$

Theorem 3.3.1. *Let $A = (S, s, T_x, \Sigma, t)$ be a weighted automaton. Then the weighted automaton $B = (\mathrm{Lin}(M, K), t^{\mathrm{T}}, T_x^{\mathrm{T}}, \Sigma, s^{\mathrm{T}})$ satisfies*

$$\rho_A(w) = \rho_B(w^{\mathrm{R}})$$

for all $w \in \Sigma^*$, where w^{R} is the reverse of a word w.

Proof. To prove the claim, let $w = x_1 x_2 \cdots x_n$ with $x_i \in \Sigma$. For some $k \in K, \rho_A(w) = k$. By definition,

$$\rho_A(w) = s \cdot T_{x_1} T_{x_2} \cdots T_{x_n} \cdot t^{\mathrm{T}} = k.$$

Taking the transpose of both sides of this equation yields $\rho_B(w^{\mathrm{R}}) = k^{\mathrm{T}} = k$. $\square$

This result is basically an application of the following two facts.

Lemma 3.3.4. *Let M be a monoid and S a set. A left action of M on S is equivalent to a right action of M^{op} on S, and vice versa.*

Proof. Straightforward. $\square$

Lemma 3.3.5. [7] *Let V, W be finite dimensional vector spaces over a field F with bases $\{v_1, v_2, \ldots, v_n\}$ and $\{w_1, w_2, \ldots, w_m\}$, respectively. Fix $\phi \in \mathrm{Lin}(V, W)$. For each $f \in W^*$, $f \circ \phi$ is a linear transformation $V \to F$. Denote this transformation ϕ^*. We have*

$$\phi^* : W^* \to V^*$$

$$\phi^*(f) = f \circ \phi$$

is a linear transformation $W^* \to V^*$. The transpose of the matrix for ϕ with respect to the bases $\{v_1, v_2, \ldots, v_n\}$ and $\{w_1, w_2, \ldots, w_m\}$ is the matrix for ϕ^* with respect to the dual bases of V and W.

Proof. The proof in [7] is valid even if "F is a field" is weakened to "F is a commutative semiring". $\square$

In general, actions "change sides" when dualizing (cf. Lemma 2.3.1). The transitions of a right K-weighted automaton define a left action of Σ^* on the dual of the underlying K-semimodule. This left action is equivalent to a right action of Σ^{op}. The transpose represents the right action of Σ^{op}, hence input words get reversed.

TENSOR PRODUCTS OVER COMMUTATIVE SEMIRINGS

We would like to define monoids in K−**Mod**. To do this, we must have a bifunctor which makes K−**Mod** into a monoidal category. The tensor product is this bifunctor.

The tensor product of a finite sequence of K-semimodules is a universal object representing multilinear maps out of the K-semimodules. Multilinear maps play a crucial role in the theory of K-linear automata, as we will see in Chapters 5 and 6. Briefly, concatenation of words corresponds to a map from a tensor product, and certain ways of combining automata involve tensor products of start, accept, and transition functions. Furthermore, the tensor product allows us to define comultiplication.

Unfortunately, the literature contains multiple inequivalent definitions of the tensor product of K-semimodules: the tensor product as defined in [10] is not the same as the tensor product defined in [22] or [13]. In fact, the tensor product defined in [10] is the trivial K-semimodule when applied to idempotent K-semimodules. We discuss alternative constructions in Section 4.3.

4.1 Constructing the Tensor Product

We assume that K is a commutative semiring and mimic the construction of the tensor product of modules over a commutative ring in [21]. This is essentially the construction used in [22] and [13]. The point is to work in the appropriate category and construct an object with the appropriate universal property.

We first recall the definition of a multilinear map.

Definition 4.1.1. Let $M_1, M_2, \ldots, M_n, F$ be K-semimodules. A map

$$f : M_1 \times M_2 \times \cdots \times M_n \to F$$

is said to be K-*multilinear* if it is K-linear in each variable. That is, for a choice of elements $m_1, m_2, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n$ the map

$$m \mapsto f(m_1, m_2, \ldots, m_{i-1}, m, m_{i+1}, \ldots, m_n)$$

is a K-linear map $M_i \to F$. Such a map is also called an n-*multilinear map*, or a K-*bilinear* map when $n = 2$.

The tensor product over a commutative ring R is defined by the following universal property. Let $M_1, M_2, ..., M_n$ be R-modules. Let C be the category whose objects are n-multilinear maps

$$f : M_1 \times M_2 \times \cdots \times M_n \to F$$

where F ranges over R-modules. To define the arrows of C, let

$$f : M_1 \times M_2 \times \cdots \times M_n \to F \text{ and } g : M_1 \times M_2 \times \cdots \times M_n \to G$$

be objects of C. An arrow $f \to g$ in C is an R-linear map $h : F \to G$ such that $h \circ f = g$. A *tensor product* of $M_1, M_2, ..., M_n$, denoted $M_1 \otimes_R M_2 \otimes_R \cdots \otimes_R M_n$, is an initial object in C. The tensor product exists and is unique up to isomorphism.

We now construct the tensor product of semimodules over a commutative semiring. Let K be a commutative semiring and $M_1, M_2, ..., M_n$ be K-semimodules. Let T be the free K-semimodule on the (underlying) set $M_1 \times M_2 \times \cdots \times M_n$. Let $\equiv$ be the congruence relation on T generated by the equivalences

$$(m_1, ..., m_i +_{M_i} m_i', ..., m_n) \equiv (m_1, ..., m_i, ..., m_n) +_T (m_1, ..., m_i', ..., m_n)$$

$$(m_1, ..., km_i, ..., m_n) \equiv k(m_1, ..., m_i, ..., m_n)$$

for all $k \in K, m_i, m_i' \in M_i, 1 \le i \le n$.

Let $\iota : M_1 \times M_2 \times \cdots \times M_n \to T$ be the canonical injection of $M_1 \times M_2 \times \cdots \times M_n$ into T. Let ϕ be the composition of ι and the quotient map $q : T \to T/\equiv$.

Theorem 4.1.1. *The map ϕ is multilinear and is a tensor product of* $M_1, M_2, ..., M_n$.

Proof. Multilinearity of ϕ is obvious from its definition. Let G be a K-semimodule and

$$g : M_1 \times M_2 \times \cdots \times M_n \to G$$

be a K-multilinear map. By freeness of T, there is a unique K-linear map $\gamma : T \to G$ such that the following diagram commutes:

The kernel of γ, denoted $\equiv_\gamma$, is a congruence relation on T given by

$$t \equiv_\gamma t' \text{ if and only if } \gamma(t) = \gamma(t')$$

for all $t, t' \in T$. Since g is K-multilinear, $t \equiv t'$ implies $t \equiv_\gamma t'$, where $\equiv$ is the congruence relation used in the definition of the tensor product. Therefore γ can be factored through $T/\equiv$, and there is a K-linear map

$$g_* : T/\equiv \to G$$

making the following diagram commute:

$$
\begin{array}{ccc}
 & & T/\!\equiv \\
 & \overset{\phi}{\nearrow} & \Big\downarrow g_* \\
M_1 \times M_2 \times \cdots \times M_n & & \\
 & \underset{g}{\searrow} & \\
 & & G.
\end{array}
$$

The image of ϕ generates $T/\!\equiv$, so g_* is uniquely determined. □

We denote the module $T/\equiv$ by $M_1 \otimes_K M_2 \otimes_K \cdots \otimes_K M_n$. When it is clear from context, we omit the subscript on the $\otimes$ symbol. For $(x_i) \in \prod M_i$, we denote $\phi(x_1, x_2, ..., x_n)$ by $x_1 \otimes x_2 \otimes \cdots \otimes x_n$. Note that we can write any element of $M \otimes N$ as a sum of terms of the form $m \otimes n$, because $k(m \otimes n) = km \otimes n$ and terms of the form $m \otimes n$ generate $M \otimes N$.

One can do away with this construction and show that the tensor product exists using purely categorical methods: see Chapter V, Section 8 of [23]. Nonetheless, this more concrete construction is useful for our purposes, since we frequently reason about specific elements of tensor products in the sequel.

4.2 Properties of the Tensor Product

Tensor products enjoy many useful properties. This section is devoted to the proofs of Theorems 4.2.1 and 4.2.2. These theorems establish that K−**Mod** is a symmetric monoidal category. They also describe the structure of tensor products of free K-semimodules. The proofs rely on the initiality of the tensor product and are straightforward generalizations of the proofs for tensor products over a commutative ring in [21]. We provide them for completeness.

Theorem 4.2.1. *Let K be a commutative semiring, n a positive integer, and $M_1, N_1, M_2, N_2, \ldots, M_n, N_n$ be K-semimodules. Then:*

1. *There is a unique isomorphism*

$$M_1 \otimes (M_2 \otimes M_3) \to (M_1 \otimes M_2) \otimes M_3$$

   such that

$$m_1 \otimes (m_2 \otimes m_3) \mapsto (m_1 \otimes m_2) \otimes m_3$$

   for all $m_i \in M_i$, $1 \leq i \leq 3$.

2. *There is a unique isomorphism*

$$M_1 \otimes M_2 \to M_2 \otimes M_1$$

   such that

$$m_1 \otimes m_2 \mapsto m_2 \otimes m_1$$

   for all $m_i \in M_i$, $1 \leq i \leq 2$.

3. *There is a unique isomorphism*

$$M_1 \otimes K \cong M_1$$

   such that

$$m \otimes k \mapsto km$$

   for all $k \in K, m \in M_1$.

4. *Let M_i, N_i be K-semimodules, $1 \leq i \leq n$, and let $f_i : M_i \to N_i$ be a collection of K-linear maps. Then there is a unique K-linear map*

$$M_1 \otimes M_2 \otimes \cdots \otimes M_n \to N_1 \otimes N_2 \otimes \cdots \otimes N_n$$

satisfying

$$m_1 \otimes m_2 \otimes \cdots \otimes m_n \mapsto f_1(m_1) \otimes f_2(m_2) \otimes \cdots \otimes f_n(m_n)$$

for all $m_i \in M_i$.

5. $M \otimes \left(\bigoplus_{i \in I} N_i\right) \cong \bigoplus_{i \in I} (M \otimes N_i)$ *for any index set I.*

6. *Let M,N be free K-semimodules with bases $\{m_i\}_{i \in I}$ and $\{n_j\}_{j \in J}$, respectively. Then $M \otimes N$ is a free K-semimodule with basis $\{m_i \otimes n_j\}_{i \in I, j \in J}$.*

Proof of (1). For $m_1 \in M_1$, let f_{m_1} be the map

$$f_{m_1} : M_2 \times M_3 \to (M_1 \otimes M_2) \otimes M_3$$

$$(m_2, m_3) \mapsto (m_1 \otimes m_2) \otimes m_3.$$

The map f_{m_1} is K-bilinear. Therefore f_{m_1} defines a K-linear map

$$f_{m_1}^* : M_2 \otimes M_3 \to (M_1 \otimes M_2) \otimes M_3$$

$$m_2 \otimes m_3 \mapsto (m_1 \otimes m_2) \otimes m_3.$$

Now consider the K-bilinear map

$$M_1 \times (M_2 \otimes M_3) \to (M_1 \otimes M_2) \otimes M_3$$

$$(m_1, m_2 \otimes m_3) \mapsto f_{m_1}^*(m_2 \otimes m_3).$$

This defines a K-linear map $M_1 \otimes (M_2 \otimes M_3) \to (M_1 \otimes M_2) \otimes M_3$ with the specified action on the generators, which guarantees uniqueness. A similar construction yields the inverse map $(M_1 \otimes M_2) \otimes M_3 \to M_1 \otimes (M_2 \otimes M_3)$. $\qquad \square$

Proof of (2). It is easy to see that the map

$$M_1 \times M_2 \to M_2 \otimes M_1$$

$$(m_1, m_2) \mapsto m_2 \otimes m_1$$

is K-bilinear. Therefore it factors through the tensor product and maps $m_1 \otimes m_2$ to $m_2 \otimes m_1$. By symmetry, there is an inverse $M_2 \otimes M_1 \to M_1 \otimes M_2$. The map is unique because elements of the form $m \otimes n$ generate the tensor product. $\quad\square$

Proof of (3). We prove a slightly more general statement. Let M be a K-semimodule and N a free K-semimodule with basis $\{n\}$. We claim that $M \otimes N$ and M are isomorphic. The map

$$f : M \times N \to M$$

$$(m, kn) \mapsto km$$

is a K-bilinear map from $M \times N$ to M, and hence induces a K-linear map

$$f' : M \otimes N \to M$$

$$m \otimes kn \mapsto km.$$

There is also a K-linear map

$$g : M \to M \otimes N$$

$$m \mapsto m \otimes n.$$

A simple calculation shows that f' and g are inverse to each other. The maps are unique since we have specified their actions on generating sets. This implies 4.2.1.3, since K can be considered as a free K-semimodule over itself with basis $\{1\}$. $\quad\square$

Proof of (4). The f_i's define a function on the product

$$\prod_i f_i : \prod_i M_i \to \prod_i N_i.$$

The function $\prod f_i$ is not K-multilinear, but it is easy to check that the composition of $\prod f_i$ with the canonical K-multilinear map

$$\phi : N_1 \times N_2 \times \cdots \times N_n \to N_1 \otimes N_2 \otimes \cdots \otimes N_n$$

is a K-multilinear map

$$M_1 \times M_2 \times \cdots \times M_n \to N_1 \otimes N_2 \otimes \cdots \otimes N_n.$$

This maps sends $(m_1, m_2, \ldots, m_n)$ to $f_1(m_1) \otimes f_2(m_2) \otimes \cdots \otimes f_n(m_n)$. By the initiality of the tensor product of the M_i's, there is a unique K-linear map

$$M_1 \otimes M_2 \otimes \cdots \otimes M_n \to N_1 \otimes N_2 \otimes \cdots \otimes N_n$$

satisfying

$$m_1 \otimes m_2 \otimes \cdots \otimes m_n \mapsto f_1(m_1) \otimes f_2(m_2) \otimes \cdots \otimes f_n(m_n).$$

This induced map is denoted $T(f_1, f_2, \ldots, f_n)$ or $f_1 \otimes f_2 \otimes \cdots \otimes f_n$. Note that $f_1 \otimes f_2 \otimes \cdots \otimes f_n$ is an abuse of notation; $f_1 \otimes f_2 \otimes \cdots \otimes f_n$ could also denote an element of $\mathrm{Lin}(M_1, N_1) \otimes \mathrm{Lin}(M_2, N_2) \otimes \cdots \otimes \mathrm{Lin}(M_n, N_n)$, since $\mathrm{Lin}(M_i, N_i)$ is a K-semimodule by Lemma 3.3.1. Nonetheless, this notation is common in the literature, and we use it occasionally. The meaning will be clear from the context. $\qquad\square$

To prove (5), we must establish a few properties of T.

Lemma 4.2.1. *The map*

$$T : \prod_{i=1}^{n} \mathrm{Lin}(M_i, N_i) \to \mathrm{Lin}\left(\bigotimes_{i=1}^{n} M_i, \bigotimes_{i=1}^{n} N_i\right)$$

is K-multilinear.

Proof. Fix an index i such that $1 \leq i \leq n$, and fix $f_j \in \text{Lin}(M_j, N_j)$ for all $1 \leq j \leq n, j \neq i$. Let $g, h \in \text{Lin}(M_i, N_i)$. We must show that the map

$$\psi : \text{Lin}(M_i, N_i) \rightarrow \text{Lin}(\bigotimes_{i=1}^{n} M_i, \bigotimes_{i=1}^{n} N_i)$$

$$\psi(g) \mapsto T(f_1, \ldots, f_{i-1}, g, f_{i+1}, \ldots f_n)$$

is a K-linear map. By definition,

$$\psi(g + h) =$$

$$T(f_1, \ldots, f_{i-1}, (g+h), f_{i+1}, \ldots, f_n) =$$

$$\lambda(x_1 \otimes \cdots \otimes x_n).f_1(x_1) \otimes \cdots \otimes (g+h)(x_i) \otimes \cdots \otimes f_n(x_n) =$$

$$\lambda(x_1 \otimes \cdots \otimes x_n).f_1(x_1) \otimes \cdots \otimes g(x_i) \otimes \cdots \otimes f_n(x_n) + f_1(x_1) \otimes \cdots \otimes h(x_i) \otimes \cdots \otimes f_n(x_n)$$

$$= \psi(g) + \psi(h).$$

A similar argument shows $\psi(kg) = k\psi(g)$. Since i was arbitrary, T is K-multilinear. $\square$

Lemma 4.2.2 establishes the functoriality of T.

Lemma 4.2.2. *Let n be a positive integer and M_i, N_i, O_i be K-semimodules for $1 \leq i \leq n$. Let $f_i : M_i \rightarrow N_i$ and $g_i : N_i \rightarrow O_i$. Then*

$$T(f_1 \circ g_1, f_2 \circ g_2, \ldots, f_n \circ g_n) = T(f_1, f_2, \ldots, f_n) \circ T(g_1, g_2, \ldots, g_n).$$

Furthermore,

$$T(1_{M_1}, 1_{M_2}, \ldots, 1_{M_n}) = \text{id},$$

where id *is the identity function on $M_1 \otimes M_2 \otimes \cdots \otimes M_n$.*

Proof. Straightforward calculation. $\square$

Example 4.2.1. For a fixed K-semimodule M, let τ_M be the endofunctor on K$-$**Mod** such that

$$\tau_M(N) = M \otimes N$$

$$\tau_M(f) = T(1_M, f)$$

for all K-semimodules N, N' and K-linear maps $f : N \to N'$. Lemma 4.2.2 ensures that τ is a functor. Lemma 4.2.1 ensures that τ's action on arrows of K$-$**Mod** is a K-linear map from $\mathrm{Lin}(N, N')$ to $\mathrm{Lin}(M \otimes N, M \otimes N')$.

The proof of (5) also requires the following criterion for a K-semimodule to be a direct product of finitely many K-subsemimodules.

Lemma 4.2.3. *Let M be a K-semimodule and $n \geq 1$. Let*

$$\phi_i : M \to M \qquad 1 \leq i \leq n$$

be a collection of K-linear maps satisfying

$$\phi_i \circ \phi_j = 0 \quad \text{if } i \neq j, \qquad \sum_{i=1}^{n} \phi_i = 1_M.$$

Let $M_i = \phi_i(M)$, and let

$$\phi : M \to \prod M_i$$

$$\phi(m) = (\phi_1(m), \phi_2(m), \ldots, \phi_n(m)).$$

Then ϕ is a K-linear isomorphism $M \to \prod M_i$.

Proof. We first note that $\phi_i^2 = \phi_i$ for $1 \leq i \leq n$:

$$\phi_i = \phi_i \circ 1_M = \phi_i \circ \sum_{j=1}^{n} \phi_j = \phi_i \circ \phi_i.$$

The map ϕ is injective: suppose $\phi(m) = \phi(n)$ for some $m, n \in M$. By definition, $\phi_i(m) = \phi_i(n)$ for $1 \leq i \leq n$. This implies $\sum \phi_i(m) = \sum \phi_i(n)$, hence $1_M(m) =$

$1_M(n)$. To see that ϕ is surjective, let $y = (y_1, y_2, \ldots, y_n) \in \prod M_i$. Let $x = \sum y_i$.
Then $\phi_i(x) = y_i$, so $\phi(x) = y$. $\qquad\square$

We now use the functorial properties of the tensor product to establish (5) for the case when the index set I is finite.

Lemma 4.2.4. *Let M be a K-semimodule and $N = \bigoplus_{i=1}^n N_i$ be a direct sum of K-semimodules. Then there is an isomorphism*

$$M \otimes \left(\bigoplus_{i=1}^n N_i\right) \cong \bigoplus_{i=1}^n (M \otimes N_i).$$

such that

$$m \otimes (n_1, n_2, \ldots, n_n) \to (m \otimes n_1, m \otimes n_2, \ldots, m \otimes n_n).$$

Proof. There are projections $\pi_i : N \to N_i \subseteq N$ such that

$$\pi_i \circ \pi_i = \pi_i \qquad \pi_i \circ \pi_j = 0 \quad \text{if } i \neq j,$$

$$\sum_{i=1}^n \pi_i = 1_N.$$

Consider the functor τ_M from Example 4.2.1. If we apply τ_M to the π_i's, we get a direct sum decomposition of $\tau_M(N) = M \otimes N$, by Lemma 4.2.3 and the remarks in Example 4.2.1. The direct summands are the $M \otimes N_i$'s. $\qquad\square$

To prove (5), we paste together the isomorphisms from Lemma 4.2.4 for all finite subsets of an arbitrary index set.

Proof of (5). Let I be an arbitrary index set and $M, \{N_i \mid i \in I\}$ be K-semimodules. We wish to construct an isomorphism

$$M \otimes \left(\bigoplus_{i \in I} N_i\right) \to \bigoplus_{i \in I} (M \otimes N_i).$$

We construct a K-linear map with this type, then argue that it is an isomorphism. By the universal property of the tensor product, such a map corresponds to a K-bilinear map

$$M \times (\bigoplus_{i \in I} N_i) \to \bigoplus_{i \in I}(M \otimes N_i).$$

Now, for any finite subset $S \subseteq I$, we have

$$M \otimes (\bigoplus_{i \in S} N_i) \cong \bigoplus_{i \in S}(M \otimes N_i)$$

by Lemma 4.2.4. Again by the universal property of the tensor product, this defines a K-bilinear map

$$M \times (\bigoplus_{i \in S} N_i) \cong \bigoplus_{i \in S}(M \otimes N_i).$$

The inclusion map $S \to I$ induces a K-linear map

$$\bigoplus_{i \in S}(M \otimes N_i) \to \bigoplus_{i \in I}(M \otimes N_i).$$

Composing these two maps yields a K-bilinear map

$$\phi_S : M \times (\bigoplus_{i \in S} N_i) \to \bigoplus_{i \in I}(M \otimes N_i)$$

for any finite $S \subseteq I$.

The desired map can then be described as follows: for any $x \in M \times (\bigoplus_{i \in I} N_i)$ of the form (m, n), where $n \neq 0$, find the smallest finite $S \subseteq I$ such that all of the nonzero components of x are in N_s for some $s \in S$. We can then view x as an element of $M \times (\bigoplus_{s \in S} N_s)$. The image of x is ϕ_S. Elements of the form $(m, 0)$ are sent to the zero element of $\bigoplus_{i \in I}(M \otimes N_i)$.

We must show that the composite map is K-multilinear. This follows from two facts. Let $S \subseteq S'$. First, there is an injection

$$M \times (\bigoplus_{s \in S} N_s) \to M \times (\bigoplus_{s' \in S'} N_{s'}).$$

Second, the restriction of $\phi_{S'}$ to S, i.e., to elements of the form (m, n), where $n \in \bigoplus_{s \in S} N_s$, is the same function as ϕ_S.

An inverse for the composite map can be constructed in a similar manner.

$\square$

Proof of (6). Suppose first that N is a free K-semimodule with basis $\{n\}$ and M is a K-semimodule. The proof of (3) exhibits an isomorphism

$$M \to M \otimes N$$

$$m \mapsto m \otimes n.$$

Note that any element of $M \otimes N$ can be written as a sum of terms of the form $m \otimes kn$ for $k \in K$ and $m \in M$. Given such a sum,

$$\sum_{i=1}^{n} m_i \otimes k_i n = \sum_{i=1}^{n} k_i m_i \otimes n = (\sum_{i=1}^{n} k_i m_i) \otimes n.$$

Hence each element of $M \otimes N$ can be written as $m \otimes n$ for some $m \in M$. The aforementioned isomorphism shows that each element of $M \otimes N$ can be written uniquely in the form $m \otimes n$.

Now suppose that N is the free K-semimodule on some set $\{n_i\}_{i \in I}$. By (5), $M \otimes N$ is isomorphic to $\bigoplus_{i \in I}(M \otimes N_i)$, where each N_i is the free K-semimodule on n_i. Therefore, using the above argument, we can write each element of $M \otimes N$ uniquely as a term

$$\sum_{i \in I} m_i \otimes n_i$$

with $m_i \in M$ and almost all of the m_i's equal to 0.

Finally, suppose that M is a free K-semimodule on a set $\{m_j\}_{j \in J}$. Arguing similarly, each element of $M \otimes N$ can expressed uniquely as a finite K-linear

combination of terms of the form $m_j \otimes n_i$. This implies that $M \otimes N$ is the free K-semimodule on basis $\{m_j \otimes n_i\}_{j \in J, i \in I}$. $\square$

Theorem 4.2.2. *The category* $K-\textbf{Mod}$ *is a symmetric monoidal category with bifunctor* $\otimes_K$. *The unit object is* K *considered as a* K-*semimodule over itself.*

Proof. The associator is given by Theorem 4.2.1.1. The left unit map is given by Theorem 4.2.1.3, and the right unit is defined similarly. The symmetry is the K-linear map in 4.2.1.4. That $\otimes$ is a bifunctor follows from Lemma 4.2.2. We must also verify the commutativity of the pentagonal diagram and the naturality of the associator, left unit, right unit, and symmetry isomorphisms. These follow immediately from the diagrams. $\square$

4.3 Alternative Constructions

The construction of the tensor product over a semiring in [10] is based on a standard construction of a tensor product over an arbitrary (not necessarily commutative) ring. We summarize the standard construction (see [7] for details) and explain the difficulties in generalizing it to semirings. In this section, let R be a ring with 1 which is not necessarily commutative.

Definition 4.3.1. Let M be a right R-module, N be a left R-module, and L an abelian group. A map $f : M \times N \to L$ is said to be *R-balanced* if it satisfies

$$f(m + m', n) = f(m, n) + f(m', n)$$

$$f(m, n + n') = f(m, n) + f(m, n')$$

$$f(mr, n) = f(m, rn)$$

for all $m, m' \in M$, $n, n' \in N$, and $r \in R$.

The tensor product of two R-modules M and N is a universal object representing R-balanced maps out of $M \times N$. To construct it, we first construct the free abelian group on the set $M \times N$ and then quotient out by a relation similar to that in Section 4.1. Denote the resulting group by $M \otimes_R N$ and let ι be induced R-balanced map $\iota : M \times N \to M \otimes_R N$. We have:

Theorem 4.3.1. (Theorem 10.10.4 [7]) *Let R be a ring with 1, M a right R-module, and N a left R-module.*

1. *If $\Phi : M \otimes_R N \to L$ is any group homomorphism from $M \otimes_R N$ to an abelian group L then the composite map $\phi = \Phi \circ \iota$ is an R-balanced map from $M \times N$ to L.*

2. *Conversely, suppose L is an abelian group and $\phi : M \times N \to L$ is any R-balanced map. Then there is a unique group homomorphism $\Phi : M \otimes_R N \to L$ such that ϕ factors through ι, i.e., $\phi = \Phi \circ \iota$.*

Note that the tensor product is just an abelian group — there is no R-module structure on it. In general, M is not a left R-module and N is not a right R-module, so we cannot expect $M \otimes_R N$ to be a right or left R-module. Hence we start with the free abelian group on $M \times N$, and not a free R-module on $M \times N$.

The construction of the tensor product of two semimodules in [10] is a generalization of this. It begins with taking the free commutative monoid on $M \times N$, where M is a right S-semimodule and N is a left S-semimodule for an arbitrary semiring S. In order to state the universal property of the tensor product constructed in [10], we first define a congruence relation on S-semimodules.

Definition 4.3.2. Let S be a semiring and M be a left S-semimodule. Let $\equiv$ be the congruence relation on M defined by $m \equiv m'$ if and only if there is an $m'' \in M$

with $m + m'' = m' + m''$. We denote the equivalence class of an element $m \in M$ by $[m]_\equiv$.

We now give the universal property of the tensor product in [10]. Note that an $\mathbb{N}$-semimodule is just a commutative monoid. We have:

Theorem 4.3.2. (Proposition 16.14 [10]) *Let R be a semiring, let M be a right R-semimodule, let N be a left R-semimodule, and let T be an $\mathbb{N}$-semimodule. If $\theta : M \otimes N \to T$ is an R-balanced function then there exists a unique $\mathbb{N}$-homomorphism $\psi : M \otimes_R N \to T/\equiv$ satisfying the condition that $\psi(m \otimes n) = [\theta(m, n)]_\equiv$ for all $m \in M$ and $n \in N$.*

We end this chapter with a lemma showing that the congruence relation in Definition 4.3.2 relates every two elements of an idempotent semimodule.

Lemma 4.3.1. *Let M be an idempotent S-semimodule for some semiring S, and let $m, m' \in M$. Then $m \equiv m'$.*

Proof. By idempotence, $m + (m + m') = m' + (m + m')$. $\qquad\square$

We now consider K-algebras, which are monoids in K$-$**Mod**. We also define K-coalgebras and K-bialgebras, which are related to K-algebras by categorical duality. Once we have the required definitions and theorems in place, we use these structures to define K-linear automata in the next chapter.

In this chapter, all semirings are commutative.

5.1 Monoids and K-algebras

We first recall the definition of an R-algebra for a commutative ring R.

Definition 5.1.1. Let R be a commutative ring. An R-*algebra* is a ring A along with a ring homomorphism $\eta : R \to A$ such that $\eta(R)$ is contained in the center of A. The homomorphism η is called the *unit* of the R-algebra. If η is an injection, it is common to abuse notation and write $R \subseteq A$.

Example 5.1.1. The set of $n \times n$ matrices over R is an R-algebra. The unit map sends $r \in R$ to $r \cdot I_n$, where I_n is the $n \times n$ identity matrix.

The map η defines an action of R on A via $r \rhd a = \eta(r)a$. This means we can view A as an R-module. Multiplication in A is then an R-bilinear map $A \times A \to A$. Therefore it corresponds to a unique R-linear map

$$\mu_A : A \otimes_R A \to A$$

$$\mu_A(a \otimes_R a') \to aa'.$$

We have the following.

Lemma 5.1.1. *An R-algebra is a monoid in the monoidal category of R-modules.*

Proof. We must argue that the diagrams in Definition 2.2.3 are satisfied. The associativity diagram is satisfied because multiplication in A is associative. That η satisfies the unit diagram is straightforward from the definitions. $\square$

Since K$-$**Mod** is a monoidal category, we can apply the diagrams in Definition 2.2.3.

Definition 5.1.2. Let K be a commutative semiring. A *K-algebra* is a monoid in K$-$**Mod**.

Example 5.1.2. Let K be a commutative semiring and Σ a finite alphabet. Let $K\Sigma^*$ denote the set of all finite K-linear sums of words in Σ^*. Concatenation of words can be extended K-linearly yielding an associative multiplication. This structure forms a K-algebra with unit map $\eta(k) = k \cdot \lambda$, where λ is the empty word.

Example 5.1.3. Let K be a commutative semiring. The semiring K can be thought of as a "K-algebra over itself" as follows. Multiplication in K defines a K-linear map $\mu_K : K \otimes K \to K$ such that $\mu_K(k_1 \otimes k_2) = k_1 k_2$. The map μ_K is an isomorphism by Theorem 4.2.1.3. There is also a unit map $\eta_K : K \to K$ given by $\eta_K(k) = k$ for all $k \in K$.

In a symmetric monoidal category, the product of two monoids is a monoid.

Theorem 5.1.1. (Lemma 9.2.12 [24]) *Let A, B be monoids in a symmetric monoidal category $\mathcal{C}$. The object $A \otimes B$ also has the structure of a monoid in $\mathcal{C}$. Multiplication is given by*

$$\mu_{A \otimes B} = (\mu_A \otimes \mu_B) \circ (1_A \otimes \sigma_{B,A} \otimes 1_B).$$

Diagrammatically,

$$A \otimes B \otimes A \otimes B \xrightarrow{1_A \otimes \sigma_{B,A} \otimes 1_A} A \otimes A \otimes B \otimes B \xrightarrow{\mu_A \otimes \mu_B} A \otimes B.$$

The unit arrow is given by

$$\eta_{A \otimes B} = \eta_A \otimes \eta_B.$$

Remark 5.1.1. In [24], the above Lemma is actually proved for *braided monoidal categories*. These are generalizations of symmetric monoidal categories in which the isomorphism $\sigma_{A,B} : A \otimes B \cong B \otimes A$ is not required to satisfy $\sigma_{A,B} \circ \sigma_{B,A} = 1_A$, but rather a set of weaker conditions known as the *hexagon conditions*.

Corollary 5.1.1. *Let A and B be K-algebras. Then $A \otimes B$ is a K-algebra with multiplication defined on simple tensors $a \otimes b, a' \otimes b' \in A \otimes B$ as*

$$(a \otimes b) \cdot (a' \otimes b') = aa' \otimes bb'$$

and extended linearly to all of $A \otimes B$. The unit map is the map

$$\eta : K \to A \otimes B$$

$$\eta(k) = 1 \otimes k = k \otimes 1.$$

We also define structure-preserving maps between K-algebras.

Definition 5.1.3. Let A, B be two K-algebras. A *K-algebra map* is a K-linear map $f : A \to B$ satisfying

$$
\begin{array}{ccc}
A \otimes A & \xrightarrow{f \otimes f} & B \otimes B \\
{\scriptstyle \mu_A} \downarrow & & \downarrow {\scriptstyle \mu_B} \\
A & \xrightarrow{f} & B
\end{array}
\qquad
\begin{array}{ccc}
 & K & \\
{\scriptstyle \eta_A} \swarrow & & \searrow {\scriptstyle \eta_B} \\
A & \xrightarrow{f} & B.
\end{array}
$$

This is just the definition of a morphism of monoids (Definition 2.2.4) applied to K$-$**Mod**.

5.2 Comonoids and K-coalgebras

Before defining comonoids, we recall the definitions of opposite categories, opposite functors, and inverse natural transformations.

Definition 5.2.1. Let C be a category. We define the *opposite category* of C, denoted C^{op}, as follows. The objects of C^{op} are the objects of C. For each arrow $f : a \to b$ of C, there is an arrow $f^{\mathrm{op}} : b \to a$ in C^{op}. The composition of $f^{\mathrm{op}} g^{\mathrm{op}} = (gf)^{\mathrm{op}}$ exists in C^{op} precisely when gf exists in C.

Definition 5.2.2. Let C, D be categories and $F : C \to D$ a functor. The *opposite functor* $F^{\mathrm{op}} : C^{\mathrm{op}} \to D^{\mathrm{op}}$ agrees with F on objects and sends an arrow g^{op} to $(F(g))^{\mathrm{op}}$.

Definition 5.2.3. Let A, B be categories, S, T functors from A to B, and $t : S \to T$ a natural isomorphism. The inverse of t, denoted t^{-1}, is the natural transformation $T \to S$ defined as follows:

$$t^{-1} : T \to S$$

$$t^{-1}(a) = (t(a))^{-1}$$

for all objects a of A.

Lemma 5.2.1. *Let* $B = \langle B, \otimes, e, a, l, r \rangle$ *be a (symmetric) monoidal category. Then* $B^{\mathrm{op}} = \langle B^{\mathrm{op}}, \otimes^{\mathrm{op}}, e, a^{-1}, l^{-1}, r^{-1} \rangle$ *is also a (symmetric) monoidal category.*

Proof. We are treating the inverse of an invertible arrow in B as an arrow in B^{op}. The relevant diagrams are easy to verify. $\square$

Definition 5.2.4. Let C be category. A *comonoid* in C is a monoid in C^{op}.

Definition 5.2.5. A comonoid in $\mathrm{K-}\mathbf{Mod}^{\mathrm{op}}$ is called a *K-coalgebra*.

Multiplication in a comonoid is called *comultiplication* and is traditionally denoted Δ. The unit of a comonoid is also known as the *counit* map and denoted by ϵ. In a K-coalgebra C, Δ is a K-linear map $C \to C \otimes C$ and ϵ is a K-linear map $C \to K$. The comultiplication and counit satisfy the following diagrams, known as the *coassociativity* and *counit* conditions:

$$
\begin{array}{ccc}
 & C \otimes C \otimes C & \\
\scriptstyle{\Delta \otimes 1_C} \nearrow & & \nwarrow \scriptstyle{1_C \otimes \Delta} \\
C \otimes C & & C \otimes C \\
\scriptstyle{\Delta} \nwarrow & & \nearrow \scriptstyle{\Delta} \\
 & C &
\end{array}
$$

$$
C \xrightarrow{\quad \Delta \quad} C \otimes C \underset{1_C \otimes \epsilon}{\overset{\epsilon \otimes 1_C}{\rightrightarrows}} C.
$$

with 1_C as the top arc from C to C.

These are simply the duals of the diagrams in Definition 2.2.3. Note that we have omitted the associator isomorphism and identified K, $C \otimes K$, and $K \otimes C$.

When performing calculations involving comultiplication, we sometimes use the notation

$$
\Delta(c) = \sum_i c_{(1)} \otimes c_{(2)}
$$

to express how c is "split" into elements of $C \otimes C$. Note that such a description of Δ requires the choice of a representative in the tensor product for each $c \in C$.

Example 5.2.1. Let K be the two-element idempotent semiring and $\Sigma = \{x, y\}$. Let P be the set of all finite K-linear sums of elements of Σ^* (cf. Example 5.1.2). Note that we can treat elements of P as polynomials in noncommuting variables x, y with coefficients in K (although multiplication in P is not needed to define a K-coalgebra).

Consider the following comultiplications on P, defined on monomials and

extended linearly:

$$\Delta_1(w) = w \otimes w$$

$$\Delta_2(w) = \sum_{w_1 w_2 = w} w_1 \otimes w_2.$$

Also consider the comultiplication defined as

$$\Delta_3(g) = 1 \otimes g + g \otimes 1, \text{ for } g \in \{x, y\}$$

and extended as an algebra map to all of P. That is, $\Delta_3(pq) = \Delta_3(p)\Delta_3(q)$ for all $p, q \in P$. For example,

$$\Delta_3(x)\Delta_3(y) = (1 \otimes x + x \otimes 1)(1 \otimes y + y \otimes 1)$$

$$= 1 \otimes xy + y \otimes x + x \otimes y + xy \otimes 1.$$

Note that the product $\Delta_3(x)\Delta_3(y)$ takes place in the tensor product $P \otimes P$. Moreover, we have two K-linear maps, ϵ_1 and ϵ_2, given by

$$\epsilon_1(p) = p(1, 1)$$

$$\epsilon_2(p) = p(0, 0)$$

for all $p \in P$. Then $(P, \Delta_1, \epsilon_1)$ is a K-coalgebra, as are $(P, \Delta_2, \epsilon_2)$ and $(P, \Delta_3, \epsilon_2)$.

Example 5.2.2. Let K be a commutative semiring. There is a K-coalgebra structure on K. Comultiplication is given by $\Delta_K(1) = 1 \otimes 1$, extended K-linearly. The counit ϵ_K is the identity map on K.

Let C and D be K-coalgebras. Since C and D are monoids in a symmetric monoidal category, there is a tensor product structure on $C \otimes D$. The counit of $C \otimes D$ is $\epsilon_C \otimes \epsilon_D$, and multiplication defined by the diagram

$$C \otimes D \xrightarrow{\Delta_C \otimes \Delta_D} C \otimes C \otimes D \otimes D \xrightarrow{1_C \otimes \sigma_{C,D} \otimes 1_D} C \otimes D \otimes C \otimes D.$$

This is just the multiplication and unit from Lemma 5.1.1 interpreted in an opposite category. There are also structure-preserving maps for K-coalgebras, which are just morphisms of monoids in $K-\mathbf{Mod}^{\mathrm{op}}$.

Definition 5.2.6. Let C and D be K-coalgebras. A *K-coalgebra map* is a K-linear map $g : C \to D$ satisfying

$$
\begin{array}{ccc}
C \otimes C & \xrightarrow{g \otimes g} & D \otimes D \\
\Delta_C \uparrow & & \uparrow \Delta_D \\
C & \xrightarrow{\quad g \quad} & D
\end{array}
\qquad\qquad
\begin{array}{ccc}
& K & \\
\epsilon_C \nearrow & & \nwarrow \epsilon_D \\
C & \xrightarrow{\quad g \quad} & D.
\end{array}
$$

5.3 Bimonoids and K-bialgebras

We now define bimonoids, which are structures with "compatible" monoid and comonoid structures.

Definition 5.3.1. Let C be a symmetric monoidal category. A *bimonoid* in C is a comonoid in the category of monoids of C.

So a bimonoid B in a symmetric monoidal category C is a comonoid in which the comultiplication and counit arrows are morphisms of monoids. An equivalent definition of a bimonoid is a monoid in the category of comonoids of C. Let B be a comonoid in the category of monoids of C. Let K denote the unit object of C. To show that B is also a monoid in the category of comonoids of C, we must show that μ_B and η_B are morphisms of comonoids.

Note that Δ_B satisfies

$$
\begin{array}{ccc}
B \otimes B & \xrightarrow{\Delta_B \otimes \Delta_B} & B \otimes B \otimes B \otimes B \\
\mu_B \downarrow & & \downarrow (\mu_B \otimes \mu_B) \circ (1_B \otimes \sigma_{B,B} \otimes 1_B) \\
B & \xrightarrow{\quad \Delta_B \quad} & B \otimes B
\end{array}
$$

since it is a morphism of monoids. Rotating this diagram $90°$ counterclockwise and reassociating yields

$$B \otimes B \otimes B \otimes B \xrightarrow{\mu_B \otimes \mu_B} B \otimes B$$

with vertical arrow $(1_B \otimes \sigma_{B,B} \otimes 1_B) \circ (\Delta_B \otimes \Delta_B)$ on the left from $B \otimes B$, vertical arrow Δ_B on the right to B, and

$$B \otimes B \xrightarrow{\mu_B} B$$

which shows that μ_B preserves comultiplication. Note the use of Lemma 5.1.1 to guarantee monoid and comonoid structures on $B \otimes B$.

Furthermore, ϵ_B is a morphism of monoids $B \to K$. By definition, the following diagram commutes:

$$
\begin{array}{ccc}
B \otimes B & \xrightarrow{\epsilon_B \otimes \epsilon_B} & K \otimes K \\
\downarrow{\scriptstyle \mu_B} & & \downarrow{\scriptstyle \mu_K} \\
B & \xrightarrow{\epsilon_B} & K.
\end{array}
$$

Rotating this diagram $90°$ counterclockwise and using the fact that $\mu_K : K \otimes K \to K$ is an isomorphism yields the diagram

$$
\begin{array}{ccc}
 & K & \\
{\scriptstyle \epsilon_B \otimes \epsilon_B} \nearrow & & \nwarrow {\scriptstyle \epsilon_B} \\
B \otimes B & \xrightarrow{\mu_B} & B,
\end{array}
$$

which shows that μ_B is a comonoid morphism $B \otimes B \to B$. Note that we are treating the unit object K as a "monoid over itself", which is a simple generalization of Example 5.1.3.

We must also show that $\eta_B : K \to B$ is a morphism of comonoids. Since $\Delta_B : B \to B \otimes B$ is a morphism of monoids, we have

$$
\begin{array}{ccc}
 & K & \\
{\scriptstyle \eta_B} \swarrow & & \searrow {\scriptstyle \eta_B \otimes \eta_B} \\
B & \xrightarrow{\Delta_B} & B \otimes B.
\end{array}
$$

Rotating this diagram $90°$ counterclockwise and using the fact that

$\Delta_K : K \to K \otimes K$ is an isomorphism yields the diagram

$$
\begin{array}{ccc}
K \otimes K & \xrightarrow{\eta_B \otimes \eta_B} & B \otimes B \\
\Delta_K \uparrow & & \uparrow \Delta_B \\
K & \xrightarrow{\eta_B} & B.
\end{array}
$$

This shows that η_B preserves comultiplication. To see that η_B preserves the counit, note that the diagram

$$
\begin{array}{ccc}
 & K & \\
{}^{\eta_B} \swarrow & & \searrow {}^{\eta_K} \\
B & \xrightarrow{\epsilon_B} & K
\end{array}
$$

commutes, since ϵ_B is a morphism of monoids. Rotating this diagram $90°$ counterclockwise yields

$$
\begin{array}{ccc}
 & K & \\
{}^{\eta_K} \nearrow & & \nwarrow {}^{\epsilon_B} \\
K & \xrightarrow{\eta_B} & B.
\end{array}
$$

Since $\eta_K = \epsilon_K$, η_B is a morphism of comonoids.

Definition 5.3.2. A K-bialgebra is a bimonoid "in" $K-$**Mod**. That is, a comonoid in the category of K-algebras, or a monoid in the category of K-coalgebras.

Let R be a commutative ring. In the literature, an R-bialgebra is frequently defined as an R-module which is both an R-algebra and an R-coalgebra, which satisfies the following diagrams:

$$
\begin{array}{ccccc}
B \otimes B & \xrightarrow{\mu_B} & B & \xrightarrow{\Delta_B} & B \otimes B \\
\Delta_B \otimes \Delta_B \downarrow & & & & \uparrow \mu_B \otimes \mu_B \\
B \otimes B \otimes B \otimes B & & \xrightarrow{1_B \otimes \sigma_{B,B} \otimes 1_B} & & B \otimes B \otimes B \otimes B
\end{array}
$$

$$
\begin{array}{ccccc}
B \otimes B & \xrightarrow{\epsilon_B \otimes \epsilon_B} R \otimes R \xrightarrow{\eta_B \otimes \eta_B} & B \otimes B & & B \\
\mu_B \downarrow & \cong \downarrow & \uparrow \Delta_B & {}^{\eta_B} \nearrow & \searrow {}^{\epsilon_B} \\
B & \xrightarrow{\epsilon_B} R \xrightarrow{\eta_B} & B & R & \xrightarrow{1_R} R.
\end{array}
$$

59

Note the "self-duality" of the defining diagrams of a R-bialgebra: swapping Δ for μ, ϵ for η, and reversing the direction of all arrows yields the same diagrams. We have that the following conditions are equivalent:

1. B is an R-bialgebra,

2. $\mu : B \otimes B \to B$ and $\eta : R \to B$ are R-coalgebra maps,

3. $\Delta : B \to B \otimes B$ and $\epsilon : B \to R$ are R-algebra maps.

This means that we could define a K-bialgebra as a K-semimodule with K-linear maps satisfying the above diagrams. This is the definition of a K-bialgebra used in [30]. When performing calculations, the following equational definition of a K-bialgebra is useful:

$$\Delta(ab) = \Delta(a)\Delta(b) \quad \Delta(1) = 1 \otimes 1 \quad \epsilon(ab) = \epsilon(a)\epsilon(b) \quad \epsilon(1) = 1.$$

Example 5.3.1. Using the notation of Example 5.2.1, straightforward calculations show that P with comultiplication Δ_1 and counit ϵ_1 is a K-bialgebra, as is P with comultiplication Δ_3 and counit ϵ_2. Note that Δ_2 does not satisfy $\Delta_2(x)\Delta_2(y) = \Delta_2(xy)$.

Example 5.3.2. Let M be a monoid and K a commutative semiring. Let $K(M)$ be the free K-semimodule on M. Define multiplication in $K(M)$ by extending multiplication in M K-linearly. Then $K(M)$ is also a K-algebra with unit map $\eta(k) = k1_M$. This is known as the *free K-algebra on M*. There is a K-coalgebra structure on $K(M)$; define

$$\Delta(m) = m \otimes m$$

$$\epsilon(m) = 1$$

for $m \in M$ and extend K-linearly to $K(M)$. A straightforward calculation shows that $K(M)$ is a K-bialgebra.

Finally, we define structure-preserving maps for K-bialgebras.

Definition 5.3.3. Let B, B' be K-bialgebras. A *K-bialgebra map* is a K-linear map $f : B \to B'$ which is both a K-algebra map and a K-coalgebra map.

CHAPTER 6

REPRESENTATIONS, AUTOMATA, AND LANGUAGES

In this chapter, we define K-linear automata as pointed, observable representations of K-algebras. We give the relation between a K-coalgebra structure on inputs to K-linear automata and the K-algebra structure on functions defined by K-linear automata, and describe how to run K-linear automata in parallel using a K-bialgebra structure on the inputs. These results explain and extend the similarities between the theory of automata and formal languages and the theory of bialgebras which are the subjects of [5], [6], [12], and [11]. Furthermore, we show the role of comultiplication as a parameter, for multiplying both languages and automata.

All semirings in this chapter are commutative.

6.1 K-algebra Representations and K-linear Automata

We treat elements of a K-algebra as inputs to an automaton. The transitions of the automaton correspond to an action of the input K-algebra on a K-semimodule, elements of which are the states of the automaton. These actions are just monoid actions in the monoidal category K$-$**Mod** (Definition 2.2.6).

Definition 6.1.1. Let A be a K-algebra and M be a K-semimodule. A *right action* of A on M is a K-linear map $M \otimes A \to M$, denoted $\triangleleft$, satisfying

$$m \triangleleft (aa') = (m \triangleleft a) \triangleleft a'$$

$$m \triangleleft 1 = m$$

62

for all $a, a' \in A, m \in M$.

Left actions are defined analogously as K-linear maps $\triangleright : A \otimes M \to M$.

Remark 6.1.1. Similar to the case for monoids in **Set**, a right action of A on M is equivalent to a K-linear map $A \to \mathrm{End}^r(M)$, where $\mathrm{End}^r(M)$ is the *right endomorphism semiring* of M. An analogous statement is true for left actions.

To define a K-linear automaton, we also need a start state and an observation function (cf. Definition 2.1.7).

Definition 6.1.2. A *right K-linear automaton* $\mathcal{A} = (M, A, s, \triangleleft, \Omega)$ consists of the following:

1. A K-algebra A, called the *input K-algebra*,

2. A K-semimodule M, called the *state semimodule*,

3. A right action $\triangleleft$ of A on M,

4. An element $s \in M$, called the *start vector*,

5. A K-linear map $\Omega : M \to K$, called the *observation function*.

Remark 6.1.2. Equivalently, we could have defined a K-linear *start function*

$$\alpha : K \to M$$

and set $s = \alpha(1)$. This is useful in Chapter 7 below, but can add unnecessary symbols to proofs. We use both variants, depending on the situation.

Left K-linear automata are defined similarly using a left action $\triangleright$. In the sequel, we give only "one side" of a theorem or definition involving K-linear automata; the other follows mutatis mutandis.
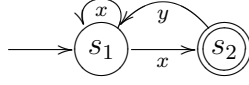
Figure 6.1: A Nondeterministic automaton to encode.

Example 6.1.1. We provide a translation of the nondeterministic automaton in Figure 6.1 into the framework of K-algebra representations.

Let K be the two-element idempotent semiring. Let M be the free K-semimodule on the set $\{s_1, s_2\}$ and let P the K-algebra of polynomials over noncommuting variables $\{x, y\}$ with coefficients from K. Define a right action of the generators of P (as a K-algebra) on M as follows:

$$\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft x = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft y = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

and extend algebraically to an action of P on M. The start vector is

$$\begin{bmatrix} 1 & 0 \end{bmatrix}$$

and the observation function is

$$\Omega \left(\begin{bmatrix} k_1 & k_2 \end{bmatrix} \right) = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Intuitively, this right K-linear automaton computes by beginning in the start state and reading the input from left to right. Each letter it reads in causes the automaton to change state; when the input has been consumed, the state is observed using the observation function Ω. Note that this could also be viewed as a weighted automaton; the difference is that we use the transition matrices to

define a right action of P on a free K-semimodule, instead of a right action of $\{x, y\}^*$.

This automaton can also be encoded as a left K-linear automaton, which begins in the "final" state, reads the input backward from right to left, and then observes whether or not the start state has been reached. In this case, P acts on the free K-semimodule on the set $\{s_1, s_2\}$ by left multiplication. This means that the states must be written as column vectors. The start state is

$$
\begin{bmatrix} 1 \\ 0 \end{bmatrix}
$$

and the observation function is

$$
\Omega\left(\begin{bmatrix} k_1 \\ k_2 \end{bmatrix}\right) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}.
$$

Such automata determine elements of $\mathrm{Lin}(A, K)$, as in [12] (cf. Definition 2.1.8).

Definition 6.1.3. Let $\mathcal{A} = (M, A, s, \lhd, \Omega)$ be a right K-linear automaton. The *language accepted* by $\mathcal{A}$ is the function $\rho_{\mathcal{A}} : A \to K$ such that

$$
\rho_{\mathcal{A}}(a) = \Omega(s \lhd a).
$$

Lemma 6.1.1. *The function $\rho_{\mathcal{A}}$ is an element of $\mathrm{Lin}(A, K)$.*

Proof. Immediate since $\lhd$ and Ω are K-linear maps. $\square$

Definition 6.1.4. Let $\mathcal{A}$ and $\mathcal{B}$ be right K-linear automata. If $\rho_{\mathcal{A}} = \rho_{\mathcal{B}}$, then $\mathcal{A}$ and $\mathcal{B}$ are said to be *equivalent*.

Functions between automata which preserve the language accepted are central to the theory of automata; such functions have K-linear analogs.

Definition 6.1.5. Let $\mathcal{A} = (M, A, s_\mathcal{A}, \lhd_\mathcal{A}, \Omega_\mathcal{A})$ and $\mathcal{B} = (N, A, s_\mathcal{B}, \lhd_\mathcal{B}, \Omega_\mathcal{B})$ be right K-linear automata. A *morphism of right K-linear automata* from $\mathcal{A}$ to $\mathcal{B}$ is a K-linear map $\phi : M \to N$ such that

$$\phi(s_\mathcal{A}) = s_\mathcal{B} \tag{6.1}$$

$$\phi(m \lhd_\mathcal{A} a) = \phi(m) \lhd_\mathcal{B} a \tag{6.2}$$

$$\Omega_\mathcal{A}(m) = \Omega_\mathcal{B}(\phi(m)) \tag{6.3}$$

for all $m \in M, n \in N, a \in A$.

Remark 6.1.3. Let V and W be R-modules. In the theory of R-algebras, an R-linear map $f : V \to W$ which satisfies (6.2) is known as a *linear intertwiner*. This is just an instance of a right action morphism in the category $\mathrm{K}-\mathbf{Mod}$.

Theorem 6.1.1. *Let $\mathcal{A} = (M, A, s_\mathcal{A}, \lhd_\mathcal{A}, \Omega_\mathcal{A})$ and $\mathcal{B} = (N, A, s_\mathcal{B}, \lhd_\mathcal{B}, \Omega_\mathcal{B})$ be right K-linear automata, and let $\phi : \mathcal{A} \to \mathcal{B}$ be a morphism of right K-linear automata. Then $\mathcal{A}$ and $\mathcal{B}$ are equivalent.*

Proof. Essentially the same as the proof of Theorem 2.1.1. $\qquad\qquad\square$

A simple calculation proves the following lemma.

Lemma 6.1.2. *Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be right K-linear automata and $\phi : \mathcal{A} \to \mathcal{B}, \phi' : \mathcal{B} \to \mathcal{C}$ be morphisms of right K-linear automata. Then $\phi' \circ \phi : \mathcal{A} \to \mathcal{C}$ is a morphism of right K-linear automata.*

Furthermore, for a right K-linear automaton $\mathcal{A}$, the identity map of the underlying K-semimodule of $\mathcal{A}$ is a morphism of right K-linear automata. We therefore have the following.

Lemma 6.1.3. *For a given commutative semiring K, the collection of right K-linear automata and morphisms thereof forms a category.*

Let A be a K-algebra. Elements of $\mathrm{Lin}(A, K)$ can be added and scaled by K, since $\mathrm{Lin}(A, K)$ is a K-semimodule by Lemma 3.3.1. Moreover, given right K-linear automata $\mathcal{A}$ and $\mathcal{B}$, there is a right K-linear automaton accepting $\rho_{\mathcal{A}} + \rho_{\mathcal{B}}$, and given $k \in K$, there is a right K-linear automaton accepting $k\rho_{\mathcal{A}}$.

Definition 6.1.6. Let $\mathcal{A} = (M, A, s_{\mathcal{A}}, \lhd_{\mathcal{A}}, \Omega_{\mathcal{A}})$ and $\mathcal{B} = (N, A, s_{\mathcal{B}}, \lhd_{\mathcal{B}}, \Omega_{\mathcal{B}})$ be right K-linear automata. The *direct sum* of $\mathcal{A}$ and $\mathcal{B}$ is the right K-linear automaton

$$\mathcal{A} \oplus \mathcal{B} = (M \oplus N, A, (s_{\mathcal{A}}, s_{\mathcal{B}}), \lhd_{\mathcal{A} \oplus \mathcal{B}}, \Omega_{\mathcal{A}} \oplus \Omega_{\mathcal{B}}),$$

where

$$\lhd_{\mathcal{A} \oplus \mathcal{B}} : (M \oplus N) \otimes A \to M \oplus N,$$

$$\lhd_{\mathcal{A} \oplus \mathcal{B}}((m, n) \otimes a) = ((m \lhd_{\mathcal{A}} a), (n \lhd_{\mathcal{B}} a))$$

and

$$\Omega_{\mathcal{A} \oplus \mathcal{B}} : M \oplus N \to K,$$

$$\Omega_{\mathcal{A} \oplus \mathcal{B}}(m, n) = \Omega_{\mathcal{A}}(m) + \Omega_{\mathcal{B}}(n).$$

The verification that $\lhd_{\mathcal{A} \oplus \mathcal{B}}$ is an action of A on $M \oplus N$ is straightforward.

Theorem 6.1.2. *Let $\mathcal{A} = (M, A, s_{\mathcal{A}}, \lhd_{\mathcal{A}}, \Omega_{\mathcal{A}})$ and $(N, A, s_{\mathcal{B}}, \lhd_{\mathcal{B}}, \Omega_{\mathcal{B}})$ be right K-linear automata. Then $\rho_{\mathcal{A} \oplus \mathcal{B}}(a) = \rho_{\mathcal{A}}(a) + \rho_{\mathcal{B}}(a)$ for all $a \in A$.*

Proof. For any $a \in A$,

$$\rho_{\mathcal{A} \oplus \mathcal{B}}(a) = \Omega_{\mathcal{A} \oplus \mathcal{B}}((s_{\mathcal{A}}, s_{\mathcal{B}}) \triangleleft_{\mathcal{A} \oplus \mathcal{B}} a)$$

$$= \Omega_{\mathcal{A} \oplus \mathcal{B}}(s_{\mathcal{A}} \triangleleft_{\mathcal{A}} a, s_{\mathcal{B}} \triangleleft_{\mathcal{B}} a)$$

$$= \Omega_{\mathcal{A}}(s_{\mathcal{A}} \triangleleft_{\mathcal{A}} a) + \Omega_{\mathcal{B}}(s_{\mathcal{B}} \triangleleft_{\mathcal{B}} a)$$

$$= \rho_{\mathcal{A}}(a) + \rho_{\mathcal{B}}(a).$$

$\square$

Theorem 6.1.3. *Let $\mathcal{A} = (M, A, s, \triangleleft, \Omega)$ be a right K-linear automaton, and let $k \in K$. Then $k\rho_{\mathcal{A}} = \rho_{\mathcal{A}'}$, where $\mathcal{A}' = (M, A, ks, \triangleleft, \Omega)$.*

Proof. For any $a \in A$, $\rho_{\mathcal{A}'} = \Omega(ks \triangleleft a) = k\Omega(s \triangleleft a) = k\rho_{\mathcal{A}}$ by linearity. $\square$

Algebra maps can be used to translate the input of an automaton.

Definition 6.1.7. Let A, A' be K-algebras and $f : A \to A'$ a K-algebra map. Suppose A' acts on a K-semimodule M. Then A also acts on M according to the formula

$$m \triangleleft a = m \triangleleft f(a)$$

for $a \in A, m \in M$. This is known as the *pullback* of the action of A'.

Automata theorists will recognize pullbacks as the main ingredient in the proof that regular languages are closed under inverse homomorphisms.

6.2 K-coalgebras and Formal Languages

Let C be a K-coalgebra. By Lemma 3.3.1, $\mathrm{Lin}(C, K)$ is a K-semimodule with the operations of pointwise addition and scalar multiplication. It is a standard fact

that the coalgebra structure of C defines an algebra structure on $\operatorname{Lin}(C, K)$.

Definition 6.2.1. Let (C, Δ, ϵ) be a K-coalgebra and $f, g \in \operatorname{Lin}(C, K)$. The *convolution product* of f and g, denoted $f * g$, is the element of $\operatorname{Lin}(C, K)$ defined by

$$f * g = \mu_K \circ (f \otimes g) \circ \Delta$$

(recall that μ_K denotes multiplication in K).

Lemma 6.2.1. *Let (C, Δ, ϵ) be a K-coalgebra. There is a K-algebra structure on $\operatorname{Lin}(C, K)$ with multiplication given by the convolution product and unit*

$$\eta : K \to C$$

$$\eta(k) = k\epsilon.$$

In particular, the multiplicative identity is ϵ.

Proof. The operation $*$ is associative because Δ is coassociative:

$$f * (g * h) = \mu_K(f \otimes (\mu_K(g \otimes h))) \circ ((1 \otimes \Delta) \circ \Delta)$$

$$(f * g) * h = \mu_K((\mu_K(f \otimes g)) \otimes h) \circ ((\Delta \otimes 1) \circ \Delta)$$

and coassociativity of Δ is exactly $((1 \otimes \Delta) \circ \Delta) = ((\Delta \otimes 1) \circ \Delta)$. The rest of the K-algebra requirements follow immediately from the definitions. $\square$

The relation between K-coalgebras and formal languages is as follows. Let K be the two-element idempotent semiring and let P be the set of polynomials over noncommuting variables $\{x, y\}$ with coefficients in K. Note that an element of $\operatorname{Lin}(P, K)$ is completely determined by its values on monomials, which we view as words over $\{x, y\}$. Thus there is a one-to-one correspondence between subsets of $\{x, y\}^*$ and elements of $\operatorname{Lin}(P, K)$.

Recall the comultiplications and counits on P from Example 5.2.1:

$$\Delta_1(w) = w \otimes w$$

$$\Delta_2(w) = \sum_{w_1 w_2 = w} w_1 \otimes w_2$$

defined on monomials w and extended K-linearly to all of P. The third comultiplication is

$$\Delta_3(x) = 1 \otimes x + x \otimes 1$$

$$\Delta_3(y) = 1 \otimes y + y \otimes 1$$

extended as an algebra map to all of P. The counit maps are

$$\epsilon_1(p) = p(1, 1)$$

$$\epsilon_2(p) = p(0, 0)$$

for all $p \in P$. Then $(P, \Delta_1, \epsilon_1)$ is a K-coalgebra as are $(P, \Delta_2, \epsilon_2)$ and $(P, \Delta_3, \epsilon_2)$.

A simple verification shows that the K-algebra on $\mathrm{Lin}(P, K)$ determined by the K-coalgebra $(P, \Delta_1, \epsilon_1)$ corresponds to language intersection, with the multiplicative identity corresponding to the language denoted by $(x + y)^*$. The K-coalgebra $(P, \Delta_2, \epsilon_2)$ corresponds to language concatenation with identity $\{\lambda\}$, where λ is the empty word. Finally, the K-coalgebra $(P, \Delta_3, \epsilon_2)$ corresponds to the shuffle product of languages, again with identity $\{\lambda\}$ (see [5] and also [24], Proposition 5.1.4). In each case, addition in the K-algebra on $\mathrm{Lin}(P, K)$ corresponds to the union of two languages.

We conclude this section with an example calculation. Let $f \in \mathrm{Lin}(P, K)$ correspond to the language denoted by x^*, and let $g \in \mathrm{Lin}(P, K)$ correspond to the language denoted by y^*. The following shows that $yx \in f * g$, where the

comultiplication is Δ_3:

$$
\begin{aligned}
\mu_k \circ f \otimes g \circ \Delta_3(xy) &= \mu_k \circ f \otimes g(1 \otimes xy + y \otimes x + x \otimes y + xy \otimes 1) \\
&= \mu_K(f(1) \otimes g(xy) + f(y) \otimes g(x) + f(x) \otimes g(y) + f(xy) \otimes g(1)) \\
&= \mu_K(1 \otimes 0 + 0 \otimes 0 + 1 \otimes 1 + 0 \otimes 1) \\
&= 0 + 0 + 1 + 0 \\
&= 1.
\end{aligned}
$$

6.3 K-bialgebras, Automata, and Languages

A K-linear automaton takes elements of a K-algebra A input. These automata define elements of $\mathrm{Lin}(A, K)$. Moreover, a K-coalgebra structure on A defines a multiplication on $\mathrm{Lin}(A, K)$. We now discuss the relation between these products on $\mathrm{Lin}(A, K)$ and K-linear automata.

We first treat the case in which A is both a K-algebra and a K-coalgebra, without assuming that A is a K-bialgebra. Let $\mathcal{A} = (M, A, s_\mathcal{A}, \lhd_\mathcal{A}, \Omega_\mathcal{A})$ and $\mathcal{B} = (N, A, s_\mathcal{B}, \lhd_\mathcal{B}, \Omega_\mathcal{B})$ be K-linear automata. Applying the convolution product to $\rho_\mathcal{A}$ and $\rho_\mathcal{B}$ yields

$$
\rho_\mathcal{A} * \rho_\mathcal{B}(a) = \mu_K \circ \Big(\sum_i \rho_\mathcal{A}(s_\mathcal{A} \lhd a_{(1)}) \otimes \rho_\mathcal{B}(s_\mathcal{B} \lhd a_{(2)}) \Big).
$$

That is, the convolution product determines a formula with comultiplication as a parameter. Different choices of comultiplication yield different products of languages, as discussed in Section 6.2. When the languages are given by K-linear automata, we can use this formula to obtain a succinct expression for the product of the two languages.

Of course, it would be even better if we could get a K-linear automaton accepting the product of the two languages. When the input forms a K-bialgebra, there is an easy way to construct such an automaton, which relies on a construction from the theory of bialgebras. In fact, this construction is an instance of a general theorem about categories of representations of a bimonoid.

We emphasize that a K-bialgebra structure is not necessary for a K-linear automaton accepting $\rho_\mathcal{A} * \rho_\mathcal{B}$ to exist. Consider Δ_2 and Δ_3 as defined in Section 6.2. They agree on x and y, which generate P as a K-algebra. Therefore at most one of them can be a K-algebra map; Δ_3 is a K-algebra map by definition. Hence Δ_2 is not part of a K-bialgebra and so we cannot use the construction to get a K-linear automaton "accepting" the concatenation of two languages. Such a K-linear automaton exists, of course, but it is not given by this construction.

It is a theorem that a category of representations of a bimonoid is itself a monoidal category. The proof, in full categorical generality, can be found in Chapter 15 of [28]. A proof for the special case of a category of representations of a bialgebra over a field can be found in Example 9.1.3 of [24]. We provide the interesting part of the proof for the special case of K-bialgebras.

Lemma 6.3.1. *Let B be a K-bialgebra which acts on K-semimodules M and N from the right. Then B acts on $M \otimes N$ from the right according to the diagram*

$$M \otimes N \otimes B \xrightarrow{1 \otimes \Delta_B} M \otimes N \otimes B \otimes B \xrightarrow{1 \otimes \sigma \otimes 1} M \otimes B \otimes N \otimes B \xrightarrow{\triangleleft_M \otimes \triangleleft_N} M \otimes N.$$

Proof. It is easy to see that the action of B on $M \otimes N$ is a K-linear map such that $(m \otimes n) \triangleleft_{M \otimes N} 1 = m \otimes n$. To see that

$$(m \otimes n) \triangleleft_{M \otimes N} ab = ((m \otimes n) \triangleleft_{M \otimes N} b) \triangleleft_{M \otimes N} a,$$

we work with the equational definition of the action, which is

$$(m \otimes n) \lhd_{M \otimes N} c = \sum_i m \lhd_M c_{(1)} \otimes n \lhd_N c_{(2)}.$$

We have

$$(m \otimes n) \lhd_{M \otimes N} (ab) = \sum_i m \lhd_M (ab)_{(1)} \otimes n \lhd_N (ab)_{(2)}$$

$$= \sum_i m \lhd_M a_{(1)}b_{(1)} \otimes n \lhd_N a_{(2)}b_{(2)}$$

$$= ((m \otimes n) \lhd_{M \otimes N} b) \lhd_{M \otimes N} a.$$

$\square$

Definition 6.3.1. Let $\mathcal{A} = (M, B, s_\mathcal{A}, \lhd_\mathcal{A}, \Omega_\mathcal{A})$ and $\mathcal{B} = (N, B, s_\mathcal{B}, \lhd_\mathcal{B}, \Omega_\mathcal{B})$ be right K-linear automata. The *tensor product* of $\mathcal{A}$ and $\mathcal{B}$, denoted $\mathcal{A} \otimes \mathcal{B}$, is the automaton $(M \otimes N, B, s_\mathcal{A} \otimes s_\mathcal{B}, \lhd_{M \otimes N}, \Omega_\mathcal{A} \otimes \Omega_\mathcal{B})$.

Remark 6.3.1. Note that since $K \otimes K \cong K$, $\Omega_M \otimes \Omega_N : M \otimes N \to K$. Similarly, if we view the start states as start functions $K \to M$, $K \to N$, then the start function of $\mathcal{A} \otimes \mathcal{B}$ has domain K.

Theorem 6.3.1. *Let $\mathcal{A} = (M, B, s_\mathcal{A}, \lhd_\mathcal{A}, \Omega_\mathcal{A})$ and $\mathcal{B} = (N, B, s_\mathcal{B}, \lhd_\mathcal{B}, \Omega_\mathcal{B})$ be right K-linear automata. Then $\rho_{\mathcal{A} \otimes \mathcal{B}} = \rho_\mathcal{A} * \rho_\mathcal{B}$.*

Proof. For any $b \in B$,

$$\rho_{\mathcal{A} \otimes \mathcal{B}}(b) = \Omega_{\mathcal{A} \otimes \mathcal{B}}((s_\mathcal{A} \otimes s_\mathcal{B}) \lhd_{\mathcal{A} \otimes \mathcal{B}} b)$$

$$= \Omega_{\mathcal{A} \otimes \mathcal{B}}\left(\sum_i (s_\mathcal{A} \lhd_\mathcal{A} b_{(1)} \otimes s_\mathcal{B} \lhd_\mathcal{B} b_{(2)}) \right)$$

$$= \sum_i \Omega_\mathcal{A}(s_\mathcal{A} \lhd_\mathcal{A} b_{(1)}) \Omega_\mathcal{B}(s_\mathcal{B} \lhd_\mathcal{B} b_{(2)})$$

$$= \rho_\mathcal{A} * \rho_\mathcal{B}(b).$$

□

This construction also appears in automata theory. It is used to run two automata in parallel.

Example 6.3.1. The automata in Figure 6.2 accept the languages denoted by $(xx)^*$ and $(yy)^*$, respectively. We provide the tensor product of the K-linear encodings of these automata, using the comultiplication Δ_3 from Example 5.2.1. We assume that both automata have input K-algebra $K\{x, y\}^*$; the action of y on the K-semimodule of the first automaton is given by the 2×2 matrix of 0's, as is the action of x on the K-semimodule of the second.

The K-semimodule of the tensor product is the free K-semimodule on the set $\{s_1 \otimes t_1, s_1 \otimes t_2, s_2 \otimes t_1, s_2 \otimes t_2\}$, by Lemma 4.2.1.6. The start vector is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix},$$

the right x, y actions are given by

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 6.2: Nondeterministic automata.

respectively, and the observation function is given by

$$\begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Using the comultiplication Δ_1 instead of Δ_3 would yield an automaton accepting the intersection of these two languages.

We can also multiply morphisms of K-linear automata.

Theorem 6.3.2. *Let B be a K-bialgebra. Let $\mathcal{A} = (M, B, s_{\mathcal{A}}, \lhd_{\mathcal{A}}, \Omega_{\mathcal{A}})$, $\mathcal{B} = (N, B, s_{\mathcal{B}}, \lhd_{\mathcal{B}}, \Omega_{\mathcal{B}})$, $\mathcal{C} = (M', B, s_{\mathcal{C}}, \lhd_{\mathcal{C}}, \Omega_{\mathcal{C}})$, and $\mathcal{D} = (N', B, s_{\mathcal{D}}, \lhd_{\mathcal{D}}, \Omega_{\mathcal{D}})$ be right K-linear automata. Let $f : \mathcal{A} \to \mathcal{C}$ and $g : \mathcal{B} \to \mathcal{D}$ be morphisms of right K-linear automata. Then*

$$f \otimes g : M \otimes N \to M' \otimes N'$$

is a morphism of right K-linear automata

$$\mathcal{A} \otimes \mathcal{B} \to \mathcal{C} \otimes \mathcal{D}.$$

Proof. The K-linear map $f \otimes g$ takes the start state of $\mathcal{A} \otimes \mathcal{B}$ to the start state of $\mathcal{C} \otimes \mathcal{D}$:

$$(f \otimes g)(s_{\mathcal{A}} \otimes s_{\mathcal{B}}) = f(s_{\mathcal{A}}) \otimes g(s_{\mathcal{B}}) = s_{\mathcal{C}} \otimes s_{\mathcal{D}}.$$

75

It also satisfies the condition on the observation functions of $\mathcal{A} \otimes \mathcal{B}$ and $\mathcal{C} \otimes \mathcal{D}$:

$$\Omega_{\mathcal{A} \otimes \mathcal{B}}(m \otimes n) = \Omega_{\mathcal{A}}(m) \otimes \Omega_{\mathcal{B}}(n)$$
$$= \Omega_{\mathcal{C}}(f(m)) \otimes \Omega_{\mathcal{D}}(g(n))$$
$$= \Omega_{\mathcal{C} \otimes \mathcal{D}}(f(m) \otimes g(n))$$
$$= \Omega_{\mathcal{C} \otimes D}((f \otimes g)(m \otimes n)).$$

Finally, we show that $f \otimes g$ commutes with the actions of $\mathcal{A} \otimes \mathcal{B}$ and $\mathcal{C} \otimes \mathcal{D}$:

$$(f \otimes g)((m \otimes n) \triangleleft_{\mathcal{A} \otimes \mathcal{B}} b) = (f \otimes g)\left(\sum_i (m \triangleleft_{\mathcal{A}} b_{(1)} \otimes n \triangleleft_{\mathcal{B}} b_{(2)}) \right)$$
$$= \sum_i f(m \triangleleft_{\mathcal{A}} b_{(1)}) \otimes g(n \triangleleft_{\mathcal{B}} b_{(2)})$$
$$= \sum_i f(m) \triangleleft_{\mathcal{C}} b_{(1)} \otimes g(n) \triangleleft_{\mathcal{D}} b_{(2)}$$
$$= ((f \otimes g)(m \otimes n)) \triangleleft_{\mathcal{C} \otimes \mathcal{D}} b.$$

$\square$

There is also a "unit" K-linear automaton, which accepts the counit of B.

Definition 6.3.2. Let B be a K-bialgebra. The *unit K-linear automaton* is the automaton $(K, B, 1, \triangleleft, \mathrm{id}_K)$, where

$$\triangleleft : K \otimes B \to K$$

$$k \otimes b \mapsto k\epsilon(b).$$

It accepts the multiplicative identity for the convolution product, i.e., the counit of B.

Proof. The counit satisfies $\epsilon(bb') = \epsilon(b)\epsilon(b')$. $\square$

The category of representations of B is a monoidal category. Since every K-linear automaton with input K-bialgebra B is just a representation of B endowed with a start state and and observation function, the category of K-linear automata with input K-bialgebra B is also a monoidal category.

CHAPTER 7

COMPLETENESS

In this chapter, we focus on the relation between K-linear automata and deterministic automata. In particular, we treat the notion of determinization of automata as a functor. In fact, this functor is part of an adjunction between certain categories of K-linear automata and deterministic automata. This allows us to transfer automata and morphisms from one category to the other. With this adjunction, we can prove a completeness theorem for a proof system for equivalence of K-linear automata. The only "rules of inference" in this proof system are morphisms of K-linear automata.

All automata (deterministic and K-linear) in the chapter are right. As usual, similar theorems hold for left automata. Moreover, in this chapter, automata have start functions rather than start states (cf. Remark 6.1.2).

7.1 Why Determinize?

To produce a witness to the equality of two K-linear automata, it is not necessary to determinize. Instead, we could use the same procedure used in the proof of Theorem 2.3.2.

Definition 7.1.1. A K-linear observable representation is a K-linear automaton without a start function.

Theorem 7.1.1. Let A be a K-algebra. Then $\mathrm{Lin}(A, K)$ has a K-linear observable representation structure.

Proof. By 3.3.1, $\mathrm{Lin}(A, K)$ is a K-semimodule. For $f \in \mathrm{Lin}(A, K)$, the map

$$\Omega_{\mathcal{F}}(f) = f(1)$$

is a K-linear map $\mathrm{Lin}(A, K) \to K$.

We define a right action of A on $\mathrm{Lin}(A, K)$ as follows. Let $a \in A$ and $f \in \mathrm{Lin}(A, K)$. Then

$$(f \vartriangleleft_{\mathcal{F}} a)(x) = f(ax)$$

for all $x \in A$. The proof that $\vartriangleleft_{\mathcal{F}}$ is a K-linear map $\mathrm{Lin}(A, K) \otimes A \to \mathrm{Lin}(A, K)$ is straightforward. The proof that $\vartriangleleft_{\mathcal{F}}$ is an action is essentially the same as Lemma 2.3.1. $\qquad\square$

Definition 7.1.2. A *morphism of K-linear observable representations* is defined as a morphism of K-linear automata, without the condition on the start states.

Definition 7.1.3. Let $\mathcal{A} = (A, M, \vartriangleleft_{\mathcal{A}}, \Omega_{\mathcal{A}})$ be a K-linear observable representation and $m \in M$. The *language accepted by m*, denoted L_m, is the K-linear map

$$L_m : A \to K$$

$$L_m(a) = \Omega_{\mathcal{A}}(m \vartriangleleft_{\mathcal{A}} a).$$

Theorem 7.1.2. *The K-linear observable representation on $\mathrm{Lin}(A, K)$ is a final object in the category of K-linear observable representations.*

Proof. Let $\mathcal{A} = (A, M, \vartriangleleft_{\mathcal{A}}, \Omega_{\mathcal{A}})$ be a K-linear observable representation and $\mathcal{F} = (A, \mathrm{Lin}(A, K), \vartriangleleft_{\mathcal{F}}, \Omega_{\mathcal{F}})$ be defined as in Theorem 7.1.1. We claim that the map

$$g : M \to \mathrm{Lin}(A, K)$$

$$g(m) = L_m$$

is the unique morphism of K-linear observable representations $\mathcal{A} \to \mathcal{F}$.

The proof is essentially the same as the proof of Lemma 2.3.2, plus an easy verification that g is a K-linear map. □

Let $\mathcal{A}$ and $\mathcal{B}$ be two equivalent K-linear automata. A witness to the equivalence of $\mathcal{A}$ and $\mathcal{B}$ is given by the following sequence of K-linear automata and morphisms:

$$\mathcal{A} \xrightarrow{g} \mathrm{Lin}(A, K) \xleftarrow{g'} \mathcal{B},$$

where g and g' are the unique morphisms described in Theorem 7.1.2. Note that we must define a K-linear automaton structure on $\mathrm{Lin}(A, K)$ by declaring the start function to map $1 \in K$ to the language accepted by $\mathcal{A}$, which is the same as the language accepted by $\mathcal{B}$.

We can even eliminate inaccessible states, using the following lemma.

Lemma 7.1.1. *Let A be a K-algebra, M a K-semimodule, and $\triangleleft$ a right action of A on M. Fix an element $m \in M$. Then the set $m_A = \{m' \mid m' = m \triangleleft a \text{ for some } a \in A\}$ is a subsemimodule of M.*

Proof. Let $n, n' \in m_A$. Then there must be $a, a' \in A$ such that $n = m \triangleleft a$ and $n' = m \triangleleft a'$. Then $(n + n') = m \triangleleft (a + a')$, so m_A is closed under addition. Furthermore, if $n = m \triangleleft a$, then $kn = m \triangleleft ka$, so m_A is closed under scalar multiplication. □

Nonetheless, we do determinize below, using the forgetful functor from $\mathrm{K-}\mathbf{Mod}$ to $\mathbf{Set}$. When K is the two-element idempotent semiring, this functor corresponds to determinization via the familiar subset construction. For other

K, it is possible for the forgetful functor to take a K-linear automaton whose underlying K-semimodule is the free K-semimodule on a finite set X and return a deterministic automaton with infinitely many states. This is not surprising; if the range of the language accepted by a deterministic automaton D is infinite, then D must have infinitely many states. Even for K the two-element idempotent semiring there are nondeterministic automata with n states such that any equivalent deterministic automaton requires a number of states exponential in n. In other words, a K-semimodule structure can be used to reduce the space needed to describe the states of an automaton significantly.

However, there are some disadvantages to K-linear automata. For example, there might not be a nice way to describe an arbitrary K-semimodule, in particular, $\mathrm{Lin}(A, K)$. The completeness proof below uses a free functor after determinizing. Free K-semimodules do have nice presentations (if K does), particularly free K-semimodules on finite sets. Moreover, maps between such K-semimodules can be described with matrices. In fact, in certain cases, the witnesses produced below can be verified in polynomial time.

Furthermore, when K is the two-element idempotent semiring, the proof system below can be used as a proof system for the equivalence of nondeterministic finite automata. It is well-known that this is a $PSPACE$-complete problem. So for any proof system, there must be equivalences whose shortest proof is exponentially long, unless $PSPACE = NP$. In general, it seems likely that any way of producing witnessing sequences must in the worst case either produce very large sequences, or sequences which are not efficiently verifiable.

7.2 Determinization as Forgetful Functor

Let $\mathcal{A} = (M, A, \alpha, \vartriangleleft, \Omega)$ be a K-linear automaton. We wish to construct a deterministic automaton D which is a "deterministic version" of A. However, this requires associating to the input K-algebra of A a monoid in **Set**. There is also a concern regarding the types of the two observation functions. In certain cases, these difficulties can be overcome using the notion of an adjunction between categories. There are many equivalent definitions of adjunctions used in practice. We recall the one best suited to our purposes.

Definition 7.2.1. Let $\mathfrak{A}$ and $\mathfrak{D}$ be categories, F a functor from $\mathfrak{D}$ to $\mathfrak{A}$, and U a functor from $\mathfrak{A}$ to $\mathfrak{D}$. An *adjunction* from $\mathfrak{D}$ to $\mathfrak{A}$ is a bijection ψ which assigns to each arrow $f : F(D) \to A$ of $\mathfrak{A}$ an arrow $\psi f : D \to U(A)$ of $\mathfrak{D}$, called the *right adjunct of f*, such that

$$\psi(f \circ Fh) = (\psi f) \circ h,$$

$$\psi(k \circ f) = Uk \circ (\psi f)$$

holds for all f and all arrows $h : D' \to D$ and $k : A \to A'$. Equivalently, for every arrow $g : D \to U(A)$,

$$\psi^{-1}(gh) = \psi^{-1}g \circ (Fh),$$

$$\psi^{-1}(Uk \circ g) = k \circ (\psi^{-1}g).$$

Example 7.2.1. Let U' be the forgetful functor from K-**Mod** to **Set** and F' the corresponding free functor that takes a set X to the free K-semimodule on X. Let M be a K-semimodule. The adjunction θ from **Set** to K-**Mod** takes as input a K-linear map $\phi : F'(X) \to M$ and returns the set map $X \to U'(M)$ obtained by restricting ϕ to X.

Remark 7.2.1. We use the notation of Example 7.2.1 throughout the sequel, namely, F', U', and θ.

Example 7.2.2. Let M be a monoid in **Set** and $K(M)$ be the free K-algebra on M as defined in Example 5.3.2. There is an adjunction between the category of monoids in **Set** and the category of K-algebras. The forgetful functor takes a K-algebra A and forgets the K-semimodule structure.

Our goal is to construct a "determinizing" functor from a category of K-linear automata to a category of deterministic automata, and a "free K-linear" functor in the opposite direction. We then wish to show that these two functors are components of an adjunction. In order for this to work nicely, we make the following assumptions about the inputs.

1. The input K-algebra of the K-linear automaton is $K(I)$, the free K-algebra on a fixed monoid I.

2. The input monoid of the deterministic automaton is I and the set of observations of the deterministic automaton is the underlying set of K.

When considering start functions, we treat K as $F'(\star)$. That is, we treat K as the free K-semimodule on a one-element set.

Let $\mathfrak{A}$ be a category of K-linear automata and morphisms of K-linear automata, satisfying assumption 1, and let $\mathfrak{D}$ be a category of deterministic automata and morphisms of deterministic automata satisfying assumption 2. We now define a functor U from $\mathfrak{A}$ to $\mathfrak{D}$.

On K-linear automata, U behaves as follows. Given a K-linear automaton

$$\mathcal{A} = (M, K(I), \alpha, \triangleleft, \Omega),$$

$$U(\mathcal{A}) = (U'(M), I, \theta(\alpha), \triangleleft', U'(\Omega), U'(K)),$$

where $\triangleleft'$ is defined as follows. The action $M \triangleleft K(I)$ is equivalent to a K-algebra map

$$K(I) \to \mathrm{End}^r(M).$$

Restricting this action to I and forgetting the K-semimodule structure on M yields a monoid homomorphism $\triangleleft' : I \to \mathrm{End}^r(U'(M))$.

We now define U on arrows of $\mathfrak{A}$. Let $\mathcal{A} = (M, K(I), \alpha_{\mathcal{A}}, \triangleleft_{\mathcal{A}}, \Omega_{\mathcal{A}})$ and $\mathcal{B} = (N, K(I), \alpha_{\mathcal{B}}, \triangleleft_{\mathcal{B}}, \Omega_{\mathcal{B}})$ be K-linear automata. A morphism of K-linear automata $\phi : \mathcal{A} \to \mathcal{B}$ is, in particular, a K-linear map $M \to N$. Define $U(\phi)$ to be the underlying set map $U'(\phi)$. To show that U takes arrows of $\mathfrak{A}$ to arrows of $\mathfrak{D}$, we must show that the commutativity of

implies the commutativity of

The transition and observation diagrams commute because the functor U' takes commutative diagrams to commutative diagrams. To show that the start function diagram commutes, note that

$$\theta(\phi \circ \alpha_{\mathcal{A}}) = U'(\phi) \circ \theta(\alpha_{\mathcal{A}})$$

since θ is an adjunction. Since $\alpha_{\mathcal{B}} = \phi \circ \alpha_{\mathcal{A}}$, we have $\theta(\alpha_{\mathcal{B}}) = U'(\phi) \circ \theta(\alpha_{\mathcal{A}})$.

Theorem 7.2.1. *The function U is a functor from $\mathfrak{A}$ to $\mathfrak{D}$.*

Proof. We have given the action of U on objects and arrows of $\mathfrak{A}$. It remains to show that

$$U(1_{\mathcal{A}}) = 1_{U(\mathcal{A})},$$

$$U(\phi' \circ \phi) = U(\phi') \circ U(\phi).$$

This is the case because U is the restriction of the functor U' to K-linear maps which are also morphisms of K-linear automata. $\qquad\square$

The following theorem follows easily from the definitions.

Theorem 7.2.2. *Let $\mathcal{A}$ be a K-linear automaton. Then $\theta(\rho_{\mathcal{A}}) = \rho_{U(\mathcal{A})}$.*

7.3 Free K-linear Automata

We now define a functor $F : \mathfrak{D} \to \mathfrak{A}$. This functor is used implicitly when using matrices to encode a deterministic automaton with finitely many states.

Given a deterministic automaton $D = (S, I, \alpha, \triangleleft, \Omega, U'(K))$, the free K-linear automaton $F(D)$ is

$$(F'(S), K(I), F'(\alpha), \triangleleft', \theta^{-1}(\Omega))$$

where $\triangleleft'$ is defined as follows. For each $i \in I$, apply F' to the function

$$s \mapsto s \triangleleft i.$$

This yields a map from I to $\mathrm{End}^r(F'(S))$, which has a unique extension to a K-algebra map $K(I) \to \mathrm{End}^r(F'(S))$.

Let $D = (S, I, \alpha_D, \triangleleft_D, \Omega_D, U'(K))$ and $E = (T, I, \alpha_E, \triangleleft_E, \Omega_E, U'(K))$ be deterministic automata and f a morphism $D \to E$. Define $F(f) = F'(f)$; we must show that $F'(f) : F'(S) \to F'(T)$ is a morphism of K-linear automata $F(D) \to F(E)$. Dual to the determinizing case, it is easy to see that $F'(f)$ satisfies the necessary diagrams for the transition and input functions. We must show that

$$\theta^{-1}(\Omega_D) = \theta^{-1}(\Omega_E) \circ F'(f).$$

This follows from the equations $\theta^{-1}(\Omega_E \circ f) = \theta^{-1}(\Omega_E) \circ F'(f)$ and $\Omega_E \circ f = \Omega_D$.

Theorem 7.3.1. *The function F defined above is a functor from $\mathfrak{D}$ to $\mathfrak{A}$.*

Proof. Similar to the proof of Theorem 7.2.1. $\qquad\qquad\qquad\qquad\qquad\square$

7.4   Adjunctions Between Categories of Automata

We now show that the functors F and U defined above are related by an adjunction. Let $D = (S, I, \alpha_D, \triangleleft_D, \Omega_D, U'(K))$ be a deterministic automaton and $\mathcal{A} = (M, K(I), \alpha_\mathcal{A}, \triangleleft_\mathcal{A}, \Omega_\mathcal{A})$ a K-linear automaton. We must find a bijection

$$\psi : \mathfrak{A}(F(D), A) \to \mathfrak{D}(D, U(A))$$

such that the defining conditions of an adjunction are satisfied. We claim that the desired ψ is a restriction of the adjunction between $K-\mathbf{Mod}$ and $\mathbf{Set}$.

Lemma 7.4.1. *Let $D = (S, I, \alpha_D, \triangleleft_D, \Omega_D, U'(K))$ be a deterministic automaton, $\mathcal{A} = (M, K(I), \alpha_\mathcal{A}, \triangleleft_\mathcal{A}, \Omega_\mathcal{A})$ a K-linear automaton, and ϕ a morphism of K-linear automata $F(D) \to \mathcal{A}$. Then*

$$\psi(\phi) = \phi|_S : D \to U(\mathcal{A})$$

is a morphism of deterministic automata $D \to U(\mathcal{A})$.

Proof. By definition of F and U and the fact that ϕ is a K-linear automaton morphism, the following diagrams commute:

$$
\begin{array}{ccc}
F'(\star) \xrightarrow{F'(\alpha_D)} F'(S) & F'(S) \xrightarrow{\lhd'_D} F'(S) & F'(S) \xrightarrow{\theta^{-1}(\Omega_D)} K \\
\quad_{\alpha_\mathcal{A}} \searrow \quad \downarrow_{\phi} & \phi \downarrow \qquad \downarrow_{\phi} & \phi \downarrow \quad \nearrow_{\Omega_\mathcal{A}} \\
M & M \xrightarrow[\lhd_\mathcal{A}]{} M & M.
\end{array}
$$

To show that $\psi(f)$ is a morphism of deterministic automata, we must show the commutativity of

$$
\begin{array}{ccc}
\star \xrightarrow{\alpha_D} S & S \xrightarrow{\lhd_D} S & S \xrightarrow{\Omega_D} U'(K) \\
\quad_{\theta(\alpha_\mathcal{A})} \searrow \quad \downarrow_{\psi(\phi)} & \psi(\phi) \downarrow \qquad \downarrow_{\psi(\phi)} & \psi(\phi) \downarrow \quad \nearrow_{U'(\Omega_\mathcal{A})} \\
U'(M) & U'(M) \xrightarrow[\lhd'_\mathcal{A}]{} U'(M) & U'(M).
\end{array}
$$

This follows readily from the definitions of the functions involved. □

Note that $\psi(\phi) = \theta(\phi)$, when ϕ is considered as a K-linear map.

Lemma 7.4.2. *Let $D = (S, I, \alpha_D, \lhd_D, \Omega_D, U'(K))$ be a deterministic automaton, $\mathcal{A} = (M, K(I), \alpha_\mathcal{A}, \lhd_\mathcal{A}, \Omega_\mathcal{A})$ a K-linear automaton, and f a morphism of deterministic automata $D \to U(\mathcal{A})$. Then*

$$
\psi^{-1}(f) = F(D) \to \mathcal{A},
$$

the K-linear extension of f, is a morphism of K-linear automata $F(D) \to \mathcal{A}$.

Proof. Let $\phi = \psi^{-1}(f)$. As in Lemma 7.4.1, it is easy to see that the commutativity of

$$
\begin{array}{ccc}
\star \xrightarrow{\alpha_D} S & S \xrightarrow{\lhd_D} S & S \xrightarrow{\Omega_D} U'(K) \\
\quad_{\theta(\alpha_\mathcal{A})} \searrow \quad \downarrow_{f} & f \downarrow \qquad \downarrow_{f} & f \downarrow \quad \nearrow_{U'(\Omega_\mathcal{A})} \\
U'(M) & U'(M) \xrightarrow[\lhd'_\mathcal{A}]{} U'(M) & U'(M)
\end{array}
$$

implies the commutativity of

$$
\begin{array}{ccc}
F'(\star) \xrightarrow{F'(\alpha_D)} F'(S) & F'(S) \xrightarrow{\lhd'_D} F'(S) & F'(S) \xrightarrow{\theta^{-1}(\Omega_D)} K \\
\ \ \searrow^{\alpha_\mathcal{A}} \quad \downarrow^{\phi} & \downarrow^{\phi} \qquad \downarrow^{\phi} & \downarrow^{\phi} \ \ \nearrow^{\Omega_\mathcal{A}} \\
M & M \xrightarrow{\lhd_\mathcal{A}} M & M.
\end{array}
$$

$\square$

Theorem 7.4.1. *The function ψ is an adjunction from $\mathfrak{D}$ to $\mathfrak{A}$.*

Proof. Lemmas 7.4.1 and 7.4.2 imply that ψ is a bijection between $\mathfrak{A}(F(D), A)$ and $\mathfrak{D}(D, U(A))$. Furthermore, ψ is the restriction of the adjunction between $K-$**Mod** and **Set** to K-linear maps which are also morphisms of K-linear automata. For all arrows $k : \mathcal{A} \to \mathcal{A}'$ in $\mathfrak{A}$ and $h : D' \to D$ in $\mathfrak{D}$, we have $Uk = U'k$ and $Fh = F'h$. Therefore

$$\psi(\phi \circ Fh) = \psi\phi \circ h,$$

$$\psi(k \circ \phi) = Uk \circ \psi\phi$$

for all arrows $\phi : F(D) \to A$. $\square$

7.5 Completeness

By Theorem 6.1.1, morphisms of K-linear automata preserve the language accepted. This can be thought of as a soundness proof for a proof system for equivalence of K-linear automata in which a proof consists of a sequence of K-linear automata and morphisms between them. We now show that given any two equivalent K-linear automata $\mathcal{A}$ and $\mathcal{B}$, we can find such a sequence. In other words, the aforementioned proof system is complete.

Theorem 7.5.1. *Let $\mathcal{A}$ be a K-linear automaton. We have the following sequence of K-linear automata and morphisms:*

$$\mathcal{A} \xleftarrow{\ \epsilon\ } F(U(\mathcal{A})) \xleftarrow{\ F(i)\ } F(U(\mathcal{A})') \xrightarrow{\ F(m)\ } F(M(U(\mathcal{A})'))$$

Proof. The morphism from $F(U(\mathcal{A}))$ to $\mathcal{A}$ is the counit of the adjunction ψ between $\mathfrak{A}$ and $\mathfrak{D}$ (Theorem 7.4.1). The remainder of the sequence follows from Theorem 2.3.2 and the functor F. The deterministic automaton $U(\mathcal{A})'$ is the accessible subautomaton of $U(\mathcal{A})$ and i is the inclusion of $U(\mathcal{A})'$ into $U(\mathcal{A})$. The morphism of deterministic automata m is the morphism from $U(\mathcal{A})'$ to $M(U(\mathcal{A})')$, the minimization of $U(\mathcal{A})'$. $\square$

Remark 7.5.1. The above sequence can be shortened since $\epsilon \circ F(i)$ is a morphism from $F(U(\mathcal{A})')$ to $\mathcal{A}$.

Corollary 7.5.1. *Let $\mathcal{A}$ and $\mathcal{B}$ be equivalent K-linear automata. There is a sequence of K-linear automata and morphisms which witnesses the equivalence.*

Proof. By Theorem 7.2.2, $U(\mathcal{A})$ and $U(\mathcal{B})$ are equivalent deterministic automata, therefore have the same minimization. Applying Theorem 7.5.1 to $\mathcal{A}$ and $\mathcal{B}$ yields sequences with the same endpoint. $\square$

Remark 7.5.2. Theorem 7.5.1 also holds for weighted automata over an arbitrary semiring S, with some slight modifications. In the weighted case, we do not have an input K-algebra. Instead, to each letter of a finite alphabet Σ, there corresponds an S-linear map $M \rightarrow M$, where M is a free (right or left) S-semimodule on a finite set. In this case, the determinization and free functors do not have to change the type of the inputs; both weighted automata and deterministic automata take elements of Σ^* as inputs.

In the next chapter, we discuss the complexity of constructing such witnessing sequences. We also use these sequences as the basis for a proof system for the equational theory of Kleene algebra.

CHAPTER 8

COMPLEXITY AND KLEENE ALGEBRA

We now apply the results from Chapter 7 to show that a $PSPACE$ transducer can produce a sequence witnessing the equivalence of two equivalent K-linear automata. This sequence is verifiable in polynomial time. Here we require K to be the two-element idempotent semiring, the underlying K-semimodules of the K-linear automata to be free K-semimodules on finite sets, and the input K-algebra to be the free K-algebra on a finite set of letters. As usual, we restrict our attention to right automata.

We use these witnessing sequences as inspiration for a proof system for the equational theory of *Kleene algebra*. The proof system essentially consists of the logic of equational implications augmented with the axioms of Kleene algebra. The proof system utilizes the fact that finite automata can be encoded as triples of terms of the free Kleene algebra on a finite set.

8.1 Kleene Algebra

A Kleene algebra (KA) is a structure $\mathcal{K} = (\mathcal{K}, 0, 1, +, \cdot, ^*)$ such that $(\mathcal{K}, 0, 1, +, \cdot)$ is an idempotent semiring which also satisfies the following laws:

$$1 + a^*a \leq a^* \qquad\qquad 1 + aa^* \leq a^*$$

$$b + ax \leq x \Rightarrow a^*b \leq x \qquad b + xa \leq x \Rightarrow ba^* \leq x.$$

The partial order $\leq$ is induced by addition, i.e.,

$$x \leq y \text{ if and only if } x + y = y.$$

Example 8.1.1. Let Σ be a finite alphabet and R be the set of regular languages over Σ. Then R is a Kleene algebra with 0 interpreted as the empty language, 1 as the language $\{\lambda\}$, $+$ as union, $\cdot$ as concatenation, and $*$ as the Kleene asterate. This is the free Kleene algebra on Σ.

Example 8.1.2. Let X be a set. The set of all binary relations on X is a Kleene algebra. Here 0 denotes the empty relation and 1 the identity relation. The operations are given as follows: $+$ is union of relations, $\cdot$ is composition of relations, and $*$ is the reflexive transitive closure of a relation.

Remark 8.1.1. Kleene algebra has many applications in theoretical computer science, in particular a strengthening known as *Kleene algebra with tests* (KAT). For example, Propositional Hoare Logic can be encoded in the equational theory of KAT [17], and KAT can be applied to verify the correctness of certain compiler optimizations [18]. The equational theory of KAT reduces to the equational theory of KA in the following sense: an equation $t_1 = t_2$ in the language of KAT can be transformed into an equation $t_1' = t_2'$ in the language of KA such that $t_1 = t_2$ is a theorem of KAT if $t_1' = t_2'$ is a theorem of KA [19].

An important fact is that the set of $n \times n$ matrices over a Kleene algebra has a natural Kleene algebra structure. Addition and multiplication correspond to standard addition and multiplication of matrices. The star operator is defined inductively. See [14] for details.

At several points below, we must reason about non-square matrices over a KA. We would like to know whether the theorems of Kleene algebra hold when the primitive letters are interpreted as matrices of arbitrary dimension and the function symbols are interpreted polymorphically. In general, this is not the case. However, there is a large class of theorems which do survive this

treatment, including all theorems used below. This is shown in [16].

8.2 Automata, Actions, and KA Terms

Matrices over a Kleene algebra are useful because they allow finite automata to be encoded as Kleene algebra terms [14]. We recall this encoding, and then relate it to the notion of a K-linear automaton.

Definition 8.2.1. A *Kleene algebra automaton* over a Kleene algebra $\mathcal{K}$ is a triple (u, A, v) where u is an $1 \times n$ $(0, 1)$-vector, v is an $n \times 1$ $(0, 1)$-vector, and A is an $n \times n$ matrix over $\mathcal{K}$. The vector u encodes the start states of (u, A, v) and is called the *start vector*. The vector v encodes the accept states of (u, A, v) and is called the *accept vector*. The matrix A is called the *transition matrix*.

Definition 8.2.2. The *language accepted* by (u, A, v) is the element uA^*v.

Definition 8.2.3. The *size* of (u, A, v), denoted $|(u, A, v)|$, is the number of states of the automaton. That is, if A is an $n \times n$ matrix, then $|(u, A, v)| = n$.

Note that an automaton over a Kleene algebra can have a transition matrix whose entries are arbitrary terms of a Kleene algebra. To relate Kleene algebra automata to K-linear automata, we first restrict the complexity of the terms appearing in the transition matrices. Moreover, we assume that the Kleene algebra in question is the free Kleene algebra on a finite set Σ. We denote this Kleene algebra by F_Σ.

Definition 8.2.4. Let (u, A, v) be an automaton over $\mathcal{F}_\Sigma$. We say that (u, A, v) is

(a) *simple* if A can be expressed as a sum

$$A = J + \sum_{a \in \Sigma} a \cdot A_a$$

where J and each A_a is a $(0,1)$-matrix.

(b) *λ-free* if J is the zero matrix.

(c) *deterministic* if it is simple, λ-free, and u and all rows of each A_a have exactly one 1.

Unsurprisingly, we are interested in Kleene algebra automata that accept the same language.

Definition 8.2.5. Let (u, A, v) and (s, B, t) be two Kleene algebra automata. We say that (u, A, v) and (s, B, t) are *equivalent* precisely when $uA^*v = sB^*t$.

The following theorem, called the *bisimulation rule*, can be used to prove Kleene algebra automata equivalent.

Theorem 8.2.1. [14] *Let $\mathcal{K}$ be a Kleene algebra and $a, b, x \in \mathcal{K}$. The following equational implication is a theorem of Kleene algebra:*

$$xa = bx \quad \Rightarrow \quad xa^* = b^*x.$$

Theorem 8.2.2. [14] *Let (u, A, v) be a Kleene algebra automaton of size n and let (s, B, t) be a Kleene algebra automaton of size m. Let X be an $m \times n$ matrix. Suppose that*

$$sX = u \tag{8.1}$$

$$XA = BX \tag{8.2}$$

$$Xv = t. \tag{8.3}$$

Then $uA^*v = sB^*t$.

Proof. Essentially the same as the proof of Theorem 2.1.1:

$$sB^*t = sB^*Xv = sXA^*v = uA^*v.$$

$\square$

Remark 8.2.1. The matrix X in the statement of the theorem is called a *disimulation matrix* from (s, B, t) to (u, A, v) in [29]. Note that we write state vectors as row vectors, so such an X takes state vectors of (s, B, t) to state vectors of (u, A, v) via right multiplication.

8.2.1 KA Automata to K-linear Automata

We now encode λ-*free* Kleene algebra automata as K-linear automata. Let (u, A, v) be a λ-free Kleene algebra automaton over F_Σ. Let $n = |(u, A, v)|$ and let M be the free K-semimodule on the set $\{s_1, s_2, \dots, s_n\}$. Then (u, A, v) corresponds to the K-linear automaton

$$(M, K\Sigma^*, u, \triangleleft, \Omega),$$

where $\triangleleft$ is defined by

$$\triangleleft : M \otimes K\Sigma^* \to M$$

$$m \triangleleft a = m \cdot A_a$$

for all $a \in \Sigma$ and extended as an action to all of $K\Sigma^*$ and $K\Sigma^*$ is the set of all finite K-linear sums of elements of Σ^*, as in Example 5.1.2. We write elements of M as row vectors and right-multiply by the A_a's to effect a transition.

The observation function Ω is given by

$$\Omega : M \to K$$

$$\Omega(m) = m \cdot v.$$

Theorem 8.2.3. *Let (u, A, v) be a λ-free Kleene algebra automaton and $(M, K\Sigma^*, u, \triangleleft, \Omega)$ be the corresponding K-linear automaton. For any $w \in \Sigma^*$,*

$$w \leq uA^*v \quad \text{if and only if} \quad \Omega(u \triangleleft w) = 1.$$

Proof. This is straightforward: since (u, A, v) is λ-free, $w \leq uA^*v$ precisely when there is a w-labelled path from a start state of (u, A, v) to an accept state. $\qquad\square$

Certain disimulation matrices correspond to morphisms of K-linear automata.

Theorem 8.2.4. *Let (u, A, v) and (s, B, t) be λ-free Kleene algebra automata over the Kleene algebra F_Σ. Let $\mathcal{A}$ be the K-linear automaton corresponding to (u, A, v) and $\mathcal{B}$ be the K-linear automaton corresponding to (s, B, t). Let X be a disimulation matrix from (s, B, t) to (u, A, v) whose entries come from the set $\{0, 1\}$. Then X encodes a morphism of K-linear automata from $\mathcal{B}$ to $\mathcal{A}$.*

Proof. Since it's a matrix, X determines a K-linear map from the underlying K-semimodule of $\mathcal{B}$ to the underlying K-semimodule of $\mathcal{A}$ via right multiplication. Note that the states of (u, A, v) are the basis elements for the underlying K-semimodule of $\mathcal{A}$ and the states of (s, B, t) are the basis elements for the underlying K-semimodule of $\mathcal{B}$. Since X satisfies Equations 8.1, 8.2, and 8.3, it also satisfies Equations 6.1, 6.2, and 6.3, so X determines a morphism of K-linear automata. This can be shown with the bisimulation lemma: Equation 8.2 show that the map determined by X commutes with each a-transition for $a \in \Sigma$. The bisimulation lemma allows us to conclude that X commutes with the extended w-transition for each $w \in \Sigma^*$. Then X commutes with the actions of each element of $K\Sigma^*$ by K-linearity. $\qquad\square$

8.2.2 K-linear Automata to KA Automata

Let $\mathcal{A} = (M, K\Sigma^*, s, \lhd, \Omega)$ be a K-linear automaton, where M the free K-semimodule on the set $\{m_1, m_2, \ldots, m_n\}$. We now give a λ-free Kleene algebra automaton (u, A, v) over $\mathcal{F}_\Sigma$ corresponding to $\mathcal{A}$.

The start vector u is $s \in M$ expressed in the basis $\{m_1, m_2, \ldots, m_n\}$ as a row vector. The accept vector is the $n \times 1$ $(0, 1)$-vector representing the map $M \to K$ using the bases $\{m_1, m_2, \ldots, m_n\}$ and $\{1\}$.

For a given $a \in \Sigma$, the A_a component of the transition matrix A is the matrix of the K-linear map $M \to M$ given by the right action of a considered as an element of $K\Sigma^*$. Again, $\{m_1, m_2, \ldots, m_n\}$ is the basis used. Note that each entry of X is either a 0 or a 1. A simple argument shows the following:

Theorem 8.2.5. *For any $w \in \Sigma^*$,*

$$\Omega(s \lhd w) = 1 \text{ if and only if } w \leq uA^*v.$$

Finally, we have a theorem relating morphisms of K-linear automata to disimulation matrices. The proof is straightforward.

Theorem 8.2.6. *Let $\mathcal{A}$ and $\mathcal{B}$ be K-linear automata, and let (u, A, v) and (s, B, t) be the corresponding λ-free K-linear automata. Let ϕ be a morphism of K-linear automata from $\mathcal{B}$ to $\mathcal{A}$. Then the matrix of ϕ with respect to the appropriate bases is a disimulation matrix from (s, B, t) to (u, A, v).*

8.3 The Proof System

The logic of the proof system is the logic of equational implications, enriched with the axioms of Kleene algebra. We also add as axioms a few theorems of Kleene algebra concerning automata. These theorems are from [14]. For each theorem, the hypotheses will be easy to verify in the proofs we produce — they will essentially be star-free equations involving 0's and 1's — so the hypotheses will be verifiable in polynomial time.

The first three theorems listed below are used to construct an automaton accepting the language denoted by a given KA term (Kleene's Theorem). The theorems algebraically represent constructions on automata. Let γ and δ be terms over some $\mathcal{F}_\Sigma$. Let (u, A, v) be an automaton accepting γ, (s, B, t) be an automaton accepting δ, and Φ be a sequence of equations or equational implications.

The first theorem is known as the *union lemma*. It is used when producing a Kleene algebra automaton accepting the union of the languages of two given Kleene algebra automata.

$$\frac{\Phi \vdash uA^*v = \gamma \qquad \Phi \vdash sB^*t = \delta}{\Phi \vdash \begin{bmatrix} u & | & s \end{bmatrix} \begin{bmatrix} A & 0 \\ \hline 0 & B \end{bmatrix}^* \begin{bmatrix} v \\ \hline t \end{bmatrix} = \gamma + \delta} .$$

The second theorem is known as the *concatenation lemma*. The term vs in the upper right corner of the transition matrix in the conclusion represents adding

98

λ-transitions from the accept states of (u, A, v) to the start states of (s, B, t):

$$\frac{\Phi \vdash uA^*v = \gamma \qquad \Phi \vdash sB^*t = \delta}{\Phi \vdash \begin{bmatrix} u & | & 0 \end{bmatrix} \begin{bmatrix} A & vs \\ \hline 0 & B \end{bmatrix}^* \begin{bmatrix} 0 \\ \hline t \end{bmatrix} = \gamma\delta}.$$

The third is known as the *asterate lemma*. The term $A + vu$ represents adding λ-transitions from the accept states of (u, A, v) back to the start states; we must also add a state to accept the empty word:

$$\frac{\Phi \vdash uA^*v = \gamma}{\Phi \vdash \begin{bmatrix} 1 & | & u \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \hline 0 & A + vu \end{bmatrix}^* \begin{bmatrix} 1 \\ \hline v \end{bmatrix} = \gamma^*}.$$

The fourth theorem we add allows us to prove that an automaton accepts the same language as the automaton obtained by eliminating λ-transitions. Let (u, A, v) and (u', F, v) be automata of size n, and let J be an $n \times n$ matrix. Suppose that the following equations hold:

$$A = J + A'$$

$$F = A'J^*$$

$$u' = uJ^*.$$

It follows that (u, A, v) and (u', F, v) are equivalent. We add the following theorem to the KA axioms, called the λ-*elimination* lemma:

$$\frac{\Phi \vdash A = J + A' \qquad \Phi \vdash F = A'J^* \qquad \Phi \vdash u' = uJ^*}{\Phi \vdash uA^*v = u'F^*v}.$$

In our applications, (u, A, v) is simple and J is a $(0, 1)$-matrix, so uJ^* is a $(0, 1)$-vector and F is λ-free.

Finally, we add a rule expressing that two automata related by a disimulation matrix are equivalent:

$$\frac{\Phi \vdash sX = u \qquad \Phi \vdash XA = BX \qquad \Phi \vdash Xv = t}{\Phi \vdash uA^*v = sB^*t.}$$

8.4 Generating Proofs

In this section, we give an algorithm to generate proofs and show that it can be implemented by a $PSPACE$ transducer. We use the theorems of Section 8.2 frequently. These theorems allow us to reason about Kleene algebra automata using theorems about K-linear automata. Note that all of the automata appearing in the proof are Kleene algebra automata.

Given a KA term α, let $|\alpha|$ be the number of nodes in the syntax tree of α.

Theorem 8.4.1. *Let $\alpha = \beta$ be an equation of Kleene algebra. A proof that $\alpha = \beta$ can be produced by a transducer using only polynomially many (in $|\alpha| + |\beta|$) worktape cells.*

Note that this was not the case for the proof of [14]. Respecting the space bound is nontrivial; we require several terms of exponential size, some of which are constructed from terms which are themselves exponentially large. To simplify proving that the space bound is respected, we divide the construction of the proof into stages. For each stage, we show that both the terms and the proofs required at that stage can be constructed in $PSPACE$. The stages:

1. Construct an automaton accepting α, an automaton accepting β, and proofs thereof.

2. For each automaton, construct an equivalent λ-free automaton, and an equivalence proof.

3. For each λ-free automaton, construct an equivalent accessible deterministic automaton, and a disimulation matrix between them.

4. Construct the minimal deterministic automaton equivalent to the accessible deterministic automaton accepting α, and a disimulation matrix between them.

5. Construct the disimulation matrix between the minimal deterministic automaton for α and the accessible automaton for β.

Stages 2 through 5 require one or more terms from previous stages. We treat each stage independently, and show that there are transducers which generate the required terms and proofs at each stage. To combine all of the stages, we use the following fact about the composition of space-bounded transducers.

Lemma 8.4.1. *Suppose $f(x)$ can be computed by a $PSPACE$ transducer F and $g(x)$ can be computed by a $PLSPACE$ transducer G (a transducer using polylog many worktape cells in the size of its input). Then $g(f(x))$ can be computed by a $PSPACE$ transducer.*

Proof. Note that $|f(x)|$ might be exponential in $|x|$, so there is not necessarily enough space to write down $f(x)$ in its entirety. Rather, a $PSPACE$ transducer H computing $g(f(x))$ computes $f(x)$ on a demand-driven basis. On input x, H begins by running G. Whenever a bit of $f(x)$ is needed, H saves the current state of G and begins running F on input x, disregarding the output of F until the required bit of $f(x)$ is produced. It then resumes running G, supplying the requested bit of $f(x)$. The transducer H needs polynomially many

worktape cells to run F, polynomially many cells to count up to the length of $f(x)$, and polynomially many cells for G's worktape, since G needs at most $O((\log|f(x)|)^d) \le O(|x|^m)$ for some m. $\qquad\square$

8.4.1 Stage 1: KA term to Automaton

We first show that the inductive construction used in the proof of Kleene's theorem can be performed by a $PSPACE$ machine. Given a term α, the machine must construct an automaton (u, A, v) accepting α and a proof that $uA^*v = \alpha$.

Given $a \in \Sigma$, the following automaton accepts the language $\{a\}$:

$$\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right).$$

There are also one-state automata for $\emptyset$ and ϵ: $([0], [0], [0])$ and $([1], [1], [1])$, respectively. We assume that for every $a \in \Sigma$, the machine has a proof that

$$a = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix}^* \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

stored in its finite control. We also assume that the machine can output proofs of the equations

$$0 = 00^*0$$

$$1 = 11^*1.$$

For the inductive step, the machine can work its way up the syntax tree of α, constructing automata using the union, concatenation, and asterate lemmas. At each step, it outputs the appropriate equation, i.e., the conclusion of one of the

three lemmas. When finished, the machine will have constructed an automaton accepting α and also will have printed a proof of this fact on the output tape. All of the terms appearing in the proof are polynomial in the size of α and straight-forward to construct. Furthermore, the automata constructed are simple.

8.4.2 Stage 2: Automaton to λ-free Automaton

We now show that there is a transducer which takes a simple automaton (u, A, v) as input and constructs from it an equivalent simple λ-free automaton (u', F, v), and that there is a transducer which takes as input the pair $((u, A, v), (u', F, v))$ and outputs a proof of the equivalence.

Constructing the λ-free automaton, (u', F, v), is easy. Since (u, A, v) is simple,

$$A = J + \sum_{a \in \Sigma} a \cdot A_a.$$

as in Definition 8.2.4.a. The transducer computes J from (u, A, v) and then com-putes J^*, which is just the reflexive transitive closure of the relation denoted by J. It also computes

$$A' = \sum_{a \in \Sigma} a \cdot A_a.$$

Then

$$u' = uJ^*$$

$$F = A'J^*.$$

It is easy to see that both u' and F can be constructed in $PSPACE$. Note that (u', F, v) might not be simple, but can easily be made so using additive idempotence. To prove equivalence, the proof-generating transducer uses the

ϵ-elimination lemma. It must prove the following hypotheses:

$$A = J + A'$$

$$F = A'J^*$$

$$u' = uJ^*$$

all of which are easily proved in $PSPACE$. The machine must also prove that the term J^* is the star of J. First, the machine proves

$$1 + J(1 + J + J^2 + \cdots + J^n) \leq (1 + J + J^2 + \cdots + J^n)$$

by direct computation, where $n = |(u, A, v)|$. This inequality is true; if the i, j entry of JJ^n is 1, then there is a path of length $n + 1$ from i to j (viewing J as the adjacency matrix of a graph). Since J has only n vertices, this path must repeat at least one vertex, and so there will be a 1 in the i, j entry of J^k for some $k < n + 1$. Reasoning in KA,

$$J^* \leq 1 + J + J^2 + \cdots + J^n.$$

Next, the machine generates a proof that for any x,

$$1 + x + x^2 + \cdots + x^n \leq x^*.$$

This inequality is an easy consequence of the KA axioms. Substituting J for X and combining these two inequalities yields

$$1 + J + J^2 + \cdots + J^n = J^*.$$

8.4.3   Stage 3: λ-free Automaton to Deterministic Automaton

It must now be shown that there is a $PSPACE$ transducer which takes in (u', F, v), a simple λ-free automaton, and outputs (s, D, t), an equivalent accessi-

ble deterministic automaton. Let $|(u', F, v)| = n$. The Kleene algebra automaton $\mathcal{A}$ corresponds to a K-linear automaton $\mathcal{A}$ by Theorem 8.2.3.

By Theorem 7.5.1, we have the following sequence of K-linear automata:

$$\mathcal{A} \xleftarrow{\ \epsilon\ } F(U(\mathcal{A})) \xleftarrow{\quad F(i)\quad} F(U(\mathcal{A})').$$

Recall that ϵ is the counit of the adjunction between $K-\mathbf{Mod}$ and $\mathbf{Set}$, $U(\mathcal{A})'$ is the accessible part of $U(\mathcal{A})$, and i is the inclusion map $U(\mathcal{A})' \to U(\mathcal{A})$. So we must simply construct (s, D, t), the Kleene algebra automaton corresponding to $U(\mathcal{A})'$, and a $(0, 1)$-matrix X encoding the K-linear map $F(i) \circ \epsilon$.

To generate (s, D, t), the machine performs the standard subset construction on (u', F, v), with the added condition that it tests each subset for accessibility before granting it state status. The subset construction forgets the module structure on the underlying K-semimodule of $\mathcal{A}$. Elements of a free K-semimodule are essentially subsets of generators when K is the two-element idempotent semiring. Encoding the subsets as entries of a matrix is an implicit application of the free K-semimodule functor.

The following lemma verifies that the test for accessibility can be performed in $PSPACE$.

Lemma 8.4.2. *Let (u', F, v) be a simple λ-free automaton with n states. It is decidable in $O(n^2)$ space whether C, a set of states of (u, F, v), is accessible when considered as a state in the deterministic automaton obtained from (u', F, v) by the subset construction.*

Proof. We first give a nondeterministic linear space machine. The machine starts with (u', F, v) and the characteristic vector of C written on its input tape. It begins by writing the start vector u' on its worktape. If $u' = C$, it halts and answers

105

yes. Otherwise it guesses an $a \in \Sigma$ and overwrites its worktape contents with the characteristic vector of $u \triangleleft_F a$. If this is equal to C, it accepts, otherwise it guesses another letter and repeats. At any time, the machine must store only $O(n)$ bits of information. By Savitch's theorem, there is an equivalent deterministic machine running in $O(n^2)$ space.   □

To construct s, the machine counts from 0 to $2^n - 1$ in binary (each number is identified with a subset of states of (u', F, v) by treating its binary representation as a characteristic vector). For each i between 0 and $2^n - 1$, it tests whether i represents an accessible state. If i does not, the machine proceeds to the next i. If i does represent an accessible state, the machine outputs 1 if i represents precisely the set of start states of (u', F, v), and 0 otherwise. The construction of t is similar, except the machine outputs 1 if any state in the subset represented by i is an accept state, or 0 if none are.

The construction of D, the transition matrix, requires three counters. The first two, i and j, range from 0 to $2^n - 1$, and are used to keep track of the rows and columns of D, respectively. The third counter, c, ranges from 0 to $m - 1$, where $m = |\Sigma|$. The machine starts with all counters set to zero. It begins by testing i for accessibility. If i is inaccessible, it increments i and repeats. If i does correspond to an accessible state, it then tests each possible value of j for accessibility. If j is not accessible, it increments j. If j does represent an accessible state, it tests each $a_k \in \Sigma$ to determine whether $i \triangleleft_F a_k = j$ (cf. Theorem 7.2.1). If yes, it outputs a_k. If none of the a_k tests succeed, it outputs 0. After testing all of the a_i's, the machine resets c to 0 and goes to the next j. After checking all of the j's, the machine resets j to 0 and goes to the next i. It is easy to see that (s, D, t) is a deterministic Kleene algebra automaton.

This transducer runs in $O(n^2)$ space, where n is $|(u', F, v)|$. The machine requires $O(n^2)$ space to perform the test in Lemma 8.4.2 and a few counters which range up to $2^n - 1$.

Let d be $|(s, D, t)|$ and let X be the $d \times n$ matrix encoding the relation in which a state of (s, D, t) is related to all of the states of (u, F, v) that it "contains". In other words, X encodes that map $F(i) \circ \epsilon$ (by right multiplication).

We must show that X can be computed without violating the space bound. The transducer which takes the pair $((u', F, v), (s, D, t))$ and outputs X can use only polynomially many (in $|(u', F, v)|$) cells, although $|(s, D, t)|$ may be exponential in n. To construct X, the machine needs one counter ranging from 0 to $2^n - 1$. For each i between 0 and $2^n - 1$, the machine tests the subset of states encoded by i for accessibility. If it is accessible, it outputs the binary representation of i as a row vector. If i does not represent an accessible state, it goes to $i + 1$.

8.4.4   Stage 4: Deterministic Automaton to Minimal Deterministic Automaton

At this stage, we require two transducers. The first constructs the minimal deterministic automaton equivalent to a given accessible deterministic automaton, and the second takes as input a pair (deterministic automaton, equivalent minimal deterministic automaton) and outputs the disimulation matrix between them. By Theorem 7.5.1, we have the following K-linear automata and mor-

phisms:

$$F(U(\mathcal{A})') \xrightarrow{\quad F(m) \quad} F(M(U(\mathcal{A})')).$$

Recall that $M(U(\mathcal{A})')$ is the minimal deterministic automaton equivalent to $U(\mathcal{A})'$. We must construct (p, M, q), the Kleene algebra automata corresponding to $M(U(\mathcal{A})')$, and a $(0, 1)$-matrix representing the map $F(m)$.

The automaton (p, M, q) is constructed by examining (s, D, t) and outputting the least-numbered state in each equivalence class of a Myhill-Nerode relation. This guarantees that each state (generator) of (p, M, q) corresponds to a formal language. We require a lemma establishing a space bound on the procedure to identify equivalent states.

Lemma 8.4.3. *Let (s, D, t) be a deterministic Kleene algebra automaton. It is decidable in $PLSPACE$ space whether i and j, two states of (s, D, t), are equivalent.*

Proof. We first give an $NLOGSPACE$ procedure to recognize distinguishable (inequivalent) states. The machine begins with (s, D, t), i, and j written on its input tape. If one of i, j is an accept state and the other is not, the machine halts and answers distinguishable. Otherwise it guesses an $a_1 \in \Sigma$ and overwrites its worktape contents with $i \triangleleft_D a_1$ and $j \triangleleft_D a_1$. If exactly one of these states is an accept state, the machine halts and answers distinguishable. If not, it guesses an $a_2 \in \Sigma$ and repeats the procedure. At any time, the machine has to remember only two states of (s, D, t), and so it runs in $NLOGSPACE$. By Savitch's theorem, there is an equivalent deterministic machine running in $O((\log |(s, D, t)|)^2)$ space. □

To construct p, the start vector, the machine scans s. For each state i, it checks whether i is equivalent to some lower-numbered state. If yes, it skips to the next

108

i. If i is the least-numbered state in its equivalence class, the machine outputs a 1 if i is equivalent to the start state of (s, D, t), and 0 otherwise. The accept vector, q, is constructed similarly. The machine scans through t, and for each state i that is the least-numbered state in its equivalence class, it outputs 1 if i is an accept state, 0 if i is not.

The construction of the transition matrix M resembles the construction of the transition matrix of the deterministic automaton in the previous stage. The machine maintains two counters, i and j. It scans through the states of (s, D, t), and for each state i which is the least-numbered state in its equivalence class, it tests each state j in turn, outputting D_{ij} for each j which is the first state in its equivalence class. It is easy to see that this procedure can be done in $PLSPACE$ and does indeed generate the equivalent minimal automaton.

A transducer to construct the disimulation matrix X in $PLSPACE$ from the pair $((s, D, t), (p, M, q))$ uses a straightforward modification of Lemma 8.4.3. It must now check whether states in different automata are equivalent. By Lemma 8.4.1, the above terms can be generated in $PSPACE$.

8.4.5 Stage 5: Deterministic Automaton for β Disimilar to Minimal Automaton for α

It suffices to use the procedure from the previous stage to generate the disimulation matrix between the two automata.

ALTERNATING AUTOMATA

We now consider alternating finite automata, which have the following interesting property.

Theorem 9.0.2. *(Theorem 2.14 [25]) Let L be a formal language. Then L is accepted by an n-state alternating finite automaton if and only if the reverse of L is accepted by a deterministic finite automaton with at most 2^n states.*

The reason for the reversal is that alternating finite automata are essentially defined using left actions, whereas the standard definition of a deterministic finite automaton involves a right action. The main construction of this chapter is a determinizing functor for alternating finite automata.

In this chapter, K always refers to the two-element idempotent semiring. Also, we denote the empty word by e, because we use λ-notation for functions.

9.1 Alternating Automata

We recall the definition of an alternating finite automaton.

Definition 9.1.1. ([15]) An *alternating finite automaton* $A = (Q, \Sigma, \delta, F, \beta)$ consists of:

1. A finite set of *states* Q,

2. A finite *input alphabet* Σ,

3. A *transition function*

$$\delta : (Q \times \Sigma) \rightarrow ((Q \rightarrow \{0,1\}) \rightarrow \{0,1\}),$$

4. A set of *final states* $F \subseteq Q$,

5. An *acceptance condition* $\beta : (Q \rightarrow \{0,1\}) \rightarrow \{0,1\}$.

The transition function δ determines a function

$$\hat{\delta} : (Q \times \Sigma^*) \rightarrow ((Q \rightarrow \{0,1\}) \rightarrow \{0,1\})$$

defined inductively as

$$\hat{\delta}(q,e)(u) = u(q)$$

$$\hat{\delta}(q,aw)(u) = \delta(q,a)(\lambda p.(\hat{\delta}(p,w)(u)))$$

where $q \in Q, a \in \Sigma, w \in \Sigma^*$, and $u \in (Q \rightarrow \{0,1\})$.

Definition 9.1.2. An AFA $A = (Q, \Sigma, \delta, F, \beta)$ is said to *accept* $w \in \Sigma^*$ precisely when

$$\beta(\lambda p.(\hat{\delta}(p,w)(F))) = 1.$$

Intuitively, the machine produces a computation tree with depth equal to the length of the input word. The characteristic vector of F provides a $\{0,1\}$-labelling of the leaves of this tree. The transition function δ is used to inductively label the lower levels of the tree. An input word is accepted if the induced labelling at level 0 satisfies β.

We now cast the definition of an alternating finite automaton into our framework. Intuitively, in an AFA, the states are described using a K-semimodule structure. However, the transitions do not necessarily respect the K-semimodule structure: the transitions are arbitrary functions from the K-semimodule to itself. Note than an AFA reads its input from right to left.

111

Definition 9.1.3. A *alternating finite K-automaton* $A = (M, \Sigma^*, s, \triangleright, \Omega)$ consists of:

1. A *semimodule of states*, M, which is the free K-semimodule on a finite set Q,

2. An *input monoid* Σ^*, which is the free monoid on a finite alphabet Σ,

3. A *start vector* $s \in M$,

4. A *left action* $\triangleright : \Sigma^* \times M \to M$, which is a monoid homomorphism from Σ^* to $\mathrm{End}^l(U'(M))$, where $U'(M)$ is the underlying set of M,

5. An *observation function* $\Omega : M \to K$.

Definition 9.1.4. Let $A = (M, \Sigma^*, s, \triangleright, \Omega)$ be an alternating finite K-automaton and $w \in \Sigma^*$. Then A is said to *accept* w precisely when

$$\Omega(w \triangleright s) = 1.$$

Let $A = (Q, \Sigma, \delta, F, \beta)$ be an alternating finite automaton. An alternating finite K-automaton equivalent to A is $B = (M, \Sigma^*, s, \triangleright, \Omega)$, where M is the free K-semimodule on Q, s is the characteristic function of F, and $\Omega = \beta$. The transition action $\triangleright$ is defined as follows. Let $u \in M$. We consider u to be a function $Q \to K$. For each $a \in \Sigma$, let

$$f_a : M \to M$$

$$f_a(u)(q) = \delta(q, a)(u).$$

We then extend this to a monoid homomorphism $\Sigma^* \to \mathrm{End}^l(U'(M))$ and denote the image of w under this homomorphism by f_w. We must show $f_w(u) = \lambda p.(\hat{\delta}(p, w))(u)$. For the base case,

$$f_e(u) = u = \lambda p.(\hat{\delta}(p, e))(u).$$

112

Now suppose that

$$f_w(u) = \lambda p.(\hat{\delta}(p, w))(u)$$

Then

$$f_{aw}(u) = f_a(f_w(u))$$

$$= f_a(\lambda p.(\hat{\delta}(p, w))(u))$$

$$= \lambda p.(\hat{\delta}(p, aw))(u).$$

This establishes the following theorem.

Theorem 9.1.1. *Let* $A = (Q, \Sigma, \delta, F, \beta)$ *be an alternating finite automaton and* $B = (M, \Sigma^*, s, \triangleright, \Omega)$ *be the corresponding alternating finite K-automaton constructed according to the procedure above. Then*

$$\Omega_B(w \triangleright_B s_B) = 1 \text{ if and only if } \beta_A(\lambda p.(\hat{\delta}_A(p, w)(F_A))) = 1$$

for all $w \in \Sigma^*$.

9.2 Determinization

Determinizing an alternating finite K-automaton is similar to the determinization of K-linear automata in Section 7.2. Let K$-$**Mod$'$** be the category whose objects are K-semimodules and whose arrows are arbitrary functions between K-semimodules. Let G' be the functor K$-$**Mod$'$** $\to$ **Set** which takes an object of K$-$**Mod$'$** to its underlying set and an arrow of K$-$**Mod$'$** to the function between the underlying sets.

Let $B = (M, \Sigma^*, s, \triangleright, \Omega)$ be an alternating finite K-automaton. We can easily modify the construction in Section 7.2 to yield a functor G from the category of

113

alternating finite K-automata to a category of deterministic automata. Applying G to B yields a *left* deterministic automaton $D = (G'(M), \Sigma^*, s', G'(\triangleright), G'(\Omega))$. Here s' is just s considered as an element of $G'(M)$. Note that $|D| = 2^n$, where n is the number of generators of the underlying K-semimodule of B. If B is the alternating finite K-automaton corresponding to some AFA A, then the start state of D corresponds to the final states of A. The observation function Ω_D satisfies $\Omega_D(s) = 1$ if and only if s corresponds to a subset of states of A which satisfies the acceptance condition.

In the literature, it is common to define deterministic automata as right deterministic automata. However, the deterministic automaton produced, D, is a left deterministic automaton. However, D can be made into a nondeterministic left finite automaton N by coding D with $(0, 1)$-matrices, i.e., applying a free functor.

This left automaton can also be viewed as a right automaton N', similar to Example 6.1.1. The reverse of N' (cf. Theorem 3.3.1) is deterministic in the sense that it has only one start state and only one outgoing a-transition at each state for each $a \in \Sigma$.

In the next chapter, we provide some remarks on different ways to use a K-semimodule structure to describe the states of an automaton efficiently.

CHAPTER 10

CONCLUSION AND FUTURE WORK

We now provide a summary, concluding remarks, and some directions for future investigation.

10.1 Summary and Concluding Remarks

Let M be a monoid in a monoidal category. We developed a view of an automaton as a pointed, observable representation of M. In the monoidal category **Set**, this yields deterministic automata with elements of M as inputs. Such automata define functions (languages) $M \to O$, where O is an arbitrary nonempty set. If M is a free monoid on a finite set and O is a two-element set, this reduces to the usual definition of a deterministic automaton, with possibly infinitely many states.

In the monoidal category $K-\mathbf{Mod}$, this yields K-linear automata, which provide an alternative to weighted automata. In this alternative, formal languages are generalized to elements of a dual semimodule, rather than being generalized to formal power series. This approach highlights the numerous analogies between the theory of algebras (bialgebras) and the theory of automata and formal languages: transitioning from one to another is essentially changing the underlying category. Furthermore, we have shown the utility of comultiplication for performing constructions on automata.

We gave an adjunction between certain categories of K-linear automata and certain categories of deterministic automata. One functor of the adjunction, a forgetful functor, generalizes the subset construction used to determinize non-

deterministic automata. Using this adjunction, we gave a categorical account of the completeness proof in [14].

Finally, we fit alternating finite automata into our framework. We showed that alternating finite automata naturally employ left actions. The standard definition of a deterministic automaton uses a right action, and so the usual subset construction for alternating finite automata— for example in [15] or [25] — must turn a left action into a right action. This explains why the usual subset construction does not yield a deterministic automaton, but rather an automaton whose reverse is deterministic.

It is instructive to compare how states are represented in deterministic, K-linear, and alternating finite automata. In the case of deterministic automata, the states are elements of unstructured sets. In contrast, alternating finite automata have a K-semimodule structure on their states. However, the transitions of an alternating finite automaton are arbitrary functions and are not required to respect this structure. With K-linear automata, there is a K-semimodule structure on the set of states, and the transitions are K-linear maps. When the K-semimodule of states is a free K-semimodule, the transitions can be completely described by their actions on the generators. In this case, the K-semimodule structure is essentially an efficient description of the states of the automaton, and this description can be used to give an efficient description of the transitions. Determinization is simply forgetting this description.

10.2   Future Work

We hope to extend the results in this dissertation in the following directions.

116

10.2.1 Bialgebras and Hopf Algebras

One possibility is to look for deeper connections between K-bialgebras and automata. For example, the *Tannaka-Krein* theorem, loosely stated, allows one to reconstruct a bialgebra from its category of representations. This theorem is generalized to other bialgebra-like structures in [24]. In the context of automata, we hope to extend this theorem to recover the input K-bialgebra from a monoidal category of K-linear automata. It seems likely that with appropriate assumptions, multiplication in the input K-bialgebra could be recovered from the transitions of the automata, and comultiplication in the input K-bialgebra could be recovered from the monoidal structure of the category of automata. In Chapter 6, we saw that comultiplication in a bialgebra defines a multiplication of automata. Essentially, we would like to investigate when the converse holds.

We would also like to find connections between automata and *Hopf algebras*. A Hopf algebra H is a bialgebra along with an *antipode* map $H \rightarrow H$, which plays a role similar to an inverse. It does not seem likely that a map satisfying the defining diagrams of an antipode will be of use when working with automata; doing so would require some sort of "inversion" of the input word. However, a standard result is that an antipode is both an anti-algebra map and an anti-coalgebra map (see, for example, Proposition 1.3.1 of [24]). In certain cases, such as the K-bialgebras in Example 5.3.1, word reversal induces an anti-algebra map and an anti-coalgebra map. It is our hope that a theory of "weak Hopf algebras", in which the antipode is replaced by an arbitrary map which is both an anti-algebra map and an anti-coalgebra map could be developed to give a comprehensive account relating reversal, duality, and minimization.

10.2.2 Proof Complexity

We would also like to investigate the complexity of the proof system in Chapter 7. As we have seen, a special case of this system yields a proof system for the equivalence of nondeterministic finite automata (encoded as K-linear automata). The proofs produced can be exponentially long, because they rely on (accessible) determinization. However, determinization is not always necessary. For example, given two isomorphic K-linear automata, an isomorphism between them can be encoded as a K-linear map. A widely-held belief of complexity theory, $NP \neq PSPACE$, implies that there must be equivalent K-linear automata such that any proof in this system is exponentially long (here K is the two-element idempotent semiring). Identifying equations which are hard to prove is interesting from the point of view of proof complexity, and identifying equations with short proofs might be of use for applications of Kleene algebra.

10.2.3 Other Automata and Other Inputs

Finally, we would like to apply ideas from automata theory to other disciplines which use bialgebras, such as combinatorics or physics. Of particular interest would be to find natural situations involving automata with non-free input monoids. In particular, bialgebras of trees have been studied by combinatorialists for some time. This may yield a new type of tree automaton.

BIBLIOGRAPHY

[1] J. Adàmek and V. Trnkovà. *Automata and Algebras in Categories*. Kluwer Academic Publishers, 1990.

[2] J. Berstel and C. Reutenauer. *Rational Series and Their Languages*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1988.

[3] P. Buchholz. Bisimulation relations for weighted automata. *Theoretical Computer Science*, 393:109–123, 2008.

[4] S.A. Cook and A.R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.

[5] G. Duchamp, M. Flouret, È Laugerotte, and J.-G. Luque. Direct and dual laws for automata with multiplicities. *Theoretical Computer Science*, 267:105–120, 2001.

[6] G. Duchamp and Christophe Tollu. Sweedler's duals and Schützenberger's calculus. Archive Preprint: arXiv:0712.0125v2.

[7] David S. Dummit and Richard M. Foote. *Abstract Algebra*. Prentice-Hall, Inc, 1999.

[8] Samuel Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, 1974.

[9] Melvin Fitting. Bisimulations and boolean vectors. *Advances in Modal Logic*, 4:97–125, 2003.

[10] Jonathan S. Golan. *Semirings and Their Applications*. Kluwer Academic Publishers, 1999.

[11] R.L. Grossman and R.G. Larson. The realization of input-output maps using bialgebras. *Forum Mathematicum*, 4:109–121, 1992.

[12] R.L. Grossman and R.G. Larson. Bialgebras and realizations. In J. Bergen, S. Catoiu, and W. Chin, editors, *Hopf Algebras*, pages 157–166. Marcel Dekker, Inc, 2004.

[13] Yefim Katsov. Tensor products and injective envelopes of semimodules over additively regular semirings. *Algebra Colloquium*, 4(2):121–131, 1997.

[14] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Infor. and Comput.*, 110(2):366–390, May 1994.

[15] Dexter Kozen. *Automata and Computability*. Springer-Verlag, 1997.

[16] Dexter Kozen. Typed Kleene algebra. Technical Report TR98-1669, Computer Science Department, Cornell University, March 1998.

[17] Dexter Kozen. On Hoare logic and Kleene algebra with tests. *Trans. Computational Logic*, 1(1):60–76, July 2000.

[18] Dexter Kozen and Maria-Cristina Patron. Certification of compiler optimizations using Kleene algebra with tests. In John Lloyd, Veronica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luis Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *Proc. 1st Int. Conf. Computational Logic (CL2000)*, volume 1861 of *Lecture Notes in Artificial Intelligence*, pages 568–582, London, July 2000. Springer-Verlag.

[19] Dexter Kozen and Frederick Smith. Kleene algebra with tests: Completeness and decidability. In D. van Dalen and M. Bezem, editors, *Proc. 10th Int. Workshop Computer Science Logic (CSL'96)*, volume 1258 of *Lecture Notes in Computer Science*, pages 244–259, Utrecht, The Netherlands, September 1996. Springer-Verlag.

[20] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1986.

[21] Serge Lang. *Algebra: Revised Third Edition*. Springer-Verlag, 2002.

[22] G.L. Litvinov, V.P. Masloc, and G.B. Shpiz. Tensor products of idempotent semimodules. An algebraic approach. *Mathematical Notes*, 65(4), 1999.

[23] Saunders MacLane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

[24] Shahn Majid. *Foundations of Quantum Group Theory*. Cambridge University Press, 1995.

[25] G. Rosenberg and A. Salomaa, editors. *Handbook Of Formal Languages*, volume 1. Springer-Verlag, 1997.

[26] J.J.M.M. Rutten. Automata and coinduction (an exercise in coalgebra). In *Proc. CONCUR '98*, volume 1466 of *LNCS*, pages 194–218. Springer-Verlag, 1998.

[27] J.J.M.M. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[28] Ross Street. *Quantum Groups: A Path To Current Algebra*. Cambridge University Press, 2007.

[29] James Worthington. Automatic proof generation in Kleene algebra. In *Proc. 10th Int. Conf. Relational Methods in Computer Science (RelMiCS10) and 5th Int. Conf Applications of Kleene Algebra (AKA5)*, volume 4988 of *LNCS*, pages 382–396. Springer-Verlag, April 2008.

[30] James Worthington. A bialgebraic approach to automata and formal language theory. In *Proc. Logical Foundations of Computer Science (LFCS '09)*, volume 5407 of *LNCS*, pages 451–467. Springer-Verlag, January 2009.