Introduction to Kleene Algebra

Dexter C. Kozen

Introduction to Kleene Algebra

Dexter C. Kozen

Contents

Lectures

Lecture 1

Course Roadmap

Kleene algebra (KA) is an algebraic system that captures axiomatically the properties of a natural class of structures arising in logic and computer science. It is named for Stephen Cole Kleene (1909–1994), who among his many other achievements invented finite automata and regular expressions, structures of fundamental importance in computer science. Kleene algebra is the algebraic theory of these objects, although it has many other natural and useful interpretations.

Kleene algebras arise in various guises in many contexts: relational algebra [91, 92, **?**], semantics and logics of programs [61, **?**], automata and formal language theory [78, 79], and the design and analysis of algorithms [1, 53, 63, 84]. Many authors have contributed to the development of Kleene algebra over the years: [2, 6, 7, 14, 15, 23, 29, 41, 56, 61, 62, 64, 77, 79, 102, 104, 106, 107], to name a few. There are various competing axiomatizations, and one topic of our study will be to understand the relationships between these definitions.

In semantics and logics of programs, Kleene algebra forms an essential component of Propositional Dynamic Logic (PDL) [36], in which it is mixed with Boolean algebra and modal logic to give a theoretically appealing and practical system for reasoning about computation at the propositional level. From a practical point of view, many simple program manipulations, such as loop unwinding and basic safety analysis, do not require the full power of PDL, but can be carried out in a purely equational

subsystem using the axioms of Kleene algebra. The Boolean algebra component is an essential ingredient, since it is needed to model conventional programming constructs such as conditionals and `while` loops that rely on Boolean tests. However, for many applications, the modal component is not essential. We define later on a variant of Kleene algebra, called *Kleene algebra with tests* (KAT), for reasoning equationally with these constructs. We will show that KAT provides an equational approach to program verification that subsumes traditional approaches such as Hoare logic. Very recently, KAT has been adapted to form NetKAT, a language and logic for reasoning about packet-switching networks [3, 37].

Cohen has studied Kleene algebra in the presence of extra Boolean and commutativity conditions. He has given several practical examples of the use of Kleene algebra in program verification, such as lazy caching [25] and concurrency control [26]. In addition, Kleene algebra has been used for verifying low-level compiler optimizations [74], data restructuring operations in parallelizing compilers [8, 82], pointer analysis [87, 88], and static analysis [73].

Much of the basic algebraic theory of KA was developed by John Horton Conway in his 1971 monograph [29]. This delightful little volume, originally published by Chapman and Hall, was out of print for many years; but due to significant renewed interest in the topic in recent years, it reappeared in a paperback edition by Dover in 2012.

We begin our study by describing several concrete examples of Kleene algebras. These will serve as motivating examples to provide intuition about the properties we are trying to capture axiomatically with the formal definition. We will conclude this lecture with the formal definition of a Kleene algebra and derive some basic properties that follow from these axioms.

Examples of Kleene Algebras

A Kleene algebra consists of a set K with distinguished binary operations $+$ and $\cdot$, unary operation *, and constants 0 and 1 with certain properties. The intuitive meaning of the operations depends on the model; however, we can at least say that the operator * typically involves some notion of *iteration*. The * operator is the most interesting aspect of Kleene algebra. For example, it allows us to express and reason about properties of simple looping constructs in programming languages.

Here are three classes of models that motivate the definition of Kleene algebra.

Language-Theoretic Models

Let Σ^* denote the set of finite-length strings over a finite alphabet Σ, including the null string ε. Define the following constants and operations on subsets of Σ^*:

$$A + B \stackrel{\text{def}}{=} A \cup B \tag{1.1}$$

$$A \cdot B \stackrel{\text{def}}{=} \{xy \mid x \in A,\ y \in B\} \tag{1.2}$$

$$0 \stackrel{\text{def}}{=} \varnothing \tag{1.3}$$

$$1 \stackrel{\text{def}}{=} \{\varepsilon\}. \tag{1.4}$$

Thus the operation $\cdot$, applied to two sets of strings A and B, produces the set of all strings obtained by concatenating a string from A with a string from B, in that order. The operator symbol $\cdot$ is often omitted, and we just write AB for $A \cdot B$.

These operations have several agreeable properties. For example, $\cdot$ distributes over $+$ on both sides, in the sense that $A(B + C) = AB + AC$ and $(A + B)C = AC + BC$; the element 0 is a both a left and right identity for $+$ in the sense that $0 + A = A + 0 = A$; and the element 1 is both a left and right identity for $\cdot$ in the sense that $1A = A1 = A$.

Now define the powers of A with respect to $\cdot$ inductively:

$$A^0 \stackrel{\text{def}}{=} \{\varepsilon\} \qquad\qquad A^{n+1} \stackrel{\text{def}}{=} A \cdot A^n.$$

The unary operation * on sets of strings is defined as follows:

$$A^* \stackrel{\text{def}}{=} \bigcup_{n \geq 0} A^n = \{x_1 \cdots x_n \mid n \geq 0 \text{ and } x_i \in A,\ 1 \leq i \leq n\}. \tag{1.5}$$

Thus A^* is the union of all powers of A; equivalently, A^* consists of all strings obtained by concatenating together any finite collection of strings from A in any order. By convention, the concatenation of the empty set of strings is ε; this is the case $n = 0$ in (1.5). Thus ε is always a member of A^* for any A, including $\varnothing$. The operation * is known as *Kleene asterate*.

Any subset of the full powerset of Σ^* containing $\varnothing$ and $\{\varepsilon\}$ and closed under the operations of $\cup$, $\cdot$, and * is a Kleene algebra, and is a subalgebra of the full powerset algebra. One such subalgebra of particular significance is the algebra of *regular sets*. This is the smallest subalgebra containing all sets $\{a\}$ for $a \in \Sigma$.

As is well known, the regular sets are also the sets of strings accepted by finite-state automata, or finite-state transition systems with an acceptance condition. The equivalence of these two representations was proved in Kleene's original paper [56] and is known in this context as *Kleene's theorem*. A proof of this result can be found in any introductory text in automata and computability; see for example [50, 67].

Relational Algebras

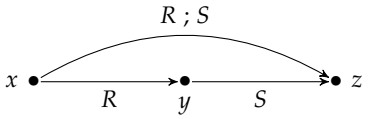
Another useful interpretation involves binary relations on a set X. Recall that a *binary relation* on a set X is just a set of ordered pairs of elements of X. Thus a binary relation on X is a subset of

$$X \times X = \{(x, y) \mid x, y \in X\}.$$

The set of all binary relations on a set X forms a Kleene algebra under the following definitions of the operators. We again interpret $+$ as set union. The multiplication operation $\cdot$ is interpreted as *relational composition*

$$R \,;\, S \stackrel{\text{def}}{=} \{(x, z) \mid \exists y \in X \; (x, y) \in R \text{ and } (y, z) \in S\}.$$

(A common alternative notation is $S \circ R$. Note that with this notation, the order of R and S is reversed. This is for consistency with the usage of the same notation for functional composition $(g \circ f)(x) = g(f(x))$.) If we view R as a set of labeled directed edges on X, then there is an edge from x to z labeled $R \,;\, S$ iff there is a node y, an edge from x to y labeled R, and an edge from y to z labeled S.



The element 0 is the null relation $\varnothing$, and 1 is the identity relation

$$1 \stackrel{\text{def}}{=} \{(x, x) \mid x \in X\}.$$

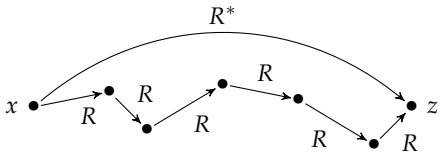
The operation * gives the *reflexive transitive closure* of a relation. Recall that a relation R is *reflexive* if $(x, x) \in R$ for all $x \in X$; that is, if R includes the identity relation as a subset. The relation R is *transitive* if $(x, z) \in R$ whenever both $(x, y) \in R$ and $(y, z) \in R$; in other words, R is transitive if $R \,;\, R \subseteq R$. The smallest reflexive and transitive relation containing R is called the *reflexive transitive closure* of R and is denoted R^*. This notation fortuitously coincides with the definition of the * operation as giving the sum of all finite powers of R as above.

$$R^* = \bigcup_{n \geq 0} R^n,$$

where

$$R^0 \stackrel{\text{def}}{=} \{(x, x) \mid x \in X\} \qquad\qquad R^{n+1} \stackrel{\text{def}}{=} R \,;\, R^n.$$

Equivalently, there is an R^* edge from x to z iff there is an R-path of length 0 or more from x to z.



A *relational Kleene algebra* is any subset of $2^{X \times X}$ closed under these operations. These models are useful in programming language semantics, because they can be used to represent the input/output relations of programs.

The $(\min, +)$ Algebra

Here is a rather unusual model that turns out to be useful in shortest path algorithms in graphs. This algebra is called the $(\min, +)$ *algebra*, also known as the *tropical semiring*. The domain is the set $\mathbb{R}_+ \cup \{\infty\}$ of nonnegative reals with an additional infinite element ∞. The Kleene algebra operation $+$ is interpreted as the operation min giving the minimum of two elements in the natural order on $\mathbb{R}_+ \cup \{\infty\}$. The Kleene algebra operation $\cdot$ is interpreted as $+$ in $\mathbb{R}_+ \cup \{\infty\}$; the usual definition of $+$ on $\mathbb{R}_+$ is extended to include ∞ in the natural way:

$$x + \infty \;=\; \infty + x \;=\; \infty + \infty \;=\; \infty.$$

The Kleene algebra constants 0 and 1 are interpreted as ∞ and the real number 0, respectively.

The * operation on this algebra is not very interesting: $x^* = 1$ (= the real number 0) for any x. However, the * of matrices over this algebra is quite interesting: it gives a way of calculating the shortest path between any two points in a finite directed graph.

The $(\max, \cdot)$ Algebra

The domain of this algebra is the unit interval $[0, 1]$. The Kleene algebra operation $+$ is interpreted as max giving the maximum of two elements in the natural order on $[0, 1]$. The operation $\cdot$ is interpreted as ordinary multiplication. The Kleene algebra constants 0 and 1 are the real numbers 0 and 1, respectively. This algebra is useful in error-correcting codes and hidden Markov models using an algorithm known as the *Viterbi algorithm*.

The $(\max, \cdot)$ algebra and the $(\min, +)$ algebra are isomorphic under the map $x \mapsto -\log x$.

Lecture 2

Axioms of Kleene Algebra

Now we give the formal definition of a Kleene algebra and derive some basic consequences.

Semigroups and Monoids

A *semigroup* is an algebraic structure $(S, \cdot)$, where S is a set and $\cdot$ is an associative binary operation on S, which means that $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ for all $x, y, z \in S$. This allows us to write $x \cdot y \cdot z$ without ambiguity. A *monoid* is an algebraic structure $(M, \cdot, 1)$ where $(M, \cdot)$ is a semigroup and 1 is a distinguished element of M that is both a left and right identity for $\cdot$ in the sense that $1 \cdot x = x \cdot 1 = x$ for all $x \in M$.

For semigroups and monoids written multiplicatively, we often omit the operator $\cdot$ in expressions, writing xy for $x \cdot y$.

Example 2.1 *The following are common examples of monoids:*

1. $(\Sigma^*, \cdot, \varepsilon)$, *where* Σ^* *is the set of finite-length strings over an alphabet* Σ, $\cdot$ *is concatenation of strings, and* ε *is the null string;*
2. $(2^{\Sigma^*}, \cdot, \{\varepsilon\})$, *where* $\cdot$ *is set concatenation as described in the last lecture;*
3. $(2^{\Sigma^*}, \cup, \varnothing)$, *where* 2^{Σ^*} *is the powerset or set of all subsets of* Σ^*, $\cup$ *is set union, and* $\varnothing$ *is the empty set;*

4. $(\mathbb{N},\ +,\ 0)$, *where* $\mathbb{N}$ *is the set of natural numbers* $\{0, 1, 2, \ldots\}$;
5. $(\mathbb{N},\ \cdot,\ 1)$;
6. $(\mathbb{N}^n,\ +,\ \overline{0})$, *where* $\mathbb{N}^n$ *is the Cartesian product of* n *copies of* $\mathbb{N}$, $+$ *is vector addition, and* $\overline{0}$ *is the zero vector;*
7. $(\mathbb{R}_+ \cup \{\infty\},\ \min,\ \infty)$, *where* $\mathbb{R}_+$ *denotes the set of nonnegative real numbers,* ∞ *is a special infinite element greater than all real numbers, and* min *gives the minimum of two elements;*
8. $(R^{n\times n},\ \cdot,\ I)$, *where* $R^{n\times n}$ *denotes the set of* $n \times n$ *matrices over a ring* R, $\cdot$ *is ordinary matrix multiplication, and* I *is the identity matrix;*
9. $(X \to X,\ \circ,\ \mathsf{id})$, *where* $X \to X$ *denotes the set of all functions from a set* X *to itself,* $\circ$ *is function composition, and* id *is the identity function.*

Examples 3–7 are *commutative* monoids, which means that $xy = yx$ for all x, y. Example 8 is never commutative for any nontrivial ring R and $n \geq 2$. Example 9 is never commutative for any X with at least 2 elements.

Idempotent Semirings

A *semiring* is an algebraic structure $(S,\ +,\ \cdot,\ 0,\ 1)$ such that

- $(S,\ +,\ 0)$ is a commutative monoid,
- $(S,\ \cdot,\ 1)$ is a monoid,
- $\cdot$ distributes over $+$ on both the left and right in the sense that $x(y + z) = xy + xz$ and $(x + y)z = xz + yz$,
- 0 is an *annihilator* for $\cdot$ in the sense that $0 \cdot x = x \cdot 0 = 0$ for all x.

A semiring is *idempotent* if $x + x = x$ for all x. We often abbreviate $x \cdot y$ to xy, and avoid parentheses by taking $\cdot$ to be higher precedence than $+$.

Collecting these axioms, we define an *idempotent semiring* to be any structure $(S,\ +,\ \cdot,\ 0,\ 1)$ satisfying the following identities for all $x, y, z \in S$:

$$
\begin{array}{ll}
x + (y + z) = (x + y) + z & x + y = y + x \\
x + 0 = x & x + x = x \\
x(yz) = (xy)z & 1x = x1 = x \\
x(y + z) = xy + xz & (x + y)z = xz + yz \\
0x = x0 = 0. &
\end{array}
$$

A *ring* is a semiring in which the additive monoid forms a group; that is, in which additive inverses exist. We cannot have additive inverses in an idempotent semiring unless the semiring is trivial, since $0 = -x + x = -x + x + x = 0 + x = x$.

Order

Recall that a *partial order* is a binary relation on a set that is

- reflexive: for all x, $x \leq x$,
- antisymmetric: for all x, y, if $x \leq y$ and $y \leq x$, then $x = y$, and
- transitive: for all x, y, z, if $x \leq y$ and $y \leq z$, then $x \leq z$.

Any idempotent semiring has a naturally-defined partial order $\leq$ associated with it:

$$x \leq y \stackrel{\text{def}}{\iff} x + y = y. \tag{2.1}$$

The order relation $\leq$ is so central to the theory that one might take it as primitive, but we will consider it an abbreviation for the equation on the right-hand side of (2.1).

In the language-theoretic and relational models of the last lecture, $\leq$ is set inclusion $\subseteq$. But beware: in the $(\min, +)$ algebra, $\leq$ is the *reverse* of the natural order on $\mathbb{R}$ extended to $\mathbb{R}_+ \cup \{\infty\}$. By (2.1), $x \leq y$ iff $\min x, y = y$, but this occurs iff x is greater than or equal to y in the natural order on $\mathbb{R}$. (Note that we are carefully avoiding the use of the notation $\leq$ for the natural order on $\mathbb{R}$!)

That $\leq$ is a partial order follow easily from the definition (2.1) and the axioms of idempotent semirings. Reflexivity is just the idempotence axiom $x + x = x$. Antisymmetry follows from commutativity of $+$: if $x \leq y$ and $y \leq x$, then $y = x + y = y + x = x$. Finally, for transitivity, if $x \leq y$ and $y \leq z$, then

$$\begin{aligned} x + z &= x + (y + z) && \text{since } y + z = z \\ &= (x + y) + z && \text{associativity of } + \\ &= y + z && \text{since } x + y = y \\ &= z && \text{again since } y + z = z, \end{aligned}$$

therefore $x \leq z$.

In any idempotent semiring, the operators $+$ and $\cdot$ are *monotone* with respect to $\leq$ in the sense that for all $x, y, z \in S$,

$$\begin{aligned} &x \leq y \Rightarrow x + z \leq y + z \\ &x \leq y \Rightarrow z + x \leq z + y \\ &x \leq y \Rightarrow xz \leq yz \\ &x \leq y \Rightarrow zx \leq zy. \end{aligned}$$

(Monotonicity will also hold for the * operator of Kleene algebra, that is, $x \leq y \Rightarrow x^* \leq y^*$, when we get around to discussing it.) These properties follow easily from the axioms.

In any idempotent semiring, the operator $+$ gives the *least upper bound* or *supremum* of any pair of elements with respect to the natural order $\leq$, and 0 is the least element of the semiring with respect to $\leq$. To say that $x + y$ is the *least upper bound* of x and y says that

- $x + y$ is an upper bound: $x \leq x + y$ and $y \leq x + y$;
- $x + y$ is the *least* upper bound: if z is any other upper bound, that is, if $x \leq z$ and $y \leq z$, then $x + y \leq z$.

The proof that $x + y$ is an upper bound uses associativity and idempotence: $x + (x + y) = (x + x) + y = x + y$, and similarly for $y \leq x + y$. The proof that it is the least upper bound uses only associativity: if $x \leq z$ and $y \leq z$, then $(x + y) + z = x + (y + z) = x + z = z$.

One observation that is not difficult to check is that the $n \times n$ matrices over a semiring again form a semiring under the natural definitions of the matrix operations. Moreover, if the underlying semiring is idempotent, then so is the matrix semiring (Exercise **??**).

The * Operator

Now we turn to the * operator. This is the most interesting part of Kleene algebra, because it captures the notion of *iteration*. Because of this, it may seem that * is inherently infinitary. Indeed, there are several infinitary axiomatizations that we will consider. However, it is possible to derive most of the interesting parts of the theory in a purely finitary way.

The * operator is a unary operator written in postfix. Intuitively, x^* represents zero or more iterations of x. In relational models, this is reflexive transitive closure; in language models, the Kleene asterate.

There are several different competing axiomatizations of *, and in part our study will be to understand the relationships among them. For now, we shall pick a particular one as our official definition for the purposes of this course. Thus we define a *Kleene algebra* to be a structure $(K, +, \cdot, ^*, 0, 1)$ that is an idempotent semiring under $+$, $\cdot$, 0, 1 satisfying the properties (2.2)–(2.5) below. We assign precedence $^* > \cdot > +$ to the operators to avoid unnecessary parentheses.

The axioms for * consist of two equations and two equational implications or *Horn formulas*. (Note that up to now, the axioms have been purely equational.) The two equational axioms for * are

$$1 + xx^* \leq x^* \tag{2.2}$$

$$1 + x^*x \leq x^* \tag{2.3}$$

and the two equational implications are

$$b + ax \leq x \Rightarrow a^*b \leq x \tag{2.4}$$
$$b + xa \leq x \Rightarrow ba^* \leq x. \tag{2.5}$$

Of course, these are all considered to be implicitly universally quantified, so that (2.4) and (2.5) are assumed to hold for all a, b, and x in any Kleene algebra.

The significance of (2.2)–(2.5) concerns the solution of linear inequalities. As we shall see, much of the theory of Kleene algebra is concerned with the solution of finite systems of linear inequalities. For example, a finite automaton is essentially such a system. Axioms (2.2) and (2.4) provide for the existence of a unique least solution to a certain single linear inequality in a single variable, namely

$$b + aX \leq X, \tag{2.6}$$

where X is a variable ranging over elements of the Kleene algebra. Axioms (2.2) and (2.4) together essentially say that a^*b is a solution to (2.6), and moreover, it is the unique least solution among all solutions in the Kleene algebra. First, (2.2) says that a^*b is a solution, since by monotonicity of multiplication and distributivity,

$$\begin{aligned} 1 + aa^* \leq a^* &\Rightarrow (1 + aa^*)b \leq a^*b \\ &\Rightarrow b + a(a^*b) \leq a^*b\ ; \end{aligned}$$

and (2.4) says exactly that a^*b is less than or equal to any other solution, therefore it is the unique least solution. Dually, the axioms (2.3) and (2.5) say that ba^* is the unique least solution to $b + Xa \leq X$.

Let us illustrate the use of (2.4) and (2.5) to show that (2.2) and (2.3) can be strengthened to equalities. We show this for (2.2); the result for (2.3) is symmetric. We already have that $1 + xx^* \leq x^*$, so by antisymmetry, it suffices to show the reverse inequality; equivalently,

$$x^*1 \leq 1 + xx^*.$$

This is the right-hand side of (2.4) with 1 substituted for b, x substituted for a, and $1 + xx^*$ substituted for x, so it suffices to show that the left-hand side of (2.4) holds under the same substitution, or

$$1 + x(1 + xx^*) \leq 1 + xx^*.$$

But this is immediate from (2.2) and monotonicity.

Recall from the last lecture that in relational models, R^* was defined to be the reflexive transitive closure of the relation R. To be reflexive means

that $\mathsf{id} \subseteq R^*$, where id is the identity relation; to be transitive means that $R^* \; ; \; R^* \subseteq R^*$; and to contain R means that $R \subseteq R^*$. Abstractly, these properties are expressed by the inequalities

$$1 \leq x^* \tag{2.7}$$
$$x^* x^* \leq x^* \tag{2.8}$$
$$x \leq x^*, \tag{2.9}$$

respectively. Equivalently,

$$1 + x^* x^* + x \leq x^*. \tag{2.10}$$

We might interpret this inequality as saying that x^* is a reflexive and transitive element dominating x. It does not, however, say that it is the reflexive transitive closure of x; for that we need the equational implication

$$1 + yy + x \leq y \Rightarrow x^* \leq y, \tag{2.11}$$

which says that x^* is the *least* reflexive and transitive element dominating x.

Now in the presence of the other axioms, (2.10) is equivalent to (2.2) (and, by symmetry, to (2.3) as well). To prove that (2.2) implies (2.10), it suffices to show that (2.2) implies (2.7)–(2.9). The inequality (2.7) is immediate from (2.2). Also, multiplying (2.7) on the left by x, by monotonicity we have $x \leq xx^*$; then (2.9) is immediate from this and (2.2). The last inequality (2.8) is left as an exercise (Homework 1, Exercise 1(a)). This argument requires either (2.4) or (2.5).

Conversely, to show that (2.10) implies (2.2), assume (2.10). Then $1 \leq x^*$ from (2.7). Also, by (2.8), (2.9), and monotonicity we have $xx^* \leq x^* x^* \leq x^*$. Since $1 + xx^*$ is the least upper bound of 1 and xx^*, we have (2.2).

Each of (2.4) and (2.5) alone implies (2.11). The converse does not hold: later, we will construct a "left-handed" Kleene algebra that is not "right-handed" (one satisfying (2.4) but not (2.5)). To show that (2.4) implies (2.11), suppose that (2.4) holds for all a, b, and x, and assume the left-hand side of (2.11). To show the right-hand side of (2.11) holds, by (2.4) it suffices to show that $1 + xy \leq y$. But this follows easily from the left-hand side of (2.11): we have $x \leq y$, and by monotonicity, $1 + xy \leq 1 + yy \leq y$.

In the presence of the other axioms, the implications (2.4) or (2.5) are equivalent to

$$ax \leq x \Rightarrow a^* x \leq x \tag{2.12}$$
$$xa \leq x \Rightarrow xa^* \leq x, \tag{2.13}$$

respectively. These alternative forms are quite useful in some contexts. The proofs of equivalence are left as an exercise (Exercise **??**).

Finally, as promised, we show that * is monotone. Suppose $x \leq y$. We wish to show that $x^* \leq y^*$. By (2.4), it suffices to show that $1 + xy^* \leq y^*$. But since $x \leq y$, by monotonicity and (2.2) we have $1 + xy^* \leq 1 + yy^* \leq y^*$.

Next time we will look at some alternative axiomatizations of *.

Lecture 3

Alternative Axiomatizations

There has been some dissent regarding the proper axiomatization of Kleene algebra. Many inequivalent axiomatizations have been proposed [29, 61, 64, 100, **?**], all serving roughly the same purpose. It is important to understand the relationships between these classes in order to extract the axiomatic essence of Kleene algebra. In this lecture we present some of these alternative axiomatizations and discuss some of the relationships among them.

Recall that our official definition is that a Kleene algebra is an idempotent semiring satisfying

$$1 + xx^* \leq x^* \tag{3.1}$$

$$1 + x^*x \leq x^* \tag{3.2}$$

$$b + ax \;\leq\; x \Rightarrow a^*b \;\leq\; x \tag{3.3}$$

$$b + xa \;\leq\; x \Rightarrow ba^* \;\leq\; x. \tag{3.4}$$

Star-Continuity

A Kleene algebra is called *star-continuous* (or sometimes *star-complete*) if it satisfies the axiom

$$xy^*z = \sup_{n \geq 0} xy^nz, \tag{3.5}$$

where $y^0 = 1$, $y^{n+1} = yy^n$, and sup refers to the *supremum* or *least upper bound* with respect to the natural order $\leq$. This property says that any possibly infinite set of the form $\{xy^nz \mid n \geq 0\}$ has a least upper bound, and that least upper bound is xy^*z. The property (3.5) is called *star-continuity*. Star-continuous Kleene algebras have been used to model programs in Dynamic Logic [61].

Every star-continuous idempotent semiring is a Kleene algebra, since one can easily show that in any idempotent semiring, the star-continuity condition (3.5) implies the axioms (3.1)–(3.4) of Kleene algebra. However, as we shall see later, not every Kleene algebra is star-continuous, although all naturally occurring ones are.

The property (3.5) is actually an infinitary condition. It is equivalent to infinitely many inequalities

$$xy^nz \leq xy^*z, \; n \geq 0, \tag{3.6}$$

which together say that xy^*z is an upper bound for all xy^nz, $n \geq 0$, along with the infinitary Horn formula

$$(\bigwedge_{n\geq 0} xy^nz \leq w) \Rightarrow xy^*z \leq w, \tag{3.7}$$

which says that it is the least such upper bound.

Another way to view (3.5) is as a combination of the statement that y^* is the supremum of y^n, $n \geq 0$, along with two infinitary distributivity properties, one on the left and one on the right.

To show that every star-continuous idempotent semiring is a Kleene algebra, we first show that (3.1) holds.

$$1 + xx^* = 1 + \sup_n xx^n = x^0 + \sup_n x^{n+1} = \sup_n x^n = x^*.$$

The general property we have used in the third step is that if A and B are any subsets of an upper semilattice such that $\sup A$ and $\sup B$ exist, then $\sup A \cup B$ exists and is equal to $\sup A + \sup B$. The proof of (3.2) is symmetric.

To show (3.3), assume that $b + ax \leq x$. We would like to show that $a^*b \leq x$. By star-continuity, it suffices to show that for all $n \geq 0$, $a^nb \leq x$. This is easily shown by induction on n. For the basis $n = 0$, we have $a^0b = b \leq x$ from our assumption. Now assuming $a^nb \leq x$, we have $a^{n+1}b = aa^nb \leq ax$ by monotonicity, and $ax \leq x$ by our assumption. Again, the proof of (3.4) is symmetric.

Closed Semirings

In the design and analysis of algorithms, a related family of structures called *closed semirings* form an important algebraic abstraction. They give

a unified framework for deriving efficient algorithms for transitive closure and all-pairs shortest paths in graphs and constructing regular expressions from finite automata [114, 1, 84]. Very fast algorithms for all these problems can be derived as special cases of a single general algorithm over an arbitrary closed semiring. Closed semirings are defined in terms of a countable summation operator $\sum$ as well as $\cdot$, 0, and 1; the operator * is defined in terms of $\sum$. Under the operations of (finite) $+$, $\cdot$, *, 0, and 1, any closed semiring is a star-continuous Kleene algebra. In fact, in the treatment of [1, 84], the sole purpose of $\sum$ seems to be to define *. A more descriptive name for closed semirings might be *ω-complete idempotent semirings*.

Formally, a *closed semiring* is an idempotent semiring in which every countable set A has a supremum $\sum A$ with respect to the natural order $\leq$, and such that for any countable set A,

$$x \cdot (\sum A) \cdot z = \sum_{y \in A} xyz. \tag{3.8}$$

The presence of x and z in (3.8) ensure a kind of infinite distributivity property on the left and right.

In any closed semiring, one can define * by

$$x^* \stackrel{\text{def}}{=} \sum_{n \geq 0} x^n,$$

where $x^0 = 1$ and $x^{n+1} = xx^n$. By infinite distributivity,

$$xy^*z = \sum_n xy^n z,$$

thus any closed semiring is a star-continuous Kleene algebra.

The regular sets $\mathsf{Reg}\,\Sigma$ do not form a closed semiring: if A is nonregular, the countable set $\{\{x\} \mid x \in A\}$ has no supremum. However, the power set of Σ^* does form a closed semiring.

Similarly, the family of all binary relations on a set forms a closed semiring under the relational operations described in Lecture 1 and set union for $\sum$.

The definition of closed semiring given above is somewhat stronger than those found in the literature on design and analysis of algorithms [1, 84]. According to [1], a closed semiring is an idempotent semiring equipped with a summation operator $\sum$ defined on countable *sequences* (not sets) that satisfies infinitary associativity and distributivity. Infinitary idempotence and commutativity are not assumed. Also, the relation between the between finitary $+$ and infinitary $\sum$ is not explicitly mentioned in [1], but can be inferred from the use of the notation $x_0 + x_1 + x_2 + \cdots$ for the infinitary sum. The element x^* is defined to be $1 + x + x^2 + \cdots$.

Infinitary associativity is defined as follows. If $(x_n \mid n \geq 0)$ is any countable sequence of elements, then for any way of partitioning the index set $\mathbb{N}$ into intervals, the sum $\sum_i x_i$ is the same as the sum of the sums of the intervals. If an interval is finite, then its sum is computed with $+$. If an interval is infinite, then its sum is computed with $\sum$. Note that any such partition must consist either of

- infinitely many finite intervals, or
- finitely many intervals, all of which are finite except the last, which is infinite.

Infinitary distributivity says that

$$x \cdot (\sum_i y_i) \cdot z = \sum xy_iz.$$

This is not the same as (3.8), since it says nothing about suprema.

The axiomatization in [84] postulates infinitary commutativity as well. Infinitary commutativity says that for any partition of the index set (not necessarily into intervals), the sum of the sums of the partition elements is the same as the sum of the original sequence.

Infinitary idempotence says that if all $x_i = x$, then $\sum_i x_i = x$. This does not follow from the axiomatizations of [1, 84], nor does the equation $x^{**} = x^*$. It can be shown that $0^* = 1$, but not that $1^* = 1$.

To see this, consider an idempotent semiring with elements $\mathbb{N} \cup \{\infty\}$. Define finitary addition $+$ to be max in the natural order on $\mathbb{N}$, with ∞ being the largest element. Multiplication is ordinary multiplication in $\mathbb{N}$, extended to ∞ as follows:

$$\infty \cdot x = x \cdot \infty \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x = 0, \\ \infty, & \text{otherwise.} \end{cases}$$

The constants 0 and 1 in the semiring are the natural numbers 0 and 1, respectively.

To define $\sum$ in this algebra, define the *support* of an infinite sequence $x = (x_n \mid n \geq 0)$ to be the set

$$\mathsf{supp}\, x \stackrel{\text{def}}{=} \{n \mid x_n \neq 0\}.$$

We define

$$\sum_n x_n \stackrel{\text{def}}{=} \begin{cases} \sum_{n \in \mathsf{supp}\, x} x_n, & \text{if } \mathsf{supp}\, x \text{ is finite,} \\ \infty, & \text{otherwise.} \end{cases}$$

One can show that infinitary associativity, commutativity, and distributivity are satisfied, and $0^* = 1$. However, $0^{**} = 1^* = \infty$, so $\sum$ is not idempotent (since $1^* = 1 + 1 + 1 + \cdots$) and $0^{**} \neq 0^*$.

It is conjectured that the axiomatization of [1] does not imply infinitary commutativity. In particular, it is conjectured that

$$x_0 + x_1 + x_2 + \cdots = (x_0 + x_2 + x_4 + \cdots) + (x_1 + x_3 + x_5 + \cdots)$$

is not provable.

One can show that our official definition of closed semirings is equivalent to a countable summation operator $\sum$ satisfying infinitary associativity, commutativity, idempotence, and distributivity. Surely supremum is associative, commutative, and idempotent, and the axiom (3.8) gives distributivity as well.

Conversely, if $\sum$ is infinitely associative, commutative, and idempotent, then its value on a given sequence is independent of the order and multiplicity of elements occurring in the sequence. Thus we might as well define $\sum$ on finite or countable subsets instead of sequences. In this view, $\sum$ gives the supremum with respect to the natural order $\leq$. To see this, let A be a nonempty finite or countable set of elements. If $x \in A$, then

$$x + \sum A = \sum(A \cup \{x\}) = \sum A,$$

thus $x \leq \sum A$; and if $x \leq y$ for all $x \in A$, then $x + y = y$ for all $x \in A$, thus

$$(\sum A) + y = (\sum_{x \in A} x) + (\sum_{x \in A} y) = \sum_{x \in A} (x + y) = \sum_{x \in A} y = y,$$

therefore

$$\sum A \leq y.$$

Thus $\sum$ gives the supremum of countable sets.

Conway's Hierarchy

Closed semirings and star-continuous Kleene algebras are strongly related to several classes of algebras defined by Conway in his 1971 monograph [29]. Conway's S-algebras are similar to closed semirings, except that arbitrary sums, not just countable ones, are permitted. A better name for S-algebras might be *complete idempotent semirings*. The operation * is defined as in closed semirings in terms of $\sum$, and again this seems to be the main purpose of $\sum$.

Conway's N-algebras are algebras of signature $(+, \cdot, ^*, 0, 1)$ that are subsets of S-algebras containing 0 and 1 and closed under (finite) $+$, $\cdot$, and *. We will show later that the classes of N-algebras and star-continuous Kleene algebras coincide.

An R-algebra is any algebra of signature $(+, \cdot, ^*, 0, 1)$ satisfying the equational theory of the N-algebras.

According to the definition in [29], an S-algebra

$$(S, \sum, \cdot, 0, 1)$$

is similar to a closed semiring, except that $\sum$ is defined not on sequences but on multisets of elements of S. A *multiset* is a set whose elements have multiplicity; equivalently, it is an equivalence class of sequences, where two sequences are considered equivalent if one is a permutation of the other. In other words, a multiset is like a sequence, except that we ignore the order of the elements. However, there is no cardinality restriction on the multiset. One consequence of this approach is that $\sum$ is too big to be represented in Zermelo-Fraenkel set theory! Since $\sum$ is a function that must be defined on multisets of arbitrary cardinality, it cannot be a set itself. However, as with closed semirings, the value that $\sum$ takes on a given multiset is independent of the multiplicity of the elements, so $\sum$ might as well be defined on subsets of S instead of multisets. So defined, $\sum A$ gives the supremum of A with respect to the order $\leq$. (We assume the axiom $\sum\{a\} = a$, which is omitted in [29].)

Thus, the only essential difference between S-algebras and closed semirings is that closed semirings are only required to contain suprema of countable sets, whereas S-algebras must contain suprema of all sets. Thus every S-algebra is automatically a closed semiring and every continuous semiring morphism (semiring morphism preserving all suprema) is automatically ω-continuous (preserves all countable suprema), and these notions coincide on countable algebras.

In a subsequent lecture we will show some very strong relationships among these classes of algebras. We will eventually show that the R-algebras, Kleene algebras, star-continuous Kleene algebras (a.k.a. N-algebras), closed semirings, and S-algebras each contain the next in the list, and all inclusions are strict. Moreover, each star-continuous Kleene algebra extends in a canonical way to a closed semiring, and each closed semiring to an S-algebra, by a construction known as *ideal completion*.

Other Approaches

There are many other approaches besides these, which we will not consider in this course.

Many authors consider Kleene algebra as synonymous with relation algebra and are not opposed to adding other relational operators such as residuation and complementation. Relation algebras were first studied by Tarski and his students and colleagues [**?**, 92, 91, 54]; see also [87, 88, 102, 65]. Bloom and Ésik [14, 12, 13] study a related structure called *iteration theories*.

In [100, **?**], Pratt gives two definitions of Kleene algebras in the context of dynamic algebra. In [100], Kleene algebras are defined to be the Kleenean component of separable dynamic algebras; in [**?**], this class is enlarged to contain all subalgebras of such algebras.

Generalizations of Kleene's and Parikh's Theorems have been given by Kuich [78] in *ℓ-complete semirings*, which are similar to S-algebras in all respects except that idempotence of $\sum$ is replaced by a weaker condition called ℓ-completeness.

Lecture 4

Characterizing the Equational Theory

Most of the early work on Kleene algebra was directed toward characterizing the equational theory of the regular sets. These are equations such as $(x+y)^* = x^*(yx^*)^*$ and $x(yx)^* = (xy)^*x$ that hold under the standard interpretation of regular expressions as regular sets of strings.

It turns out that this theory is quite robust and can be characterized in many different ways. We have defined several different but related classes of algebras: Kleene algebras, star-continuous Kleene algebras, closed semirings, and Conway's S-algebras, N-algebras, and R-algebras, all defined axiomatically, as well as relational and language-theoretic algebras defined model-theoretically. All these classes of models have the same equational theory over the signature $+$, $\cdot$, *, 0, 1 of Kleene algebra, and it is the same as the equational theory of the regular sets.

Let us say more carefully what we are talking about. Let σ denote the signature $+$, $\cdot$, *, 0, 1 of Kleene algebra. A *σ-algebra* is any structure of signature σ. This just means that there are distinguished binary operations $+$ and $\cdot$, a distinguished unary operation *, and distinguished constants 0 and 1 defined on C. The structure need not satisfy the axioms of Kleene algebra.

The set of regular expressions $\mathsf{Exp}\,\Sigma$ over an alphabet Σ can be regarded as a σ-algebra. The elements of $\mathsf{Exp}\,\Sigma$ are just the well-formed terms over variables Σ and operators $+$, $\cdot$, *, 0, 1. The distinguished operations are the syntactic ones; for example, $+$ in $\mathsf{Exp}\,\Sigma$ is the binary op-

eration that takes regular expressions s and t and produces the regular expression $s + t$.

For any two σ-algebras C and C', a *homomorphism* from C to C' is a map $h : C \to C'$ that commutes with all the distinguished operations and constants of σ; that is, for all $x, y \in C$,

$$h(x + y) = h(x) + h(y) \qquad h(xy) = h(x) \cdot h(y)$$
$$h(x^*) = h(x)^* \qquad h(0) = 0 \qquad h(1) = 1.$$

Here the operators and constants on the left-hand side are interpreted in C and on the right-hand side in C'. A homomorphism h is

- an *epimorphism* if it is onto; that is, if for all $y \in C'$, there is an $x \in C$ such that $h(x) = y$;
- a *monomorphism* if it is one-to-one; that is, if for all $x, y \in C$, if $h(x) = h(y)$ then $x = y$;
- an *isomorphism* if it is both an epimorphism and a monomorphism.

An *interpretation* is just a homomorphism whose domain is $\mathsf{Exp}\,\Sigma$. For example, let $\mathsf{Reg}\,\Sigma$ denote the Kleene algebra of regular sets over alphabet Σ. The *canonical interpretation* over $\mathsf{Reg}\,\Sigma$ is the unique homomorphism $R_\Sigma : \mathsf{Exp}\,\Sigma \to \mathsf{Reg}\,\Sigma$ such that $R_\Sigma(a) = \{a\}$, $a \in \Sigma$. We will show that this interpretation alone characterizes the equational theory of Kleene algebras, as well as all the other classes of algebras mentioned above (Theorem 14.1).

In general, for any σ-algebra C and any function $h : \Sigma \to C$ defined on Σ, h extends uniquely to a homomorphism $\widehat{h} : \mathsf{Exp}\,\Sigma \to C$; that is, h and $\widehat{h}$ agree on Σ. Because of this property, the structure $\mathsf{Exp}\,\Sigma$ is called the *free σ-algebra on generators* Σ. We say *the* free σ-algebra because it is unique up to isomorphism. Intuitively, once we know how to interpret the letters in Σ, that uniquely determines the interpretation of any regular expression over Σ.

Let s, t be regular expressions and let $I : \mathsf{Exp}\,\Sigma \to C$ be an interpretation. We say that the equation $s = t$ *holds under interpretation* I if $I(s) = I(t)$. We say that $s = t$ *holds in* C or that C *satisfies* $s = t$ if $s = t$ holds under all interpretations $I : \mathsf{Exp}\,\Sigma \to C$. If $\mathcal{A}$ is a class of algebras or a class of interpretations, we say that $s = t$ *holds in* $\mathcal{A}$ if it holds in all members of $\mathcal{A}$. The *equational theory* of $\mathcal{A}$ is the set of equations that hold in $\mathcal{A}$. The equational theory of $\mathcal{A}$ is denoted $\mathcal{E}\,\mathcal{A}$.

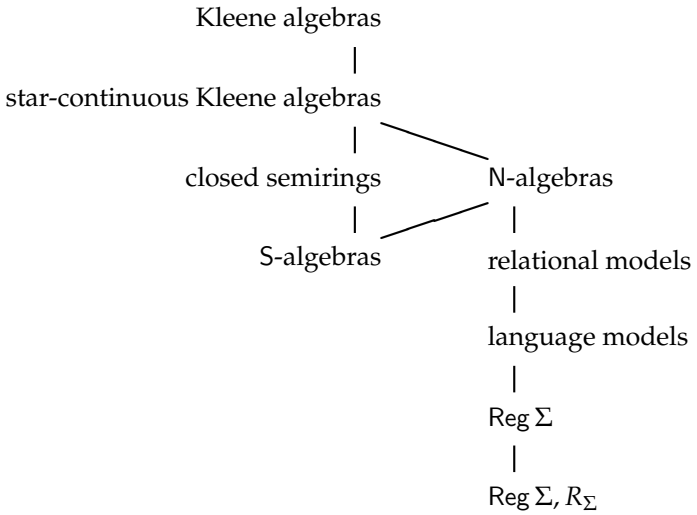
Theorem 4.1 *The following classes of algebras all have the same equational theory: Kleene algebras, star-continuous Kleene algebras, closed semirings,* S*-algebras,* N*-algebras,* R*-algebras, language models, and relational models. Moreover, an equation $s = t$ over alphabet Σ is a member of this theory iff it holds under the canonical interpretation $R_\Sigma : \mathsf{Exp}\,\Sigma \to \mathsf{Reg}\,\Sigma$.*

One can see from this theorem that the equational theory of Kleene algebras is quite robust indeed. If the equational theory were all that we were interested in, there would not be much more to say.

Some Constructions

We will not be able to complete the proof of Theorem 14.1 today. Some parts of the theorem follow immediately from inclusion relationships among the classes of interpretations, but others are more difficult.

First we note that if $\mathcal{A}$ and $\mathcal{B}$ are two classes of algebras or classes of interpretations and $\mathcal{A} \subseteq \mathcal{B}$, then $\mathcal{E}\,\mathcal{B} \subseteq \mathcal{E}\,\mathcal{A}$, since any equation that holds in all members of $\mathcal{B}$ must perforce hold in all members of $\mathcal{A}$. We have already established the following inclusions among the classes mentioned in Theorem 14.1:



If class $\mathcal{A}$ occurs above $\mathcal{B}$ in this diagram and there is a path from $\mathcal{A}$ down to $\mathcal{B}$, then $\mathcal{E}\,\mathcal{A} \subseteq \mathcal{E}\,\mathcal{B}$. Note that for the two lowest entries in this diagram, the upper one $\mathsf{Reg}\,\Sigma$ refers to the equations that hold under any interpretation in $\mathsf{Reg}\,\Sigma$, whereas the lower one $\mathsf{Reg}\,\Sigma, R_\Sigma$ refers to the equations that hold under the canonical interpretation only.

First we observe that the equational theories of the S-algebras and the N-algebras coincide. Recall that the N-algebras are the subsets of S-algebras closed under the Kleene algebra operations considered as σ-algebras. We have $\mathcal{E}\,\mathsf{N} \subseteq \mathcal{E}\,\mathsf{S}$, since every S-algebra is a subalgebra of itself, therefore is an N-algebra. Conversely, since equations are universal sentences, any equation holding in an S-algebra A holds in any subalgebra of A; therefore $\mathcal{E}\,\mathsf{S} \subseteq \mathcal{E}\,\mathsf{N}$.

This observation says that the equational theories of the following classes of interpretations are linearly ordered by inclusion as follows: Kleene algebras, star-continuous Kleene algebras, closed semirings, S-algebras, N-algebras, relational models, language models, $\mathsf{Reg}\,\Sigma$, and $\mathsf{Reg}\,\Sigma, R_\Sigma$.

We might also add R-algebras to this list. Recall that R-algebras are those algebras that satisfy all the same equations as N-algebras, thus $\mathcal{E}\,\mathsf{R} = \mathcal{E}\,\mathsf{N}$. It will turn out that all the algebras in the diagram above are R-algebras, since they all share the same equational theory, so the class of R-algebras sits at the very top of the diagram above and at the head of the list in the previous paragraph.

However, the concept of R-algebra is not very interesting or useful for axiomatic purposes. Conway [29, p. 102] gives a four-element R-algebra R_4 that is not a star-continuous Kleene algebra. The elements of R_4 are $\{0,1,2,3\}$, and the operations are given by the following tables:

$+$	0	1	2	3
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	2	3
3	0	3	3	3

$*$	
0	1
1	1
2	3
3	3

To show that R_4 is an R-algebra, it suffices to construct an epimorphism $h : \mathsf{Reg}\,\Sigma \to R_4$, since any equation that holds in a σ-algebra also holds in all its homomorphic images. Take $h(0) \stackrel{\text{def}}{=} 0$, $h(1) \stackrel{\text{def}}{=} 1$, and for any other set A,

$$h(A) \stackrel{\text{def}}{=} \begin{cases} 2, & \text{if } A \text{ is finite,} \\ 3, & \text{otherwise.} \end{cases}$$

One can verify easily that this is an epimorphism, therefore R_4 is an R-algebra. It is not a star-continuous Kleene algebra, since $2^n = 2$ for all n, but $2^* = 3$. It is also easily shown that all finite Kleene algebras are star-continuous, therefore R_4 is not a Kleene algebra either.

The family $\mathsf{Reg}\,\Sigma$ of regular events over an alphabet Σ gives an example of a star-continuous Kleene algebra that is not a closed semiring. If A is nonregular, the countable set $\{\{x\} \mid x \in A\}$ has no supremum. However, the power set of Σ^* does form a closed semiring.

To construct a closed semiring that is not an S-algebra, we might take the countable and co-countable subsets of ω_1 (the first uncountable ordinal) with operations of set union for $\sum$, set intersection for $\cdot$, $\varnothing$ for 0, ω_1 for 1, and $A^* = \omega_1$.

To complete the picture, we should construct a Kleene algebra that is not star-continuous. Let ω^2 denote the set of ordered pairs of natural numbers and let $\bot$ and $\top$ be new elements. Order these elements such that $\bot$ is

the minimum element, $\top$ is the maximum element, and ω^2 is ordered lexicographically in between. Define $+$ to give the supremum in this order. Define $\cdot$ as follows:

$$x \cdot \bot = \bot \cdot x = \bot \quad x \cdot \top = \top \cdot x = \top,\ x \neq \bot \quad (a,b) \cdot (c,d) = (a+c, b+d).$$

Then $\bot$ is the additive identity and $(0,0)$ is the multiplicative identity. Finally, define

$$a^* = \begin{cases} (0,0), & \text{if } a = \bot \text{ or } a = (0,0), \\ \top, & \text{otherwise.} \end{cases}$$

It is easily checked that this is a Kleene algebra. We verify the axiom

$$ax \leq x \Rightarrow a^* x \leq x$$

explicitly. Assuming $ax \leq x$, we wish to show $a^* x \leq x$. If $a = \bot$ or $a = (0,0)$, then $a^* = (0,0)$ and we are done, since $(0,0)$ is the multiplicative identity. If $x = \bot$ or $x = \top$, we are done. Otherwise, $a > (0,0)$ and $x = (u,v)$, in which case $ax > x$, contradicting the assumption.

This Kleene algebra is not star-continuous, since $(0,1)^* = \top$, but

$$\sum_n (0,1)^n = \sum_n (0,n) = (1,0).$$

Lecture 5

Completeness of Star-Continuity

We argued in the previous lecture that the equational theory of each of the following classes of interpretations is contained in the next in the list: Kleene algebras, star-continuous Kleene algebras, closed semirings, S-algebras, N-algebras, relational models, language models, $\mathsf{Reg}\,\Sigma$, and $\mathsf{Reg}\,\Sigma$ under the single canonical interpretation R_Σ.

In this lecture we show that all these classes from star-continuous Kleene algebra onward have the same equational theory. It suffices to show that any equation that holds under the canonical interpretation $R_\Sigma : \mathsf{Exp}\,\Sigma \to \mathsf{Reg}\,\Sigma$ holds in all star-continuous Kleene algebras. Later on we will add Kleene algebras to this list as well.

Lemma 5.1 *For any regular expressions $s, t, u \in \mathsf{Exp}\,\Sigma$, the following property holds in any star-continuous Kleene algebra K:*

$$stu = \sup_{x \in R_\Sigma(t)} sxu.$$

In other words, if K is star-continuous, then under any interpretation $I : \mathsf{Exp}\,\Sigma \to K$, the supremum of the set

$$\{I(sxu) \mid x \in R_\Sigma(t)\}$$

exists and is equal to $I(stu)$. In particular, in the special case $s = u = 1$,

$$t = \sup_{x \in R_\Sigma(t)} x.$$

The star-continuity axiom states that

$$ab^*c = \sup_{n \geq 0} ab^n c = \sup_{x \in R_\Sigma(b^*)} axc,$$

that is, K is star-continuous if under any interpretation $I : \mathsf{Exp}\{a, b, c\} \to K$, the supremum of $\{I(ab^n c) \mid n \geq 0\}$ exists and is equal to $I(ab^*c)$. This is a special case of the lemma for $s = a$, $t = b^*$, and $u = c$. The lemma expresses a natural extension of the star-continuity property to all expressions s, t, u. Later on we will do the same thing with the axioms of Kleene algebra; there we will extend the axioms, which give the ability to solve one linear inequality, to a theorem that gives the solution to any finite system of inequalities.

Proof. Let K be an arbitrary star-continuous Kleene algebra. We proceed by induction on the structure of t. There are three base cases, corresponding to the regular expressions $a \in \Sigma$, 1, and 0. For $a \in \Sigma$, we have $R_\Sigma(a) = \{a\}$ and

$$\sup_{x \in R_\Sigma(a)} sxu = sau,$$

since the supremum of a singleton set is just the unique element of that set. The case of 1 is similar, since $R_\Sigma(1) = \{\varepsilon\}$. Finally, since $R_\Sigma(0) = \varnothing$ and since 0 is the least element in K and therefore the supremum of the empty set,

$$\sup_{x \in R_\Sigma(0)} sxu = \sup \varnothing = 0 = s0u.$$

There are three cases to the inductive step, one for each of the operators $+$, $\cdot$, *. We give a step-by-step argument for the case $+$, followed by a justification of each step.

$$\begin{aligned}
s(t_1 + t_2)u &= st_1u + st_2u && (5.1)\\
&= \sup_{x \in R_\Sigma(t_1)} sxu + \sup_{y \in R_\Sigma(t_2)} syu && (5.2)\\
&= \sup_{z \in R_\Sigma(t_1) \cup R_\Sigma(t_2)} szu && (5.3)\\
&= \sup_{z \in R_\Sigma(t_1 + t_2)} szu. && (5.4)
\end{aligned}$$

Equation (5.1) follows from the distributive laws of Kleene algebra; (5.2) follows from the induction hypothesis on t_1 and t_2; (5.3) follows from the general property of Kleene algebras that if A and B are two sets whose suprema $\sup A$ and $\sup B$ exist, then the supremum of $A \cup B$ exists and is equal to $\sup A + \sup B$ (this requires proof—see below); finally, equation (5.4) follows from the fact that the interpretation R_Σ is a homomorphism.

The general property used in equation (5.3) states that if A and B are two subsets whose suprema $\sup A$ and $\sup B$ exist, then the supremum $\sup(A \cup B)$ exists and is equal to $\sup A + \sup B$.

To prove this, we must show two things:

(i) $\sup A + \sup B$ is an upper bound for $A \cup B$; that is, for any $x \in A \cup B$, $x \leq \sup A + \sup B$; and

(ii) $\sup A + \sup B$ is the least such upper bound; that is, for any other upper bound y of the set $A \cup B$, $\sup A + \sup B \leq y$.

To show (i),

$$\begin{aligned} x \in A \cup B &\Rightarrow x \in A \text{ or } x \in B \\ &\Rightarrow x \leq \sup A \text{ or } x \leq \sup B \\ &\Rightarrow x \leq \sup A + \sup B. \end{aligned}$$

To show (ii), let y be any other upper bound for $A \cup B$. Then

$$\begin{aligned} \forall x \in A \cup B\ x \leq y &\Rightarrow \forall x \in A\ x \leq y \text{ and } \forall x \in B\ x \leq y \\ &\Rightarrow \sup A \leq y \text{ and } \sup B \leq y \\ &\Rightarrow \sup A + \sup B \leq y + y\ =\ y. \end{aligned}$$

Now we give a similar chain of equalities for the case of the operator $\cdot$, but omit the justifications.

$$\begin{aligned} s(t_1 t_2)u &= st_1(t_2 u) \\ &= \sup_{x \in R_\Sigma(t_1)} sx(t_2 u) \\ &= \sup_{x \in R_\Sigma(t_1)} (sx)t_2 u \\ &= \sup_{x \in R_\Sigma(t_1)} \sup_{y \in R_\Sigma(t_2)} sxyu \\ &= \sup_{x \in R_\Sigma(t_1),\ y \in R_\Sigma(t_2)} sxyu \qquad (5.5) \\ &= \sup_{z \in R_\Sigma(t_1 t_2)} szu. \end{aligned}$$

The general property of suprema used in step (5.5) is that if $\mathcal{C}$ is a collection of subsets such that $\sup A$ exists for all $A \in \mathcal{C}$, and if $\sup_{A \in \mathcal{C}} \sup A$ exists, then $\sup \bigcup \mathcal{C}$ exists and is equal to $\sup_{A \in \mathcal{C}} \sup A$.

Finally, for the case *, we have

$$\begin{aligned} st^*u &= \sup_{n \geq 0} st^n u \\ &= \sup_{n \geq 0} \sup_{x \in R_\Sigma(t^n)} sxu \\ &= \sup_{x \in \bigcup_{n \geq 0} R_\Sigma(t^n)} sxu \\ &= \sup_{x \in R_\Sigma(t^*)} sxu. \end{aligned}$$

□

Theorem 5.2 *Let s, t be regular expressions over Σ. The equation $s = t$ holds under all interpretations in all star-continuous Kleene algebras iff $R_\Sigma(s) = R_\Sigma(t)$.*

Proof. The forward implication is immediate, since $\mathsf{Reg}\,\Sigma$ is a star-continuous Kleene algebra. Conversely, by two applications of Lemma 5.1, if $R_\Sigma(s) = R_\Sigma(t)$, then under any interpretation in any star-continuous Kleene algebra,

$$s = \sup_{x \in R_\Sigma(s)} x = \sup_{x \in R_\Sigma(t)} x = t.$$

□

Free Algebras

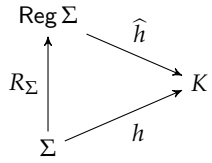
We have shown that the equational theory of the star-continuous Kleene algebras coincides with the equations true in $\mathsf{Reg}\,\Sigma$ under the interpretation R_Σ. Another way of saying this is that $\mathsf{Reg}\,\Sigma$ is the *free star-continuous Kleene algebra on generators* Σ. The term *free* intuitively means that $\mathsf{Reg}\,\Sigma$ is free from any equations except those that it is forced to satisfy in order to be a star-continuous Kleene algebra.

Formally, a member A of a class of algebraic structures $\mathcal{C}$ of the same signature is said to be *free on generators X for the class $\mathcal{C}$* if

- A is generated by X;
- any function h from X into another algebra $B \in \mathcal{C}$ extends to a homomorphism $\widehat{h} : A \to B$.

The extension is necessarily unique, since a homomorphism is completely determined by its action on a generating set.

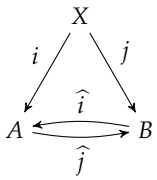
Thus to say that $\mathsf{Reg}\,\Sigma$ is the free star-continuous Kleene algebra on generators Σ says that $\mathsf{Reg}\,\Sigma$ is generated by Σ (actually, by $\{\{a\} \mid a \in \Sigma\} = \{R_\Sigma(a) \mid a \in \Sigma\}$), and for any star-continuous Kleene algebra K and map $h : \Sigma \to K$, there is a homomorphism $\widehat{h} : \mathsf{Reg}\,\Sigma \to K$ such that the following diagram commutes:



(5.6)

In other words, $\widehat{h} \circ R_\Sigma = h$.

Free algebras, if they exist, are unique up to isomorphism. If A and B are free on generators X for a class $\mathcal{C}$, let i and j be the embeddings of X into A and B, respectively. Since both are free, they extend to homomorphisms $\widehat{i} : B \to A$ and $\widehat{j} : A \to B$, respectively.



Thus $\widehat{i} \circ j = i$ and $\widehat{j} \circ i = j$. Then $\widehat{j} \circ \widehat{i} \circ j = \widehat{j} \circ i = j$, so for any $x \in X$, $(\widehat{j} \circ \widehat{i})(j(x)) = j(x)$. Since B is generated by $\{j(x) \mid x \in X\}$, this says that $\widehat{j} \circ \widehat{i} : B \to B$ is the identity—it agrees with the identity on X, and homomorphisms are uniquely determined by their action on a generating set. Symmetrically, $\widehat{i} \circ \widehat{j} : A \to A$ is also the identity, so $\widehat{i}$ and $\widehat{j}$ are inverses, therefore A and B are isomorphic.

Congruence Relations and the Quotient Construction

Any class of algebras defined by universal equations or equational implications, even infinitary ones, has free algebras. Recall that the axioms for star-continuous Kleene algebra are of this form:

$$xy^n z \leq xy^* z, \quad n \geq 0 \qquad\qquad \bigwedge_{n \geq 0} (xy^n z \leq w) \Rightarrow xy^* z \leq w.$$

There is a general technique for constructing free algebras called a *quotient construction*. Here we give a brief general account of the quotient construction and how to apply it to obtain free algebras.

Fix a signature σ. Let A be any σ-algebra. A binary relation $\equiv$ on A is call a *congruence* if

(i) $\equiv$ is an equivalence relation (reflexive, symmetric, transitive);

(ii) $\equiv$ is respected by all the distinguished operations of the signature; that is, if f is n-ary, and if $x_i \equiv y_i$, $1 \leq i \leq n$, then $f(x_1, \ldots, x_n) \equiv f(y_1, \ldots, y_n)$. For example, for the signature of Kleene algebra, this means

$$\begin{aligned} x_1 \equiv y_1 \wedge x_2 \equiv y_2 &\Rightarrow x_1 + x_2 \equiv y_1 + y_2 \\ x_1 \equiv y_1 \wedge x_2 \equiv y_2 &\Rightarrow x_1 x_2 \equiv y_1 y_2 \\ x \equiv y &\Rightarrow x^* \equiv y^*. \end{aligned}$$

The *kernel* of a homomorphism $h : A \to B$ is

$$\ker h \stackrel{\text{def}}{=} \{(x, y) \mid h(x) = h(y)\}.$$

One can show that the kernel of any homomorphism with domain A is a congruence on A (Exercise **??**).

Conversely, given any congruence $\equiv$ on A, one can construct a σ-algebra B and an epimorphism $h : A \to B$ such that $\equiv$ is the kernel of h. This is called the *quotient construction*. For any $x \in A$, define

$$[x] \stackrel{\text{def}}{=} \{y \in A \mid x \equiv y\},$$

the *congruence class* of x. Let

$$A/\!\equiv \stackrel{\text{def}}{=} \{[x] \mid x \in A\}.$$

One can make this into a σ-algebra by defining

$$f^{A/\equiv}([x_1], \ldots, [x_n]) \stackrel{\text{def}}{=} [f^A(x_1, \ldots, x_n)].$$

The properties of congruence ensure that $f^{A/\equiv}$ is well defined and that the map $x \mapsto [x]$ is an epimorphism $A \to A/\!\equiv$.

Free Algebras as Quotients

Now we apply the quotient construction to obtain free algebras. As previously observed, if σ is any signature, the set of well-formed terms $T_\sigma(X)$

over variables X can be regarded as a σ-algebra in which the operations of σ have their syntactic interpretation. For the signature of Kleene algebra, we have been calling the variables Σ, and the terms over Σ are the regular expressions $\mathsf{Exp}\,\Sigma$.

Let Δ be a set of equations or equational implications over $T_\sigma(X)$, and let $\mathrm{Mod}\,\Delta$ denote the class of σ-algebras that satisfy Δ. For example, if Δ consists of the axioms of star-continuous Kleene algebra, then $\mathrm{Mod}\,\Delta$ will be the class of all star-continuous Kleene algebras.

Let $\equiv$ be the smallest congruence on terms in $T_\sigma(X)$ containing all substitution instances of equations in Δ and closed under all substitution instances of equational implications in Δ. The relation $\equiv$ can be built inductively, starting with the substitution instances of equations in Δ and the reflexivity axiom $s \equiv s$ and adding pairs $s \equiv t$ as required by the substitution instances of equational implications in Δ and the symmetry, transitivity, and congruence rules.

One can now show that the quotient $T_\sigma(X)/\!\equiv$ is the free $\mathrm{Mod}\,\Delta$ algebra on generators X. For any algebra A satisfying Δ and any map $h : X \to A$, h extends uniquely to a homomorphism $T_\sigma(X) \to A$, which we also denote by h. The kernel of h is the set of equations satisfied by A under interpretation h. Since A satisfies Δ, the kernel of h contains all the equations of Δ and is closed under the equational implications of Δ. Since $\equiv$ is the smallest such congruence, $\equiv$ refines (is contained in) $\ker h$. Thus we can define $\widehat{h}([x]) \stackrel{\text{def}}{=} h(x)$ and the resulting map $\widehat{h}$ will be well defined. This is the desired homomorphism $T_\sigma(X)/\!\equiv \to A$.

For star-continuous Kleene algebra, the free algebra given by the quotient construction is $\mathsf{Exp}\,\Sigma/\!\equiv$, where $\equiv$ is the smallest congruence containing all substitution instances of the axioms of idempotent semirings, e.g. $s + (t + u) \equiv (s + t) + u$ for all $s, t, u \in \mathsf{Exp}\,\Sigma$, etc., and $st^n u \leqq st^* u$ for all $s, t, u \in \mathsf{Exp}\,\Sigma$ and $n \geq 0$ (where $s \leqq t$ is an abbreviation for $s + t \equiv t$), and contains $st^* u \leqq w$ whenever $st^n u \leqq w$ for all $n \geq 0$.

To show that $\mathsf{Reg}\,\Sigma$ is free (therefore isomorphic to $\mathsf{Exp}\,\Sigma/\!\equiv$), let $h : \Sigma \to K$ be an arbitrary function into a star-continuous Kleene algebra K, and extend h to a homomorphism $h : \mathsf{Exp}\,\Sigma \to K$. By Theorem 5.2, the set of equations that hold under the interpretation R_Σ, which is $\ker R_\Sigma$, is contained in the set of equations that hold under any interpretation in any star-continuous Kleene algebra; in particular under the interpretation h in K. Thus $\ker R_\Sigma$ refines $\ker h$. This says that we can define $\widehat{h}(R_\Sigma(s)) \stackrel{\text{def}}{=} h(s)$, and the resulting map $\widehat{h} : \mathsf{Reg}\,\Sigma \to K$ will be well defined. This is the desired homomorphism making the diagram (5.6) commute.

Lecture 6

Relations Among Algebras

The notion of free algebra described in the previous lecture is an example of a more general phenomenon called *adjunction*. An adjunction is a way of describing a particular relationship between categories of algebraic structures.

There are many examples of adjunctions in mathematics, but one very common occurrence is when some category C of algebras has more structure than another category D, and there is a canonical way to extend any D-algebra to a C-algebra. The construction normally constitutes a functor $\mathsf{F} : \mathsf{D} \to \mathsf{C}$ called the *left adjoint* of the adjunction. There is normally a corresponding *forgetful functor* $\mathsf{G} : \mathsf{C} \to \mathsf{D}$ going in the opposite direction that ignores the extra structure, called the *right adjoint*.

For example, every closed semiring is a star-continuous Kleene algebra, because the * operation can be defined in terms of the countable supremum operation and the star-continuity condition follows from the axioms of closed semirings. This constitutes a forgetful functor from the category of closed semirings, the category with more structure, to the category of star-continuous Kleene algebras, the category with less structure. Typically, forgetful functors do not modify the sets in any way, they just ignore some structure.

In the other direction, not every star-continuous KA is a closed semiring—the regular sets are not—but it turns out that every star-continuous KA can be extended to a closed semiring in a canonical way.

For $\mathrm{Reg}\,\Sigma$, this construction would give 2^{Σ^*}, the closed semiring of all subsets of Σ^*.

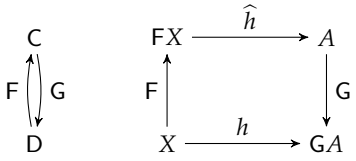
We will show that adjunctions characterize the relationships among the following categories:

- KA^*, the category of star-continuous Kleene algebras and Kleene algebra morphisms;
- CS, the category of closed semirings and ω-continuous semiring morphisms (semiring morphisms that preserve suprema of countable sets); and
- SA, the category of S-algebras and continuous semiring morphisms (semiring morphisms that preserve arbitrary suprema).

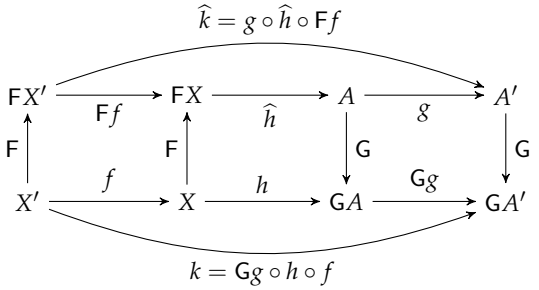
The construction of the free algebra on a set of generators is an example of such an adjunction in which the category with less structure is Set, the category with no structure at all.

Adjunction

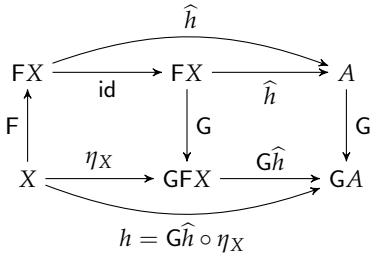
Formally, let $\mathsf{F} : \mathsf{D} \to \mathsf{C}$ and $\mathsf{G} : \mathsf{C} \to \mathsf{D}$ be functors between two categories C and D. Think of D as the category with less structure. We say that F is a *left adjoint* of G and that G is a *right adjoint* of F if for any D-algebra X and C-algebra A, there is a natural one-to-one correspondence between morphisms $h : X \to \mathsf{G}A$ in D and morphisms $\widehat{h} : \mathsf{F}X \to A$ in C.

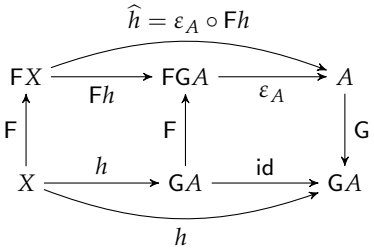


By *natural* we just mean that the one-to-one correspondence commutes with morphisms in the two categories; that is, if $f : X' \to X$ and $g : A \to A'$, and if $k = \mathsf{G}g \circ h \circ f : X' \to \mathsf{G}A'$, then $\widehat{k} = g \circ \widehat{h} \circ \mathsf{F}f : \mathsf{F}X' \to A'$.



There are some special morphisms that exist. When $A = \mathsf{F}X$, the morphism $\eta_X : X \to \mathsf{GF}X$ down below corresponding to the identity $\mathsf{id} : \mathsf{F}X \to \mathsf{F}X$ up top is called the *unit* of the adjunction. Similarly, when $X = \mathsf{G}A$, the morphism $\varepsilon_A : \mathsf{FG}A \to A$ up top corresponding to the identity $\mathsf{id} : \mathsf{G}A \to \mathsf{G}A$ down below is called the *counit* of the adjunction. It turns out that the one-to-one correspondence between h and $\widehat{h}$ is uniquely determined by η_X and ε_A:

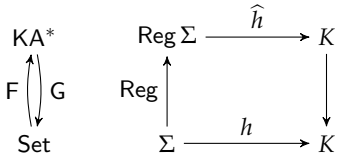




The left-hand diagram shows that $h = \mathsf{G}\widehat{h} \circ \eta_X$ and the right-hand diagram shows that $\widehat{h} = \varepsilon_A \circ \mathsf{F}h$.

In all instances we will consider, G is a *forgetful functor*, which means that the algebras $\mathsf{G}A$ and A have the same carrier, and $\mathsf{G}g$ is set-theoretically the same function as g; it just has less structure to preserve.

In the free construction of the last lecture, the two categories are the category KA^* of star-continuous Kleene algebras and Kleene algebra homomorphisms and the category Set of sets and set functions. The free construction is the left adjoint of the forgetful functor that associates to every star-continuous Kleene algebra its underlying set.

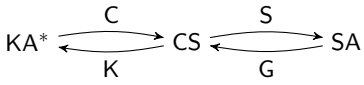


Here the unit of the adjunction is the map $R_\Sigma : \Sigma \to \mathsf{Reg}\,\Sigma$ given by $R_\Sigma(a) = \{a\}$.

As observed, every closed semiring C gives a star-continuous Kleene algebra $\mathsf{K}C$ by defining $x^* = \sum_n x^n$. Also, if $h : C \to C'$ is an ω-continuous semiring morphism between closed semirings, then h must preserve *, therefore is a Kleene algebra morphism $\mathsf{K}h : \mathsf{K}S \to \mathsf{K}S'$. Similarly, every S-algebra S is a closed semiring $\mathsf{G}S$, and every complete semiring morphism is ω-complete. Thus we have a forgetful functors $\mathsf{G} : \mathsf{SA} \to \mathsf{CS}$ and $\mathsf{K} : \mathsf{CS} \to \mathsf{KA}^*$.

In the other direction, not every star-continuous Kleene algebra is a closed semiring. However, it is possible to construct, in a canonical way, a closed semiring $\mathsf{C}K$ extending any star-continuous Kleene algebra K. Similarly, although not every closed semiring is an S-algebra, every closed semiring can be extended to one.

Furthermore, any Kleene algebra homomorphism $h : K \to K'$ extends naturally to an ω-complete semiring morphism $\mathsf{C}g : \mathsf{C}K \to \mathsf{C}K'$, and every ω-complete semiring morphism $g : C \to C'$ extends to a complete semiring morphism $\mathsf{S}g : \mathsf{S}C \to \mathsf{S}C'$. The functors $\mathsf{C} : \mathsf{KA}^* \to \mathsf{CS}$ and $\mathsf{S} : \mathsf{CS} \to \mathsf{SA}$ are left adjoints to K and G, respectively.



Completion by Star-Ideals

The basic construction used here is known as *completion by star-ideals* and was used by Conway to extend a star-continuous Kleene algebra to an S-algebra [29, Theorem 1, p. 102]. Thus Conway's construction is equivalent to the composition $\mathsf{S} \circ \mathsf{C}$. The construction C, which shows that every star-continuous Kleene algebra is embedded in a closed semiring, can be described as a completion by *countably generated* star-ideals.

Definition 6.1 (Conway [29]) *Let K be a star-continuous Kleene algebra. A* star-ideal *is a subset I of K such that*

- *I is nonempty*
- *I is closed under $+$*
- *I is closed downward under $\leq$*
- *if $ab^n c \in I$ for all $n \geq 0$, then $ab^* c \in I$.*

A nonempty set A *generates* a star-ideal I if I is the smallest star-ideal containing A. We write $\langle A \rangle$ to denote the star-ideal generated by A. A star-ideal is *countably generated* if it has a countable generating set. If A is a singleton $\{x\}$, then we abbreviate $\langle \{x\} \rangle$ by $\langle x \rangle$. Such an ideal is called *principal with generator x*.

Let K be a star-continuous Kleene algebra. We define a closed semiring $\mathsf{C}K$ as follows. The elements of $\mathsf{C}K$ will be the countably generated star-ideals of K. For any countable set of countably generated star-ideals I_n, define

$$\sum_n I_n = \langle \bigcup_n I_n \rangle.$$

This ideal is countably generated, since if A_n is countable and generates I_n for $n \geq 0$, then $\bigcup_n A_n$ is countable and generates $\sum_n I_n$. The operator $\sum$ is associative, commutative, and idempotent, since $\bigcup$ is.

For any pair of elements I, J, define

$$I \cdot J = \langle \{ab \mid a \in A,\ b \in B\} \rangle.$$

This ideal is countably generated if I and J are, since

$$\langle A \rangle \cdot \langle B \rangle = \langle \{ab \mid a \in \langle A \rangle,\ b \in \langle B \rangle\} \rangle = \langle \{ab \mid a \in A,\ b \in B\} \rangle,$$

and $\{ab \mid a \in A,\ b \in B\}$ is countable if A and B are (Exercise **??**).

The ideal $\langle 0 \rangle = \{0\}$ is included in every ideal and is thus an additive identity. It is also a multiplicative annihilator:

$$\langle 0 \rangle \cdot I = \langle \{ab \mid a \in \langle 0 \rangle,\ b \in I\} \rangle = \langle \{ab \mid a \in \{0\},\ b \in I\} \rangle = \langle 0 \rangle.$$

The ideal $\langle 1 \rangle$ is a multiplicative identity:

$$\begin{aligned} \langle 1 \rangle \cdot I &= \langle \{ab \mid a \in \langle 1 \rangle,\ b \in I\} \rangle \\ &= \langle \{ab \mid a \in \{1\},\ b \in I\} \rangle \quad \text{by Exercise **??**} \\ &= \langle I \rangle = I. \end{aligned}$$

Finally, the distributive laws hold:

$$\begin{aligned} I \cdot \sum_n J_n &= \langle \{ab \mid a \in I,\ b \in \sum_n J_n\} \rangle = \langle \{ab \mid a \in I,\ b \in \langle \bigcup_n J_n \rangle\} \rangle \\ &= \langle \{ab \mid a \in I,\ b \in \bigcup_n J_n\} \rangle \text{ by Exercise **??**} \\ &= \langle \bigcup_n \{ab \mid a \in I,\ b \in J_n\} \rangle = \langle \bigcup_n \langle \{ab \mid a \in I,\ b \in J_n\} \rangle \rangle \\ &= \langle \bigcup_n \{ab \mid a \in I,\ b \in J_n\} \rangle = \sum_n I \cdot J_n, \end{aligned}$$

and symmetrically.

Closed Semirings and S-algebras

The other half of the factorization of Conway's construction embeds an arbitrary closed semiring into an S-algebra. In comparison to the previous construction, this construction is much less interesting. We give the main construction and omit formal details.

Recall that closed semirings and S-algebras are both idempotent semirings with an infinite summation operator $\sum$ satisfying infinitary associativity, commutativity, idempotence, and distributivity laws. The only dif-

ference is that closed semirings allow only countable sums, whereas S-algebras allow arbitrary sums. Morphisms of closed semirings are the ω-continuous semiring morphisms and those of S-algebras are the continuous semiring morphisms.

To embed a given closed semiring C in an S-algebra $\mathsf{S}C$, we complete C by ideals. An *ideal* is a subset $A \subseteq C$ such that

- A is nonempty
- A is closed under countable sum
- A is closed downward under $\leq$.

Take $\mathsf{S}C$ to be the set of ideals of C with the following operations:

$$\sum_\alpha I_\alpha = \langle \bigcup_\alpha I_\alpha \rangle \qquad I \cdot J = \langle \{ab \mid a \in I,\ b \in J\}\rangle \qquad 0 = \langle 0 \rangle \qquad 1 = \langle 1 \rangle.$$

The arguments from here on are quite analogous to those of the previous section.

Lecture 7

Equational Theory of Kleene Algebra

We now turn to the equational theory of Kleene algebra. This and the next lecture will be devoted to proving that equational theory of Kleene algebra is the same as the equational theory of the regular sets under the standard interpretation. In other words, an equation $s = t$ over Σ is an element of the kernel of the standard interpretation R_Σ over $\mathsf{Reg}\,\Sigma$ iff $s = t$ is a consequence of the axioms of Kleene algebra.

The equational theory of the regular sets, or *regular events* as they are sometimes called, was first studied by Kleene [56], who posed axiomatization as an open problem. Salomaa [107] gave two complete axiomatizations of the algebra of regular events in 1966. Salomaa's axiomatization is not a universal Horn axiomatization, since it depends on rules whose validity is not preserved under substitution, thus are not sound under nonstandard interpretations. Redko [104] proved in 1964 that no finite set of equational axioms could characterize the algebra of regular events. The algebra of regular events and its axiomatization is the subject of the extensive monograph of Conway [29]; as we have seen, the bulk of Conway's treatment is infinitary.

In Lecture 5, we gave a complete infinitary equational deductive system for the algebra of regular events that is sound over all star-continuous Kleene algebras [61]. A completeness theorem for relational algebras with *, a proper subclass of Kleene algebras, was given by Ng and Tarski [92, 91]. Their axiomatization relies on the presence of a converse operator.

Schematic equational axiomatizations for the algebra of regular events, necessarily representing infinitely many equations, have been given by Krob [77] and Bloom and Ésik [14].

Salomaa's Axiomatizations

Salomaa [107] was the first to axiomatize the equational theory of the regular sets. Here is a brief account of his axiomatization.

Recall that R_Σ denotes the interpretation of regular expressions over Σ in the Kleene algebra $\mathsf{Reg}\,\Sigma$ in which $R_\Sigma(a) = \{a\}$, $a \in \Sigma$. This is called the *standard interpretation*.

Salomaa [107] presented two axiomatizations F_1 and F_2 for the equational theory of the regular sets and proved their completeness. Aanderaa [2] independently presented a system similar to Salomaa's F_1. Backhouse [7] gave an algebraic version of F_1. These systems are equational except for one rule of inference in each case that is sound under the standard interpretation R_Σ, but not sound in general over other interpretations.

Salomaa defined a regular expression to have the *empty word property* (EWP) if the regular set it denotes under R_Σ contains the null string ε. He also observed that the EWP can be characterized syntactically: a regular expression s has the EWP if either

- $s = 1$;
- $s = t^*$ for some t;
- s is a sum of regular expressions, at least one of which has the EWP; or
- s is a product of regular expressions, both of which have the EWP.

Another way to say this is that a regular expression s over Σ has the EWP iff $\varepsilon(s) = 1$, where ε denotes the unique homomorphism $\varepsilon : \mathsf{Exp}\,\Sigma \to \{0, 1\}$ such that $\varepsilon(a) = 0$, $a \in \Sigma$.

Salomaa's system F_1 contains the rule

$$\frac{u + st = t}{s^*u = t} \text{ (s does not have the EWP)} \tag{7.1}$$

where s, t, and u are regular expressions. The rule (7.1) is sound under the standard interpretation R_Σ, but not under nonstandard interpretations. The problem is that the side-condition "s does not have the EWP" is not preserved under substititution. For example, if s, t, and u are single letters, then (7.1) holds; but it does not hold after the substitution

$$s \mapsto 1 \qquad\qquad t \mapsto 1 \qquad\qquad u \mapsto 0,$$

as $0 + 1 \cdot 1 = 1$ but $1^* \cdot 0 \neq 1$. Thus (7.1) must not be interpreted as a universal Horn formula

$$u + st = t \Leftrightarrow s^* u = t.$$

Salomaa's system F_2 is somewhat different from F_1 but contains a similar nonalgebraic proviso.

In contrast, the axioms for Kleene algebra are all equations or equational implications in which the symbols are regarded as universally quantified, so substitution is allowed.

Equational Logic

By general considerations of equational logic, the axioms of Kleene algebra, along with the usual axioms for equality, instantiation, and rules for the introduction and elimination of implications, constitute a complete deductive system for the universal Horn theory of Kleene algebras (the set of universally quantified equational implications

$$s_1 = t_1 \wedge \cdots \wedge s_n = t_n \Leftrightarrow s = t \tag{7.2}$$

true in all Kleene algebras) [111, 115].

More specifically, let Δ be a set of implicitly universally quantified Horn formulas over some signature and variables X (in our application, Δ is the set of axioms of Kleene algebra). Let $d, e, \ldots$ denote equations, A a sequence of equations, σ a substitution of terms for variables, and φ Horn formula. The equational axioms are

$$x = x$$
$$x = y \Leftrightarrow y = x$$
$$x = y \Leftrightarrow y = z \Leftrightarrow x = z$$
$$x_1 = y_1 \Leftrightarrow \ldots \Leftrightarrow x_n = y_n \Leftrightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n),$$

where in the last, f is an n-ary function symbol of the signature. These are considered to be implicitly universally quantified. This set of Horn formulas is denoted E. The rules of inference are:

$$\vdash \sigma(\varphi),\ \varphi \in \Delta \cup E \qquad e \vdash e \qquad \frac{A \vdash \varphi}{A, e \vdash \varphi} \qquad \frac{A, e \vdash \varphi}{A \vdash e \Leftrightarrow \varphi} \qquad \frac{A \vdash e \qquad A \vdash e \Leftrightarrow \varphi}{A \vdash \varphi}$$

and structural rules for permuting A.

Encoding Combinatorial Arguments

To show completeness, we will show how to encode several classical combinatorial constructions of the theory of finite automata algebraically. The

first step will be to construct a transition matrix representing a finite automaton equivalent to a given regular expression. This construction is essentially implicit in the work of Kleene [56] and appears in Conway's monograph [29]. The algebraic approach to the elimination of ε-transitions appears in the work of Kuich and Salomaa [79] and Sakarovitch [106]. The results on the closure of Kleene algebras under the formation of matrices essentially go back to Conway's monograph [29] and the thesis of Backhouse [7]. It was shown in [64] how to encode algebraically two other fundamental constructions in the theory of finite automata:

- determinization of an automaton via the subset construction, and
- state minimization via equivalence modulo a Myhill-Nerode equivalence relation.

We then use the uniqueness of the minimal deterministic finite automaton to obtain completeness.

We recall some elementary consequences of the axioms of Kleene algebra proved in Exercise **??** of Homework **??**.

$$xy = yz \Leftrightarrow x^*y = yz^* \tag{7.3}$$
$$(xy)^*x = x(yx)^* \tag{7.4}$$
$$(x+y)^* = x^*(yx^*)^*. \tag{7.5}$$

These are called the *bisimulation rule*, the *sliding rule*, and the *denesting rule*, respectively.

Matrices over a Kleene Algebra

Under the natural definitions of the Kleene algebra operators $+$, $\cdot$, *, 0, and 1, the family $\mathrm{Mat}(n,K)$ of $n \times n$ matrices over a Kleene algebra K again forms a Kleene algebra. This is a standard result that holds for various classes of Kleene algebra-like structures [7, 29]. The proof for Kleene algebras in our sense is from [64].

Define $+$ and $\cdot$ on $\mathrm{Mat}(n,K)$ to be the usual operations of matrix addition and multiplication, respectively, Z_n the $n \times n$ zero matrix, and I_n the $n \times n$ identity matrix. For example, for $n = 2$,

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix} \qquad Z_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix} \qquad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Be careful: multiplication is not commutative in general, so the order of the letters is important.

The partial order $\leq$ is defined on $\mathrm{Mat}(n, K)$ by

$$A \leq B \stackrel{\text{def}}{\Longleftrightarrow} A + B = B.$$

Under these definitions, it is routine to verify that the structure

$$(\mathrm{Mat}(n, K), +, \cdot, Z_n, I_n)$$

is an idempotent semiring.

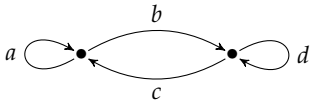
The definition of E^* for $E \in \mathrm{Mat}(n, K)$ comes from [29, 34, 79]. We first consider the case $n = 2$. This construction will later be applied inductively. If

$$E = \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

define

$$E^* \stackrel{\text{def}}{=} \begin{bmatrix} (a + bd^*c)^* & (a + bd^*c)^*bd^* \\ (d + ca^*b)^*ca^* & (d + ca^*b)^* \end{bmatrix}. \tag{7.6}$$

To understand where this definition comes from, consider a two-state finite automaton over the alphabet $\Sigma = \{a, b, c, d\}$ and transitions as defined in the following diagram.



The matrix E is the transition matrix of this automaton. For each pair of states s, t, the st^{th} entry of E^* is a regular expression describing the set of strings over the alphabet Σ going from state s to state t.

Lemma 7.1 *The matrix E^* defined in* (7.6) *satisfies the Kleene algebra axioms for *. That is, for any X,*

$$I + EE^* \leq E^* \qquad EX \leq X \Leftrightarrow E^*X \leq X \tag{7.7}$$

$$I + E^*E \leq E^* \qquad XE \leq X \Leftrightarrow XE^* \leq X. \tag{7.8}$$

Proof. We prove the left-handed star rules (7.7). The arguments for the right-hand rules (7.8) are symmetric.

The matrix inequality on the left-hand side of (7.7) reduces to the four inequalities

$$\begin{aligned} 1 + a(a + bd^*c)^* + b(d + ca^*b)^*ca^* &\leq (a + bd^*c)^* \\ a(a + bd^*c)^*bd^* + b(d + ca^*b)^* &\leq (a + bd^*c)^*bd^* \\ c(a + bd^*c)^* + d(d + ca^*b)^*ca^* &\leq (d + ca^*b)^*ca^* \\ 1 + c(a + bd^*c)^*bd^* + d(d + ca^*b)^* &\leq (d + ca^*b)^* \end{aligned}$$

in K. These simplify to

$$\begin{aligned} 1 &\leq (a + bd^*c)^* \\ a(a + bd^*c)^* &\leq (a + bd^*c)^* \\ b(d + ca^*b)^*ca^* &\leq (a + bd^*c)^* \end{aligned} \tag{7.9}$$

$$\begin{aligned} a(a + bd^*c)^*bd^* &\leq (a + bd^*c)^*bd^* \\ b(d + ca^*b)^* &\leq (a + bd^*c)^*bd^* \end{aligned} \tag{7.10}$$

$$c(a + bd^*c)^* \leq (d + ca^*b)^*ca^* \tag{7.11}$$

$$d(d + ca^*b)^*ca^* \leq (d + ca^*b)^*ca^*$$

$$\begin{aligned} 1 &\leq (d + ca^*b)^* \\ c(a + bd^*c)^*bd^* &\leq (d + ca^*b)^* \\ d(d + ca^*b)^* &\leq (d + ca^*b)^*, \end{aligned} \tag{7.12}$$

of which all but the labeled inequalities (7.9)–(7.12) are trivial. By symmetry, it suffices to show only (7.9) and (7.10). Using the denesting rule, we can rewrite these as

$$b(d^*ca^*b)^*d^*ca^* \leq (a^*bd^*c)^*a^* \qquad b(d^*ca^*b)^*d^* \leq (a^*bd^*c)^*a^*bd^*,$$

and by the sliding rule,

$$bd^*ca^*(bd^*ca^*)^* \leq a^*(bd^*ca^*)^* \qquad bd^*(ca^*bd^*)^* \leq a^*bd^*(ca^*bd^*)^*,$$

which follow directly from the axioms.

We now establish the implication on the right-hand side of (7.7). We show that this implication holds for an arbitrary column vector X of length 2; then it will also hold for any $2 \times n$ matrix X by applying this result to the columns of X separately. Let

$$X = \begin{bmatrix} x \\ y \end{bmatrix}.$$

We need to show that under the assumptions

$$ax + by \leq x \tag{7.13}$$

$$cx + dy \leq y \tag{7.14}$$

we can derive

$$(a + bd^*c)^*x + (a + bd^*c)^*bd^*y \leq x \tag{7.15}$$

$$(d + ca^*b)^*ca^*x + (d + ca^*b)^*y \leq y. \tag{7.16}$$

Note that (7.15) and (7.16) are the same inequality under the exchange $a \leftrightarrow d$, $b \leftrightarrow c$, $x \leftrightarrow y$, so by symmetry it suffices to show just (7.15). Simplifying, it suffices to show

$$(a + bd^*c)^*x \leq x \tag{7.17}$$
$$(a + bd^*c)^*bd^*y \leq x. \tag{7.18}$$

For both (7.17) and (7.18), it suffices to show

$$bd^*y + (a + bd^*c)x \leq x,$$

and for this it suffices to show (i) $ax \leq x$, (ii) $bd^*cx \leq x$, and (iii) $bd^*y \leq x$. Now (i) is immediate from the assumption (7.13), and (ii) is immediate from (iii) and (7.14). For (iii), we have $d^*y \leq y$ by (7.14) and an axiom of Kleene algebra, and then $bd^*y \leq by \leq x$ by (7.13) and monotonicity. □

To extend to matrices of arbitrary dimension, we recall the following fact established in Lecture 2:

Lemma 7.2 *In any Kleene algebra, a^*b is the unique least solution of the inequality $b + ax \leq x$, and ba^* is the unique least solution of $b + xa \leq x$.*

Lemma 7.3 *Let $E \in \mathrm{Mat}(n, K)$. There is a unique matrix $E^* \in \mathrm{Mat}(n, K)$ satisfying the Kleene algebra axioms* (7.7)*.*

Proof. Partition E into submatrices A, B, C, and D such that A and D are square.

$$E = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{7.19}$$

By the induction hypothesis, A^* and D^* exist and are unique. Again by the induction hypothesis, $A + BD^*C$ and $D + CA^*B$ exist and are unique (one must also check that these matrices are square–note that B and C are not necessarily square). We define

$$E^* \stackrel{\text{def}}{=} \begin{bmatrix} (A + BD^*C)^* & (A + BD^*C)^*BD^* \\ (D + CA^*B)^*CA^* & (D + CA^*B)^* \end{bmatrix} \tag{7.20}$$

and claim that E^* satisfies (7.7). The proof is essentially identical to the proof of Lemma 7.1. We must check that the axioms and basic properties of Kleene algebra used in the proof of Lemma 7.1 still hold when the primitive symbols of regular espressions are interpreted as matrices of various dimensions, provided there is no type mismatch in the application of the operators.

The uniqueness of E^* follows from Lemma 7.2. □

It follows from Lemma 7.3 that

Theorem 7.4 *The structure* $(\mathrm{Mat}(n,K), +, \cdot, ^*, Z_n, I_n)$ *is a Kleene algebra.*

The inductive definition (7.20) of E^* in Lemma 7.3 is independent of the partition of E chosen in (7.19). This is a consequence of Lemma 7.2, once we have established that the resulting structure is a Kleene algebra under *some* partition; cf. [29, Theorem 4, p. 27], which establishes the same result for S-algebras.

In the proof of Lemma 7.3, we needed to know that the axioms of Kleene algebra still hold when the primitive letters of regular expressions are interpreted as matrices of various shapes, possibly nonsquare, provided there is no type mismatch in the application of operators. For example, one cannot add two matrices unless they are the same shape, one cannot form the matrix product AB unless the column dimension of A is the same as the row dimension of B, and one cannot form the matrix A^* unless A is square. In general, all the axioms and basic properties of Kleene algebra still hold when the primitive letters are interpreted as possibly nonsquare matrices over a Kleene algebra, provided that there are no type conflicts in the application of the Kleene algebra operators.

For example, consider the distributive law

$$a(b+c) = ab + ac.$$

Interpreting a, b, and c as matrices over a Kleene algebra K, this equation makes sense provided the shapes of b and c are the same and the column dimension of a is the same as the row dimension of b and c. Other than that, there are no type constraints. It is easy to verify that the distributive law holds for any matrices a, b and c satisfying these constraints.

For a more involved example, consider the equational implication

$$ax = xb \Leftrightarrow a^*x = xb^*.$$

The type constraints say that a and b must be square (say $s \times s$ and $t \times t$ respectively) and that x must be $s \times t$. Under this typing, all steps of the proof of this implication involve only well-typed expressions, thus the proof remains valid.

Lecture 8

Finite Automata

Regular expressions and finite automata have traditionally been used as syntactic representations of the regular languages over an alphabet Σ. The relationship between these two formalisms forms the basis of a well-developed classical theory. Classical developments range from the more combinatorial [50, 67, 80] to the more algebraic [11, 34, 39, 106, 108].

In this lecture we define the notion of an automaton over an arbitrary Kleene algebra. In subsequent sections, we will use this formalism to derive the classical results of the theory of finite automata (equivalence with regular expressions, determinization via the subset construction, elimination of ε-transitions, and state minimization) as consequences of the axioms of Kleene algebra.

Although we consider regular expressions and automata as syntactic objects, as a matter of convenience we will be reasoning modulo the axioms of Kleene algebra. Officially, regular expressions will denote elements of $\mathcal{F}_\Sigma$, the free Kleene algebra over Σ. The Kleene algebra $\mathcal{F}_\Sigma$ is constructed by taking the quotient of the regular expressions modulo the congruence generated by the axioms of Kleene algebra. The associated canonical map assigns to each regular expression its equivalence class in $\mathcal{F}_\Sigma$. Since we will be interpreting expressions only over Kleene algebras, and all interpretations factor through $\mathcal{F}_\Sigma$ via the canonical map, this usage is without loss of generality.

We recall the following basic theorems of Kleene algebra that were proved in Exercise **??** of Homework **??**, of which we will make extensive use:

$$xy = yz \Rightarrow x^*y = yz^* \tag{8.1}$$
$$(xy)^*x = x(yx)^* \tag{8.2}$$
$$(x+y)^* = x^*(yx^*)^*. \tag{8.3}$$

These are called the *bisimulation rule*, the *sliding rule*, and the *denesting rule*, respectively.

Algebraic Definition of Finite Automata

Definition 8.1 *A* finite automaton *over K is a triple $\mathcal{A} = (u, A, v)$, where $u, v \in \{0,1\}^n$ and $A \in \mathrm{Mat}(n, K)$ for some n.*

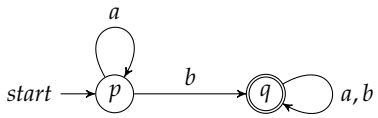
The states *are the row and column indices. The vector u determines the* start states *and the vector v determines the* final states*; a* start state *is an index i for which $u(i) = 1$ and a* final state *is one for which $v(i) = 1$. The $n \times n$ matrix A is called the* transition matrix.

The language accepted by $\mathcal{A}$ *is the element $u^T A^* v \in K$.*

For automata over $\mathcal{F}_\Sigma$, the free Kleene algebra on free generators Σ, this definition is essentially equivalent to the classical combinatorial definition of an automaton over the alphabet Σ as found in [80, 50]. A similar definition can be found in [29].

Example 8.2 *Consider the two-state automaton in the sense of [50, 80] with states $\{p, q\}$, start state p, final state q, and transitions*

$$p \xrightarrow{a} p \qquad q \xrightarrow{a} q \qquad p \xrightarrow{b} q \qquad q \xrightarrow{b} q.$$



Classically, this automaton accepts the set of strings over $\Sigma = \{a, b\}$ containing at least one occurrence of b. In our formalism, this automaton is specified by the triple

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a & b \\ 0 & a+b \end{bmatrix}^* \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a^* & a^*b(a+b)^* \\ 0 & (a+b)^* \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = a^*b(a+b)^*. \tag{8.4}$$

The language in Reg Σ *accepted by this automaton is the image under* R_Σ *of the expression* (8.4).

Definition 8.3 *Let* $\mathcal{A} = (u, A, v)$ *be an automaton over* $\mathcal{F}_\Sigma$*, the free Kleene algebra on free generators* Σ*. The automaton* $\mathcal{A}$ *is said to be* simple *if* A *can be expressed as a sum*

$$A = J + \sum_{a \in \Sigma} a \cdot A_a \tag{8.5}$$

where J *and the* A_a *are 0-1 matrices. In addition,* $\mathcal{A}$ *is said to be* ε-free *if* J *is the zero matrix. Finally,* $\mathcal{A}$ *is said to be* deterministic *if it is simple and* ε*-free, and* u *and all rows of* A_a *have exactly one* 1*.*

In Definition 8.3, ε refers to the null string. The matrix A_a in (8.5) corresponds to the adjacency matrix of the graph consisting of edges labeled a in the combinatorial model of automata [50, 80] or the image of a under a linear representation map in the algebraic approach of [11, 108]. An automaton is deterministic according to this definition iff it is deterministic in the sense of [50, 80].

The automaton of Example 8.2 is simple, ε-free, and deterministic.

Completeness

In this section we prove the completeness of the axioms of Kleene algebra for the algebra of regular events. Another way of stating this is that Reg Σ is isomorphic to $\mathcal{F}_\Sigma$, the free Kleene algebra on free generators Σ, and the standard interpretation $R_\Sigma : \mathcal{F}_\Sigma \to$ Reg Σ collapses to an isomorphism of Kleene algebras.

The first lemma asserts that Kleene's representation theorem [11, 34, 56, 106] is a theorem of Kleene algebra.

Lemma 8.4 *For every regular expression* α *over* Σ *(or more accurately, its image in* $\mathcal{F}_\Sigma$ *under the canonical map), there is a simple automaton* (u, A, v) *over* $\mathcal{F}_\Sigma$ *such that*

$$\alpha = u^T A^* v.$$

Proof. The proof is by induction on the structure of the regular expression. We essentially implement the combinatorial constructions as found for example in [50, 80]. The ideas behind this construction are well known and can be found for example in [29].

For $a \in \Sigma$, the automaton

$$\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right)$$

suffices, since

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix}^* \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = a.$$

For the expression $\alpha + \beta$, let $\mathcal{A} = (u, A, v)$ and $\mathcal{B} = (s, B, t)$ be automata such that

$$\alpha = u^T A^* v \qquad\qquad \beta = s^T B^* t.$$

Consider the automaton

$$\left(\begin{bmatrix} u \\ s \end{bmatrix}, \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}, \begin{bmatrix} v \\ t \end{bmatrix} \right).$$

This construction corresponds to the combinatorial construction of forming the disjoint union of the two sets of states, taking the start states to be the union of the start states of $\mathcal{A}$ and $\mathcal{B}$, and the final states to be the union of the final states of $\mathcal{A}$ and $\mathcal{B}$. Then

$$\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}^* = \begin{bmatrix} A^* & 0 \\ 0 & B^* \end{bmatrix},$$

and

$$\begin{bmatrix} u^T & s^T \end{bmatrix} \cdot \begin{bmatrix} A^* & 0 \\ 0 & B^* \end{bmatrix} \cdot \begin{bmatrix} v \\ t \end{bmatrix} = u^T A^* v + s^T B^* t = \alpha + \beta.$$

For the expression $\alpha\beta$, let $\mathcal{A} = (u, A, v)$ and $\mathcal{B} = (s, B, t)$ be automata such that

$$\alpha = u^T A^* v \qquad\qquad \beta = s^T B^* t.$$

Consider the automaton

$$\left(\begin{bmatrix} u \\ 0 \end{bmatrix}, \begin{bmatrix} A & vs^T \\ 0 & B \end{bmatrix}, \begin{bmatrix} 0 \\ t \end{bmatrix} \right)$$

This construction corresponds to the combinatorial construction of forming the disjoint union of the two sets of states, taking the start states to be the start states of $\mathcal{A}$, the final states to be the final states of $\mathcal{B}$, and connecting the final states of $\mathcal{A}$ with the start states of $\mathcal{B}$ by ε-transitions (this is the purpose of the vs^T in the upper right corner of the matrix). Then

$$\begin{bmatrix} A & vs^T \\ 0 & B \end{bmatrix}^* = \begin{bmatrix} A^* & A^* vs^T B^* \\ 0 & B^* \end{bmatrix},$$

and

$$\begin{bmatrix} u^T & 0 \end{bmatrix} \cdot \begin{bmatrix} A^* & A^* v s^T B^* \\ 0 & B^* \end{bmatrix} \cdot \begin{bmatrix} 0 \\ t \end{bmatrix} = u^T A^* v s^T B^* t = \alpha\beta.$$

For the expression α^*, let $\mathcal{A} = (u, A, v)$ be an automaton such that $\alpha = u^T A^* v$. We first produce an automaton equivalent to the expression $\alpha\alpha^*$. Consider the automaton

$$(u, A + vu^T, v).$$

This construction corresponds to the combinatorial construction of adding ε-transitions from the final states of $\mathcal{A}$ back to the start states. Using (8.3) and (8.2),

$$u^T(A + vu^T)^* v = u^T A^*(vu^T A^*)^* v = u^T A^* v(u^T A^* v)^* = \alpha\alpha^*.$$

Once we have an automaton for $\alpha\alpha^*$, we can get an automaton for $\alpha^* = 1 + \alpha\alpha^*$ by the construction for $+$ given above, using a trivial one-state automaton for 1. □

Now we get rid of ε-transitions. This construction is also folklore and can be found for example in [79, 106]. This construction models algebraically the combinatorial idea of computing the *ε-closure* of a state; see [50, 80].

Lemma 8.5 *For every simple automaton (u, A, v) over $\mathcal{F}_\Sigma$, there is a simple ε-free automaton (s, B, t) such that*

$$u^T A^* v = s^T B^* t.$$

Proof. By Definition 8.3, the matrix A can be written as a sum $A = J + A'$ where J is a 0-1 matrix and A' is ε-free. Then

$$u^T A^* v = u^T (A' + J)^* v = u^T J^* (A' J^*)^* v$$

by (8.3), so we can take

$$s^T = u^T J^* \qquad B = A' J^* \qquad t = v.$$

Note that J^* is 0-1 and therefore B is ε-free. □

The next step in the proof will be to give algebraic analogs of the determinization of finite automata via the subset construction and the minimization of deterministic automata via the collapsing of equivalent states under a Myhill-Nerode equivalence relation. We will do this next time.

Lecture 9

Completeness

Here we continue the program begun in the previous lecture to show the completeness of Kleene algebra for the equational theory of the regular sets.

As in the previous two lectures, we will make extensive use of the bisimulation, sliding, and denesting rules:

$$xy = yz \Rightarrow x^*y = yz^* \tag{9.1}$$
$$(xy)^*x = x(yx)^* \tag{9.2}$$
$$(x+y)^* = x^*(yx^*)^*. \tag{9.3}$$

The following two results are algebraic analogs of the determinization of automata via the subset construction and the minimization of deterministic automata via the collapsing of equivalent states under a Myhill-Nerode equivalence relation. The original combinatorial versions of these results are due to Rabin and Scott [103] and Myhill and Nerode [89, 90] respectively; see [50, 67, 80] for an elementary exposition. The construction here is from [64].

Lemma 9.1 *For every simple ε-free automaton (u, A, v) over $\mathcal{F}_\Sigma$, there is an equivalent deterministic automaton $(\widehat{u}, \widehat{A}, \widehat{v})$ over $\mathcal{F}_\Sigma$. That is, $u^T A^* v = \widehat{u}^T \widehat{A}^* \widehat{v}$.*

Proof. We model the subset construction algebraically. Let (u, A, v) be a simple ε-free automaton with states Q. Since (u, A, v) is simple, A can be expressed as a sum

$$A = \sum_{a \in \Sigma} a \cdot A_a, \tag{9.4}$$

where each A_a is a 0-1 matrix.

Let 2^Q denote the power set of Q. We identify elements of 2^Q with their characteristic vectors in $\{0,1\}^n$. For each $s \in 2^Q$, let e_s be the $2^Q \times 1$ vector with 1 in position s and 0 elsewhere.

Let X be the $2^Q \times Q$ matrix whose s^{th} row is s^T; that is,

$$e_s^T X = s^T. \tag{9.5}$$

For each $a \in \Sigma$, let $\widehat{A}_a$ be the $2^Q \times 2^Q$ matrix whose s^{th} row is $e_{s^T A_a}$; in other words,

$$e_s^T \widehat{A}_a = e_{s^T A_a}^T. \tag{9.6}$$

Let

$$\widehat{u} = e_u \qquad \widehat{A} = \sum_{a \in \Sigma} a \cdot \widehat{A}_a \qquad \widehat{v} = Xv. \tag{9.7}$$

The automaton $(\widehat{u}, \widehat{A}, \widehat{v})$ is simple and deterministic.

The relationship between A and $\widehat{A}$ is expressed succinctly by the equation

$$XA = \widehat{A}X. \tag{9.8}$$

Intuitively, this says that the actions of the two automata in the two spaces K^Q and K^{2^Q} commute with the projection X. To prove (9.8), observe that for any $s \in 2^Q$, by (9.5) and (9.6) we have

$$s^T A_a = e_{s^T A_a}^T X = e_s^T \widehat{A}_a X,$$

thus by (9.5), (9.4), and (9.7) we have

$$e_s^T XA = s^T A = \sum_{a \in \Sigma} a \cdot s^T A_a = \sum_{a \in \Sigma} a \cdot e_s^T \widehat{A}_a X = e_s^T \widehat{A} X.$$

By (9.8) and (9.1) (or rather its extension to nonsquare matrices as described in Lecture 7), we have $XA^* = \widehat{A}^* X$. The theorem now follows: using (9.7) and (9.5),

$$\widehat{u}^T \widehat{A}^* \widehat{v} = e_u^T \widehat{A}^* Xv = e_u^T XA^* v = u^T A^* v.$$

□

Lemma 9.2 *Let (u, A, v) be a simple deterministic automaton, and let $(\overline{u}, \overline{A}, \overline{v})$ be the equivalent minimal deterministic automaton obtained from the classical state minimization procedure. Then*

$$u^T A^* v = \overline{u}^T \overline{A}^* \overline{v}.$$

Proof. In the combinatorial approach, the unique minimal automaton is obtained as a quotient by a Myhill-Nerode equivalence relation after removing inaccessible states. We simulate this construction algebraically.

Let Q denote the set of states of (u, A, v). For $q \in Q$, let $e_q \in \{0,1\}^Q$ denote the vector with 1 in position q and 0 elsewhere. Since (u, A, v) is simple, A can be written as a sum

$$A = \sum_{a \in \Sigma} a \cdot A_a,$$

where the A_a are 0-1 matrices. For each $a \in \Sigma$ and $p \in Q$, let $\delta(p, a)$ be the unique state in Q such that the p^{th} row of A_a is $e^T_{\delta(p,a)}$; that is,

$$e_p^T A_a = e^T_{\delta(p,a)}.$$

The state $\delta(p, a)$ exists and is unique since the automaton is deterministic.

First we show how to get rid of inaccessible states. A state q is *accessible* if

$$u^T A^* e_q \neq 0,$$

otherwise it is *inaccessible*. Let R be the set of accessible states and let $U = Q - R$ be the set of inaccessible states. Partition A into four submatrices A_{RR}, A_{RU}, A_{UR}, and A_{UU} such that for $S, T \in \{R, U\}$, A_{ST} is the $S \times T$ submatrix of A. Then A_{RU} is the zero matrix, otherwise a state in U would be accessible. Similarly, partition the vectors u and v into u_R, u_U, v_R and v_U. The vector u_U is the zero vector, otherwise a state in U would be accessible. We have

$$u^T A^* v = \begin{bmatrix} u_R^T & 0 \end{bmatrix} \cdot \begin{bmatrix} A_{RR} & 0 \\ A_{UR} & A_{UU} \end{bmatrix}^* \cdot \begin{bmatrix} v_R \\ v_U \end{bmatrix}$$

$$= \begin{bmatrix} u_R^T & 0 \end{bmatrix} \cdot \begin{bmatrix} A_{RR}^* & 0 \\ A_{UU}^* A_{UR} A_{RR}^* & A_{UU}^* \end{bmatrix} \cdot \begin{bmatrix} v_R \\ v_U \end{bmatrix} = u_R^T A_{RR}^* v_R.$$

Moreover, the automaton (u_R, A_{RR}, v_R) is simple and deterministic, and all states are accessible.

Assume now that (u, A, v) is simple and deterministic and all states are accessible. An equivalence relation $\equiv$ on Q is called *Myhill-Nerode* if

$$p \equiv q \Rightarrow \delta(p, a) \equiv \delta(q, a),\ a \in \Sigma \qquad\qquad p \equiv q \Rightarrow e_p^T v = e_q^T v. \tag{9.9}$$

In combinatorial terms, $\equiv$ is *Myhill-Nerode* if it is respected by the action of the automaton under any input symbol $a \in \Sigma$, and the set of final states is a union of $\equiv$-classes.

Let $\equiv$ be any Myhill-Nerode equivalence relation, and let

$$[p] \stackrel{\text{def}}{=} \{q \in Q \mid q \equiv p\} \qquad\qquad Q/\!\equiv \stackrel{\text{def}}{=} \{[p] \mid p \in Q\}.$$

For $[p] \in Q/\!\equiv$, let $e_{[p]} \in \{0,1\}^{Q/\equiv}$ denote the vector with 1 in position $[p]$ and 0 elsewhere. Let Y be the $Q \times Q/\!\equiv$ matrix whose $[p]^{\text{th}}$ column is the characteristic vector of $[p]$; that is,

$$e_p^T Y = e_{[p]}^T.$$

For each $a \in \Sigma$, let $\overline{A}_a$ be the $Q/\!\equiv \times Q/\!\equiv$ matrix whose $[p]^{\text{th}}$ row is $e_{[\delta(p,a)]}$; that is,

$$e_{[p]}^T \overline{A}_a = e_{[\delta(p,a)]}^T.$$

The matrix $\overline{A}_a$ is well-defined by the left-hand implication of (9.9). Let

$$\overline{A} = \sum_{a \in \Sigma} a \cdot \overline{A}_a \qquad\qquad \overline{u}^T = u^T Y.$$

Also, let $\overline{v} \in \{0,1\}^{Q/\equiv}$ be the vector such that

$$e_{[p]}^T \overline{v} = e_p^T v.$$

The vector $\overline{v}$ is well-defined by the right-hand implication of (9.9). Note also that

$$e_p^T Y \overline{v} = e_{[p]}^T \overline{v} = e_p^T v,$$

therefore $Y\overline{v} = v$. The automaton $(\overline{u}, \overline{A}, \overline{v})$ is simple and deterministic.

As in the proof of Lemma 9.1, the actions of A and $\overline{A}$ commute with the linear projection Y:

$$AY = Y\overline{A}. \tag{9.10}$$

To prove (9.10), observe that for any $p \in Q$,

$$e_p^T AY = \sum_{a \in \Sigma} a \cdot e_p^T A_a Y = \sum_{a \in \Sigma} a \cdot e_{\delta(p,a)}^T Y = \sum_{a \in \Sigma} a \cdot e_{[\delta(p,a)]}^T = \sum_{a \in \Sigma} a \cdot e_{[p]}^T \overline{A}_a = \sum_{a \in \Sigma} a \cdot e_p^T Y \overline{A}_a = e_p^T Y \overline{A}.$$

Now by (9.10) and (9.1), we have $A^* Y = Y\overline{A}^*$, therefore

$$\overline{u}^T \overline{A}^* \overline{v} = u^T Y \overline{A}^* \overline{v} = u^T A^* Y \overline{v} = u^T A^* v.$$

□

Lemma 9.3 *Let p be an invertible element of a Kleene algebra with inverse p^{-1}. Then*

$$p^{-1}x^*p = (p^{-1}xp)^*.$$

Proof. We have

$$x^*p = (pp^{-1}x)^*p = p(p^{-1}xp)^*$$

by the sliding rule (9.2). The result follows by multiplying on the left by p^{-1}. □

Theorem 9.4 (Completeness) *Let α and β be two regular expressions over Σ denoting the same regular set. Then $\alpha = \beta$ is a theorem of Kleene algebra.*

Proof. Let $\mathcal{A} = (s, A, t)$ and $\mathcal{B} = (u, B, v)$ be minimal deterministic finite automata over $\mathcal{F}_\Sigma$ such that

$$R_\Sigma(\alpha) = R_\Sigma(s^TA^*t) \qquad R_\Sigma(\beta) = R_\Sigma(u^TB^*v).$$

By Lemmas 8.4, 9.1, and 9.2, we have

$$\alpha = s^TA^*t \qquad \beta = u^TB^*v$$

as theorems of Kleene algebra. Since $R_\Sigma(\alpha) = R_\Sigma(\beta)$, by the uniqueness of minimal automata, $\mathcal{A}$ and $\mathcal{B}$ are isomorphic. Let P be a permutation matrix giving this isomorphism. Then

$$A = P^TBP \qquad s = P^Tu \qquad t = P^Tv.$$

Using Lemma 9.3, we have

$$\alpha = s^TA^*t = (P^Tu)^T(P^TBP)^*(P^Tv) = u^TP(P^TBP)^*P^Tv = u^TPP^TB^*PP^Tv = u^TB^*v = \beta.$$

□

Lecture 10

Complexity of the Equational Theory

In the next few lectures we will develop some efficient algorithms for deciding the equational theories of KA and KAT. *Efficient* must be taken with a grain of salt: both theories are PSPACE-complete, which means there are no worst-case polynomial-time algorithms unless P = PSPACE. Nevertheless, there are algorithms that have been implemented and work well in practice [96, 97]. These algorithms are useful in the automation of systems based on KA and KAT.

We start with a classic complexity result showing that the equational theory of Kleene algebra is PSPACE-complete.

Theorem 10.1 (Stockmeyer and Meyer [113]) *Let $R : \mathsf{Exp}\,\Sigma \to \mathsf{Reg}\,\Sigma$ be the standard interpretation of regular expressions as regular subsets of Σ^*. The problem of deciding whether $R(s) = R(t)$ for given $s, t \in \mathsf{Exp}\,\Sigma$ is PSPACE-complete. The problem is PSPACE-hard even if one of s, t is a trivial expression representing Σ^*.*

Proof. To show that the problem is in PSPACE, we first convert the expressions to nondeterministic finite automata M and N using the construction of Kleene's theorem and ask whether the two automata accept the same set. The conversion of Kleene's theorem is linear. We will actually give a nondeterministic PSPACE algorithm for determining whether the two automata are different—that is, whether there exists a string accepted

by one and not by the other. The nondeterminism will be used to guess the string in the symmetric difference of the two sets. By Savitch's theorem [109], nondeterministic and deterministic PSPACE are equivalent, so this is sufficient.

The algorithm starts with a pebble on each start state. It then guesses an input string x symbol by symbol, moving pebbles on the states of M and N to mark all states reachable from the start state under the string guessed so far. We do not need to remember the guessed string, just the states that M and N could currently be in. The procedure accepts if it ever sees a pebble configuration in which an accept state of M is occupied by a pebble, but no accept state of N is occupied by a pebble, which indicates that the string x guessed so far is accepted by M but not by N, or vice versa. Since the pebble configurations can be represented in polynomial space, this is a nondeterministic PSPACE computation.

We remark that in general the procedure may have to run for an exponential length of time before the acceptance situation described above occurs, if at all. This is because the shortest string *not* accepted by one of the machines may be exponential in the size of the machine. This may be surprising in light of the fact that the shortest string accepted is linear in the size of the machine, if any string at all is accepted. But consider for example an automaton over a single letter alphabet with a start state going to n disjoint loops with pairwise relatively prime lengths $p_1, \ldots, p_n$. Make every state an accept state except for the one in each loop farthest from the start state. There are $1 + p_1 + p_2 + \cdots + p_n$ states, but the shortest string not accepted is of length $p_1 p_2 \cdots p_n$, which is exponential in $1 + p_1 + p_2 + \cdots + p_n$.

To show that the problem is hard for PSPACE, we will encode computation histories of an arbitrary one-tape polynomial-space-bounded deterministic Turing machine. Let T be such a machine, and let n^c be its space bound. Assume without loss of generality that T always erases its tape and moves its tape head all the way to the left end of the tape before accepting. Given an input x, $|x| = n$, we build a regular expression s of length $O(n^c)$ over a fixed finite alphabet $\Delta \cup \{\#\}$ such that $R(s)$ consists of all strings that are *not* valid accepting computation histories of T on input x. Then $R(s) = (\Delta \cup \{\#\})^*$ iff there is no such valid accepting computation history, which occurs iff T does not accept x. Thus we are reducing the set accepted by T to the set $\{s \mid R(s) \neq (\Delta \cup \{\#\})^*\}$. The alphabet Δ, the constant c, and the constant of proportionality in the $O(n^c)$ may depend on T but are independent of x.

Formally, an *accepting computation history* of T on input x, $|x| = n$, is a string of the form

$$\#\alpha_0\#\alpha_1\#\alpha_2\#\cdots\#\alpha_{m-1}\#\alpha_m\# \tag{10.1}$$

where each α_i is a string of length n^c over Δ encoding a configuration of T on input x, such that:

1. α_0 represents the start configuration of T on x;
2. α_m represents the accept configuration of T on x;
3. each α_{i+1} follows from α_i according to the transition rules of T.

If a string is *not* an accepting computation history, then either it is not of the form (10.1), or one of the three conditions 1–3 fails. The regular expression s will thus consist of a sum of four subexpressions s_0, s_1, s_2, and s_3, where s_0 describes those strings in $(\Delta \cup \{\#\})^*$ that are not of the form (10.1), and s_i represents the set of strings that violate condition i, $1 \leq i \leq 3$.

To be of the form (10.1), a string must be a member of the regular set denoted by $(\#\Delta^{n^c})^*\#$, plus a few other simple conditions on the format: exactly one state of T in each configuration, each configuration begins and ends with endmarkers, etc. A deterministic finite automaton M requires $O(n^c)$ states to check these conditions. Checking conditions 1 and 2 involves just checking whether the input begins and ends with a certain fixed strings of length n^c. These strings can be encoded in the finite control of M.

Finally, to check condition 3, observe that whether α_{i+1} is the correct successor configuration of α_{i+1} according to the transition rules of T depends on a finite set of local conditions involving the $j-1^{\text{st}}$, j^{th}, and $j+1^{\text{st}}$ symbols α_i and the j^{th} symbol of α_{i+1}. The condition 3 holds iff these local conditions are met for all $1 \leq j \leq n^c$ and all $0 \leq i < m$. The local conditions describe all the valid transitions and depend only on the description of T. To check that 3 is violated, M scans across the input and at some point guesses nondeterministically where the violation occurs. It remembers the next three symbols in its finite control, skips over the next n^c symbols, and accepts if the next symbol is not correct. □

Lecture 11

Extra

In this lecture we show:

- Conversion of regular expressions without sharing of common subexpressions to finite automata is linear.
- Conversion of regular expressions with sharing to regular expressions without sharing or to finite automata is exponential.
- Conversion of finite automata to regular expressions with sharing is at most cubic.
- The equational theory of Kleene algebra is PSPACE-complete.
- Deciding whether two finite automata accept the same set is PSPACE-complete.
- It is even PSPACE-hard to decide whether one automaton accepts all strings or whether a regular expression over Σ is equivalent to Σ^*.
- Deciding whether two expressions with sharing are equivalent, or whether one expression with sharing is equivalent to Σ^*, is exponential-space complete.

Lecture 12

Equational Implications

In many applications of Kleene algebra, one needs to reason in the presence of extra conditions. For example, the fact that two atomic actions cannot affect each other and can be done in either order might be represented by a commutativity condition $pq = qp$. The fact that an atomic program p does not affect the truth value of a test b might be represented similarly as $pb = bp$. It is therefore of interest to study universally quantified equational implications or *Horn formulas*, which are formulas of the form

$$s_1 = t_1 \wedge \cdots \wedge s_n = t_n \quad \Rightarrow \quad s = t.$$

In general, the universal Horn theory is much more complex than the equational theory. For unrestricted premises, the universal Horn theory of Kleene algebras is Σ_1^0-complete (r.e. complete) and that of star-continuous Kleene algebras is Π_1^1-complete [69]. The universal Horn theory of relational algebras is also Π_1^1-complete [44].

However, there is an interesting and useful subclass of Horn formulas in which the problem is no less tractable than the equational theory and that actually arises quite often in practice. The subclass consists of Horn formulas in which all premises are of the form $r = 0$. For example, the premise $pb = bp$ mentioned above is equivalent (under Boolean algebra axioms for tests b) to $bp\overline{b} + \overline{b}pb = 0$. Formulas of this form are sufficient to

encode Hoare logic, a classical logic for program correctness. We call Horn formulas satisfying this restriction *Hoare formulas* after C. A. R. Hoare, the inventor of Hoare logic. Thus a *Hoare formula* is a Horn formula of the form

$$r_1 = 0 \wedge \cdots \wedge r_n = 0 \quad \Rightarrow \quad s = t. \tag{12.1}$$

In this lecture we prove a result of Cohen [23] that shows how to eliminate premises of the form $r = 0$. Specifically, we show that any Hoare formula reduces efficiently to a single equation that is valid iff the original Hoare formula was. The Hoare formula and the resulting equation are not equivalent under all interpretations, but one is valid iff the other is.

Let $p, q, r \in \mathsf{Exp}\,\Sigma$. Let u be the *universal expression* $(a_1 + \cdots + a_m)^*$, where $\Sigma = \{a_1, \ldots, a_m\}$. Under the standard interpretation R_Σ in $\mathsf{Reg}\,\Sigma$, the term u represents the set of all strings over Σ. It is not difficult to show that for any $x \in \mathsf{Exp}\,\Sigma$, $x \leq u$ is a theorem of Kleene algebra.

Let us write $\mathsf{KA} \vDash \varphi$ or $\mathsf{KA}^* \vDash \varphi$ to indicate that a Horn formula φ is a theorem of Kleene algebra or star-continuous Kleene algebra, respectively. If φ is an equation, these two notions coincide (Theorem 17.4), in which case we write $\vDash \varphi$.

First we observe that the conjunction of the premises $r_1 = 0 \wedge \cdots \wedge r_n = 0$ in (12.1) is equivalent to the single premise $r_1 + \cdots + r_n = 0$, so we can without loss of generality restrict our attention to Hoare formulas of the form

$$r = 0 \quad \Rightarrow \quad s = t. \tag{12.2}$$

Theorem 12.1 (Cohen [23]) *The following are equivalent:*

(i) $\mathsf{KA} \models r = 0 \Rightarrow p = q$

(ii) $\mathsf{KA}^* \models r = 0 \Rightarrow p = q$

(iii) $\models p + uru = q + uru.$

In (iii), we do not need to write KA or KA^*, since we have shown that the equational theories coincide (Theorem 17.4). Note that the equivalence of (i) and (ii) does not follow immediately from this result, since they are not equations but equational implications.

Proof. We first define a congruence on regular expressions in $\mathsf{Exp}\,\Sigma$. For $s, t \in \mathsf{Exp}\,\Sigma$, define

$$s \equiv t \quad \overset{\text{def}}{\iff} \quad \vDash s + uru = t + uru.$$

The relation $\equiv$ is an equivalence relation. We show that it is a star-continuous Kleene algebra congruence.

If $s = t$ is a theorem of Kleene algebra, then $s \equiv t$, since $\vDash s = t$ implies $\vDash s + uru = t + uru$.

To show $\equiv$ is a congruence with respect to $+$, we need to show that $s \equiv t$ implies $s + w \equiv t + w$. But this says only that $\vDash s + uru = t + uru$ implies $\vDash s + w + uru = t + w + uru$, which is immediately apparent.

To show $\equiv$ is a congruence with respect to $\cdot$, we need to show that $s \equiv t$ implies $sw \equiv tw$ and $ws \equiv wt$. We establish the former; the latter follows by symmetry.

$$
\begin{aligned}
&\vDash s + uru = t + uru \\
&\quad\Rightarrow\quad \vDash (s + uru)w = (t + uru)w \\
&\quad\Rightarrow\quad \vDash sw + uruw = tw + uruw \\
&\quad\Rightarrow\quad \vDash sw + uruw + uru = tw + uruw + uru \\
&\quad\Rightarrow\quad \vDash sw + ur(uw + u) = tw + ur(uw + u) \\
&\quad\Rightarrow\quad \vDash sw + uru = tw + uru.
\end{aligned}
$$

The last implication follows from the fact that $uw \leq u$, so $uw + u = u$.

To show $\equiv$ is a congruence with respect to *, we need to show that $s \equiv t$ implies $s^* \equiv t^*$.

$$
\begin{aligned}
&\vDash s + uru = t + uru \\
&\quad\Rightarrow\quad \vDash (s + uru)^* = (t + uru)^* \\
&\quad\Rightarrow\quad \vDash s^*(urus^*)^* = t^*(urut^*)^* \\
&\quad\Rightarrow\quad \vDash s^*(1 + urus^*(urus^*)^*) = t^*(1 + urut^*(urut^*)^*) \\
&\quad\Rightarrow\quad \vDash s^* + s^*urus^*(urus^*)^* = t^* + t^*urut^*(urut^*)^* \\
&\quad\Rightarrow\quad \vDash s^* + s^*urus^*(urus^*)^* + uru = t^* + t^*urut^*(urut^*)^* + uru \\
&\quad\Rightarrow\quad \vDash s^* + uru = t^* + uru.
\end{aligned}
$$

Finally, to show that $\equiv$ respects star-continuity condition, we need only show that if $st^nv \leqq y$ for all n, then $st^*v \leqq y$, where $p \leqq q$ is an abbreviation for $p + q \equiv q$.

$$
\begin{aligned}
&\vDash (st^nv + y) + uru = y + uru \text{ for all } n \\
&\quad\Rightarrow\quad \vDash st^nv + (y + uru) = y + uru \text{ for all } n \\
&\quad\Rightarrow\quad \vDash st^*v + (y + uru) = y + uru \qquad (12.3) \\
&\quad\Rightarrow\quad \vDash (st^*v + y) + uru = y + uru.
\end{aligned}
$$

The crucial step (12.3) follows from the fact that if $st^nv \leq y + uru$ for all n in all star-continuous Kleene algebras, then $st^*v \leq y + uru$ in all star-continuous Kleene algebras.

Since $\equiv$ is a KA^* congruence on $\mathsf{Exp}\,\Sigma$, we can form the quotient $\mathsf{Exp}\,\Sigma/\equiv$ and canonical interpretation $s \mapsto [s]$, where $[s]$ denotes the

$\equiv$-congruence class of s, and this structure is a star-continuous Kleene algebra. The equation $r = 0$ is satisfied under this interpretation, since

$$\models r + uru \quad = \quad uru \quad = \quad 0 + uru,$$

so $r \equiv 0$.

Now we are ready to prove the equivalence of the three conditions in the statement of the theorem.

(i) $\Rightarrow$ (ii) Any formula true in all Kleene algebras is certainly true in all star-continuous Kleene algebras.

(ii) $\Rightarrow$ (iii) If $\mathsf{KA}^* \models r = 0 \Rightarrow p = q$, then since $\mathsf{Exp}\,\Sigma/\equiv$ is a star-continuous Kleene algebra and $\mathsf{Exp}\,\Sigma/\equiv$, $[\,] \models r = 0$, we have $\mathsf{Exp}\,\Sigma/\equiv$, $[\,] \models p = q$. By definition, $p \equiv q$, which is what we wanted to show.

(iii) $\Rightarrow$ (i) Suppose $\models p + uru = q + uru$. Let K be an arbitrary Kleene algebra and let I be an arbitrary interpretation over K such that $K, I \models r = 0$. Then $K, I \models p = p + uru = q + uru = q$. Since K and I were arbitrary, $\mathsf{KA} \models r = 0 \Rightarrow p = q$. □

Lecture 13

Parikh's Theorem in Commutative KA

Parikh's theorem [94] says that every context-free language is "letter-equivalent" to a regular set; formally, the commutative image of any context-free language is also the commutative image of some regular set, where the *commutative image* of a set A of strings over the finite alphabet $\{a_1, \ldots, a_k\}$ is the set of k-tuples

$$\{(\#a_1(x), \ldots, \#a_k(x)) \in \mathbb{N}^k \mid x \in A\} \quad \subseteq \quad \mathbb{N}^k,$$

where $\#a_i(x)$ is the number of occurrences of a_i in x. The k-tuples in $\mathbb{N}^k$ are often called *Parikh vectors*. For example, the context-free language $\{a^n b^n \mid n \geq 0\}$ is letter-equivalent to the regular set $(ab)^*$; these two sets have a common commutative image $\{(n, n) \mid n \geq 0\}$.

The usual combinatorial proofs of Parikh's theorem (see for example [67]) involve an induction on parse trees of context-free grammars. In this lecture and the next we prove the following general theorem of commutative Kleene algebra, of which Parikh's theorem is a special case:

Theorem 13.1 *Every system of inequalities*

$$f_i(x_1, \ldots, x_n) \quad \leq \quad x_i, \quad 1 \leq i \leq n, \tag{13.1}$$

where the f_i are polynomials in $K[x_1, \ldots, x_n]$ over a commutative Kleene algebra K, has a unique least solution in K^n; moreover, the components of the solution are given by polynomials in the coefficients of the f_i.

We might take the statement of Theorem 14.1 as a definition of *algebraic closure* in KA, in which case the theorem says that any commutative Kleene algebra is algebraically closed.

Pilling [95] proves Theorem 14.1 in the special case of the commutative KA $\mathsf{Reg}\,\mathbb{N}^k$, the algebra of regular sets of Parikh vectors, and argues that this is the essential content of Parikh's theorem. Indeed, context-free grammars are just systems of set inequalities, and the context-free languages they generate are the minimal solutions. For example, the context-free grammar $S \to aSb \mid \varepsilon$ is essentially the system consisting of the single inequality $axb + 1 \leq x$ whose least solution in $2^{\{a,b\}^*}$ is the context-free language $\{a^n b^n \mid n \geq 0\}$. Under the assumption of commutativity, the inequality can be rewritten $abx + 1 \leq x$, whose least solution is the regular set $(ab)^*$. For a somewhat more difficult example, the context-free grammar $S \to [S] \mid SS \mid \varepsilon$ generating the set of balanced strings of parentheses is essentially the system consisting of the single inequality $[x] + xx + 1 \leq x$. Under the assumption of commutativity, the inequality can be rewritten $[]x + x^2 + 1 \leq x$, whose least solution is the regular set $([])^*$.

Using Theorem **??** of Lecture **??**, one can generalize Pilling's proof to any star-continuous KA. However, the proof makes essential use of various infinitary properties such as the continuity of regular operators and the fact that a^* is the supremum of the a^n, $n \geq 0$.

Kuich [78] also gives a generalization of Parikh's theorem that holds for any commutative idempotent ω-continuous semiring. Kuich's result implies Pilling's, since $\mathsf{Reg}\,\mathbb{N}^k$ is embedded in the commutative idempotent ω-continuous semiring $2^{\mathbb{N}^k}$. Conversely, since every commutative idempotent ω-continuous semiring is a commutative KA under the usual definition of the * operator

$$a^* \quad = \quad \sum_{n \geq 0} a^n,$$

Pilling's result, suitably generalized to star-continuous Kleene algebras, would imply Kuich's. But again, these proofs depend on the strong infinitary properties of star-continuous algebras.

Our result is a generalization of these results in that it holds in all commutative KAs. The main difference here is that KA as defined in [64] has a *finitary* algebraic axiomatization consisting of finitely many equations and equational implications. Thus one might say that we are replacing the *analytic* arguments of Pilling and Kuich with *algebraic* arguments. The fact that we cannot argue combinatorially in the model $\mathsf{Reg}\,\mathbb{N}^k$ or use the infinitary properties of star-continuous algebras makes the proof more difficult, but also makes the result considerably stronger.

The situation is analogous to the fundamental theorem of algebra, which states that the complex numbers $\mathbb{C}$ are algebraically closed. The

most common proof of this theorem, originally due to Gauss, depends on the analytic structure of $\mathbb{C}$ and uses second-order arguments (see e.g. [117]). However, one can give a first-order, purely algebraic proof of the more general result that if R is any real closed field (such as $\mathbb{R}$ or $\mathbb{A}$, the real algebraic numbers), then $R[i]$ is algebraically closed (see e.g. [112]). Like the fundamental theorem of algebra, our result also deals with solutions of polynomial systems, and our proof replaces arguments referring to the analytic or second-order structure of $\mathsf{Reg}\,\mathbb{N}^k$, embodied in the star-continuity axiom, with first-order equational arguments referring only to the finitary algebraic structure of commutative KA.

Our development involves the definition of differential operators $\frac{\partial}{\partial x}$ on commutative Kleene algebras of polynomials and a version of Taylor's theorem:

$$f(x+d) \quad = \quad f(x) + f'(x+d) \cdot d.$$

Differential operators allow us to define the *Jacobian matrix* of a system of inequalities, which we use to give a closed form solution.

The results of this lecture and the next are from [51].

Commutative Kleene Algebra

Recall that a Kleene algebra is *commutative* if it satisfies the property $xy = yx$ for all x, y. Commutativity assumptions also arise in practice [68].

We have observed previously that the following is a theorem of commutative KA that does not hold in KA in general:

$$(p+q)^* \quad = \quad p^*q^*. \tag{13.2}$$

We have also observed that one can prove a normal form theorem that says that every expression is equivalent to a sum $y_1 + \cdots + y_n$, where each y_i is a product of atomic symbols and expressions of the form $(a_1 \cdots a_k)^*$, where the a_i are atomic symbols. For example,

$$(((ab)^*c)^* + d)^* \quad = \quad d^* + (ab)^*c^*cd^*.$$

This normal form was observed by Pilling [95] in the context of $\mathsf{Reg}\,\mathbb{N}^k$, but using (13.2) it is easily shown to hold in all commutative KAs.

Polynomials

If K is a commutative KA, we denote by $K[\mathbf{x}]$ the commutative KA of polynomials in indeterminates $\mathbf{x}$ over K. These are very much like polynomials over a ring or field. We can think of a polynomial as a regular expression

over K and $\mathbf{x}$ reduced modulo the axioms of commutative KA and the diagram of K (the set of ground identities that hold in K). Typical examples of polynomials are

$$(ax+by)^*$$
$$1+(ax^*b^*)^*+bx+cy$$
$$a+xy(bxy)^*,$$

where x, y are indeterminates and $a, b, c \in K$.

Formally, $K[\mathbf{x}]$ is defined to be the direct sum (coproduct) of K with the free commutative KA on generators $\mathbf{x}$ in the category of commutative KAs. The most significant property of polynomials is that any pair of maps h, h', where $h : K \to L$ is a KA homomorphism and $h' : \mathbf{x} \to L$ is a set function, extend simultaneously and uniquely to a KA homomorphism $\widehat{h} : K[\mathbf{x}] \to L$. When h is the identity on K, the map $\widehat{h}$ is just polynomial evaluation; intuitively, applying $\widehat{h}$ can be regarded as substituting the values $h'(x)$ for the indeterminates $x \in \mathbf{x}$ and then evaluating the resulting expression.

If $\mathbf{x} = x_1, \ldots, x_n$ and $\mathbf{a} = a_1, \ldots, a_n$, we write $f(\mathbf{a})$ or $f(\mathbf{x})\mid_{\mathbf{x}=\mathbf{a}}$ for the value of f evaluated at $x_i \mapsto a_i$, $1 \le i \le n$.

Differential Operators

A map $D : K \to K$ on a commutative KA K is called a *differential operator* if for all $x, y \in K$,

$$\begin{array}{rcl} D(x+y) & = & Dx+Dy \\ D(xy) & = & xDy+yDx \\ D(x^*) & = & x^*Dx \\ D0 & = & D1 \;=\; 0. \end{array} \tag{13.3}$$

For example, in $\mathsf{Reg}\,\mathbb{N}^k$, for every $1 \le i \le k$, the map

$$\begin{array}{rcl} A & \mapsto & \{(a_1, \ldots, a_{i-1}, a_i - 1, a_{i+1}, \ldots, a_k) \mid \\ & & \quad (a_1, \ldots, a_k) \in A,\ a_i > 0\} \end{array}$$

is a differential operator.

Theorem 13.2 *Any differential operator $D : K \to K$ and set function $D : \mathbf{x} \to K$ have a unique joint extension to a differential operator $D : K[\mathbf{x}] \to K[\mathbf{x}]$.*

Proof. The given maps D can be extended by induction to $D : K[\mathbf{x}] \to K[\mathbf{x}]$ using (13.3); but we must take care that the extended D is well-defined on equivalence classes modulo the axioms of commutative KA and the diagram of K. That the extended D respects the diagram of K follows from the fact that the given $D : K \to K$ is a differential operator on

K. To prove that D respects the commutative KA axioms requires a case for each axiom. We argue the cases $a^* = 1 + aa^*$ and $ab \leq b \Rightarrow a^*b \leq b$ explicitly.

For the case $a^* = 1 + aa^*$,

$$\begin{aligned} D(1 + aa^*) &= D1 + D(aa^*) \\ &= 0 + aD(a^*) + a^*Da \\ &= aa^*Da + a^*Da \\ &= a^*Da \\ &= D(a^*). \end{aligned}$$

For the case $ab \leq b \Rightarrow a^*b \leq b$, suppose $ab \leq b$. By the induction hypothesis, $D(ab) \leq Db$, and we wish to show that $D(a^*b) \leq Db$. From $D(ab) \leq Db$ we have that $aDb + bDa \leq Db$, thus by Kleene algebra we have that $a^*Db \leq Db$ and $a^*bDa \leq Db$. Therefore

$$\begin{aligned} D(a^*b) &= a^*Db + bD(a^*) \\ &= a^*Db + ba^*Da \\ &\leq Db. \end{aligned}$$

□

In particular, for $x \in \mathbf{x}$, we define a certain differential operator $\frac{\partial}{\partial x} : K[\mathbf{x}] \to K[\mathbf{x}]$ as follows. The value of $\frac{\partial}{\partial x}$ applied to $f \in K[\mathbf{x}]$ is denoted $\frac{\partial f}{\partial x}$ or $\frac{\partial f}{\partial x}(\mathbf{x})$. We define $\frac{\partial}{\partial x}$ to be the unique differential operator such that $\frac{\partial x}{\partial x} = 1$, $\frac{\partial y}{\partial x} = 0$ for $y \in \mathbf{x} - \{x\}$, and $\frac{\partial a}{\partial x} = 0$ for $a \in K$.

For univariate polynomials $f, e \in K[x]$, we sometimes write f' for $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial x}(e)$ or $f'(e)$ for the result of evaluating the polynomial $\frac{\partial f}{\partial x}$ at $x \mapsto e$.

For example, if $f(x) = axb + x^2 + 1$, then $f'(x) = ab + x$.

Note that $\frac{\partial}{\partial x}(f(e))$ and $\frac{\partial f}{\partial x}(e)$ are different in general. The former refers to the result of evaluating $f(x)$ at $x \mapsto e$ first, then applying the differential operator $\frac{\partial}{\partial x}$ to $f(e)$; whereas the latter refers to the result of applying the differential operator $\frac{\partial}{\partial x}$ to $f(x)$ first, then evaluating the resulting polynomial $\frac{\partial f}{\partial x}(x)$ at $x \mapsto e$. These two expressions are related by the *chain rule*:

Theorem 13.3 (chain rule) *For $f, e \in K[x]$,*

$$\frac{\partial}{\partial x}(f(e)) = \frac{\partial f}{\partial x}(e) \cdot \frac{\partial e}{\partial x},$$

or in more conventional notation,

$$f(e(x))' = f'(e(x)) \cdot e'(x).$$

Proof. This is a straightforward induction on the structure of f. We argue the cases $f = gh$ and $f = g^*$ explicitly.

$$\begin{aligned}(g(e)h(e))' &= g(e)h(e)' + h(e)g(e)' \\ &= g(e)h'(e)e' + h(e)g'(e)e' \\ &= (g(e)h'(e) + h(e)g'(e))e' \\ &= (gh' + hg')(e)e' \\ &= (gh)'(e)e'.\end{aligned}$$

$$\begin{aligned}(g(e)^*)' &= g(e)^*g(e)' \\ &= g(e)^*g'(e)e' \\ &= (g^*g')(e)e' \\ &= (g^*)'(e)e'.\end{aligned}$$

□

For example, if $f(x) = ax + x^* + 1$, then

$$\begin{aligned}f'(x) &= a + x^* \\ f(f(x)) &= a(ax + x^* + 1) + (ax + x^* + 1)^* + 1 \\ &= a^2x + ax^* + (ax)^*x^* + 1 \\ f(f(x))' &= (a^2x + ax^* + (ax)^*x^* + 1)' \\ &= a^2 + (a+1)x^*(ax)^* \\ f'(f(x))f'(x) &= (a + (ax + x^* + 1)^*)(a + x^*) \\ &= a^2 + (a+1)x^*(ax)^*.\end{aligned}$$

We also have the following version of Taylor's theorem in commutative KA.

Theorem 13.4 (Taylor's theorem) *For $f, d \in K[x]$,*

$$f(x + d) = f(x) + f'(x + d) \cdot d.$$

In particular, evaluating at $x \mapsto 0$,

$$f(d) = f(0) + f'(d) \cdot d.$$

Proof. This is again a straightforward induction on the structure of f. As before, we argue the cases $f = gh$ and $f = g^*$ explicitly. For the case

$f = gh$,

$$\begin{aligned} & gh(x+d) \\ &= g(x+d)h(x+d) \\ &= g(x+d)h(x) + g(x+d)h'(x+d)d \\ &= g(x)h(x) + g'(x+d)h(x)d \\ &\quad + g(x+d)h'(x+d)d, \end{aligned}$$

and by symmetry,

$$\begin{aligned} & gh(x+d) \\ &= g(x)h(x) + g'(x+d)h(x+d)d \\ &\quad + g(x)h'(x+d)d, \end{aligned}$$

therefore by monotonicity,

$$\begin{aligned} & gh(x+d) \\ &= g(x)h(x) + g'(x+d)h(x)d + g(x+d)h'(x+d)d \\ &\quad + g'(x+d)h(x+d)d + g(x)h'(x+d)d \\ &= g(x)h(x) + g(x+d)h'(x+d)d \\ &\quad + g'(x+d)h(x+d)d \\ &= gh(x) + (gh)'(x+d)d. \end{aligned}$$

For the case $f = g^*$,

$$\begin{aligned} & g(x+d)^* \\ &= (g(x) + g'(x+d)d)^* \\ &= g(x)^*(g'(x+d)d)^* \quad \text{by (13.2)} \\ &= g(x)^* + g(x)^* g'(x+d)d(g'(x+d)d)^* \\ &= g(x)^* + g'(x+d)dg(x)^*(g'(x+d)d)^* \\ &= g(x)^* + g'(x+d)dg(x+d)^* \\ &= g(x)^* + g(x+d)^* g'(x+d)d \\ &= g(x)^* + (g^*)'(x+d)d. \end{aligned}$$

□

Continuing with the example $f(x) = ax + x^* + 1$ above,

$$\begin{aligned} f(x+d) &= a(x+d) + (x+d)^* + 1 \\ &= ax + ad + x^*d^* + 1 \\ f'(x+d) &= a + (x+d)^* \\ &= a + x^*d^* \\ f(x) + f'(x+d)d &= ax + x^* + 1 + ad + x^*d^*d \\ &= ax + ad + x^*d^* + 1. \end{aligned}$$

We often wish to differentiate simultaneously with respect to a sequence of indeterminates $\mathbf{y} = y_1, \ldots, y_k$. We define an operator $\frac{\partial}{\partial \mathbf{y}}$ that when applied to an element $f \in K[\mathbf{x}]$ produces a row vector of length k whose i^{th} component is $\frac{\partial f}{\partial y_i}$. More generally, $\frac{\partial}{\partial \mathbf{y}}$ applied to a column vector consisting of m elements $f_1, \ldots, f_m \in K[\mathbf{x}]$ produces an $m \times k$ matrix whose i, j^{th} element is $\frac{\partial f_i}{\partial y_j}$.

By iterating Theorem 13.4, one can show that for $f \in K[\mathbf{x}]$ and $\mathbf{e} = e_1, \ldots, e_n$,

$$\begin{aligned} f(\mathbf{e}) &= f(0, \ldots, 0) + \tfrac{\partial f}{\partial x_1}(\mathbf{e})e_1 + \cdots + \tfrac{\partial f}{\partial x_n}(\mathbf{e})e_n \\ &= f(\mathbf{0}) + \tfrac{\partial f}{\partial \mathbf{x}}(\mathbf{e}) \bullet \mathbf{e}, \end{aligned}$$

where $\bullet$ denotes dot product of vectors. The same holds for a column vector $\mathbf{f} = f_1, \ldots, f_m$ of elements of $K[\mathbf{x}]$; here $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})$ is an $m \times n$ matrix whose i, j^{th} element is $\frac{\partial f_i}{\partial x_j}$. This matrix is called the *Jacobian* of $\mathbf{f}$. We thus have

Theorem 13.5

$$\mathbf{f}(\mathbf{e}) = \mathbf{f}(\mathbf{0}) + \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{e}) \cdot \mathbf{e},$$

where $\cdot$ denotes matrix-vector multiplication.

We also have the following vector-vector and matrix-vector versions of the chain rule, Theorem 13.3:

Theorem 13.6

$$\begin{aligned} \tfrac{\partial}{\partial z}(f(\mathbf{e})) &= \tfrac{\partial f}{\partial x_1}(\mathbf{e})\tfrac{\partial e_1}{\partial z} + \cdots + \tfrac{\partial f}{\partial x_n}(\mathbf{e})\tfrac{\partial e_n}{\partial z} \\ &= \tfrac{\partial f}{\partial \mathbf{x}}(\mathbf{e}) \cdot \tfrac{\partial \mathbf{e}}{\partial z} \end{aligned}$$

$$\tfrac{\partial}{\partial z}(\mathbf{f}(\mathbf{e})) = \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{e}) \cdot \tfrac{\partial \mathbf{e}}{\partial z}.$$

The proof of these theorems are straightforward generalizations of their one-dimensional versions.

In the next lecture we will apply these general results to obtain an algebraic version of Parikh's theorem.

Lecture 14

Parikh's Theorem in Commutative KA, continued

In this lecture we use the infrastructure developed in the previous lecture to prove

Theorem 14.1 *Every system of inequalities*

$$f_i(x_1, \ldots, x_n) \quad \leq \quad x_i, \quad 1 \leq i \leq n, \tag{14.1}$$

where the f_i are polynomials in $K[x_1, \ldots, x_n]$ over a commutative Kleene algebra K, has a unique least solution in K^n; moreover, the components of the solution are given by polynomials in the coefficients of the f_i.

We first prove the result for $n = 1$, then extend it to arbitrary n. Many arguments in this section are inspired by those of Pilling [95] (see also [29]) but generalized to apply to arbitrary commutative KAs.

Theorem 14.2 *Let K be a commutative* KA *and let $f(x) \in K[x]$. The unique least solution of the inequality $f(x) \leq x$ is*

$$f'(f(0))^* \cdot f(0). \tag{14.2}$$

Moreover, this holds uniformly over all homomorphic images of K.

For example, the context-free language $A = \{a^n b^n \mid n \geq 0\}$ is generated by the grammar $S \to aSb \mid \varepsilon$, which translates to the one-dimensional system $axb + 1 \leq x$. Letting $f(x) = axb + 1$, we get $f'(x) = ab$ and $f(0) = 1$, thus (14.2) gives $(ab)^*$. This is a regular expression describing a regular set letter-equivalent to A.

Proof. First we argue that (14.2) is a solution to $f(x) \leq x$. It follows by a straightforward inductive argument that for any polynomial $h(x)$,

$$ac \leq bc \quad \Rightarrow \quad h(a)c \leq h(b)c. \tag{14.3}$$

Applying this with $b = f(0)$, $c = f'(b)^*$, and $a = bc$,

$$\begin{aligned}
f(a) &= f(bc) \\
&= f(0) + f'(bc)bc && \text{by Taylor's theorem (Theorem 13.4 of Lecture ??)} \\
&= b + f'(bc)bc \\
&\leq b + f'(b)bc && \text{by (14.3)} \\
&= b + f'(b)f'(b)^*b \\
&= f'(b)^*b && \text{by Kleene algebra} \\
&= a.
\end{aligned}$$

Now we show that (14.2) is the least solution. Suppose y is any solution; thus $f(y) \leq y$. We wish to show that

$$f'(f(0))^* f(0) \quad \leq \quad y.$$

By Kleene algebra, it suffices to show

$$f(0) + f'(f(0)) \cdot y \quad \leq \quad y.$$

But by monotonicity, $f(0) \leq f(y) \leq y$, and

$$\begin{aligned}
&f(0) + f'(f(0)) \cdot y \\
&\quad \leq f(0) + f'(y) \cdot y && \text{by monotonicity} \\
&\quad = f(y) && \text{by Taylor's theorem (Theorem 13.4 of Lecture ??)} \\
&\quad \leq y.
\end{aligned}$$

The expression (14.2) gives the least solution of $f(x) \leq x$ uniformly over all homomorphic images of K because the axioms of KA used in the proof hold universally under any interpretation. □

The uniformity condition of Theorem 14.2 may seem obvious, but it is actually a rather subtle point. The issue is that equations are preserved under homomorphisms, but in general Horn formulas (equational implications) are not. The homomorphic image $h(e)$ of a solution e of an inequality $f(x) \leq x$ is a solution of the homomorphic image of the inequality,

because the inequality is equivalent to an equation $f(x) + x = x$; but that $h(e)$ is the *least* solution does not follow from the fact that e is least, since this property requires a Horn formula.

Proof of Theorem 14.1. We iterate the one-dimensional solution as follows. Consider the two-dimensional system

$$\begin{array}{rcl} f(x,y) & \leq & x \\ g(x,y) & \leq & y. \end{array} \tag{14.4}$$

Viewing $K[x,y]$ as $K[x][y]$, first compute the least solution to the one-dimensional system $g(x,y) \leq y$ in $K[x]$; call it $h(x)$. Then compute the least solution a of $f(x,h(x)) \leq x$ in K.

We claim that $(a,h(a))$ is the desired least solution to (14.4) in K^2. Surely $f(a,h(a)) \leq a$ by the one-dimensional argument. Moreover, by the uniformity observation, we also have $g(a,h(a)) \leq h(a)$, since it is the image of $g(x,h(x)) \leq h(x)$ under the evaluation homomorphism $x \mapsto a$.

To show $(a,h(a))$ is the least solution, suppose (b,c) is any other solution. Then $f(b,c) \leq b$ and $g(b,c) \leq c$. Using the uniformity observation with the evaluation morphism $x \mapsto b$, we have that $h(b)$ is the least solution of $g(b,y) \leq y$. Then $h(b) \leq c$. But by monotonicity, $f(b,h(b)) \leq f(b,c) \leq b$. Since a is the least solution to $f(x,h(x)) \leq x$, we have that $a \leq b$. Again by monotonicity, $h(a) \leq h(b) \leq c$. Thus $(a,h(a)) \leq (b,c)$.

By iterating this process inductively, we can obtain the existence of a solution to any $n \times n$ system. □

A Closed Form Solution

The iterated construction of the previous section does not give a symmetric closed-form expression for any dimension greater than one. In this section we provide a symmetric closed-form solution.

Let K be a commutative KA and consider an $n \times n$ system

$$\mathbf{f}(\mathbf{x}) \quad \leq \quad \mathbf{x} \tag{14.5}$$

where $\mathbf{x} = x_1, \ldots, x_n$ and $\mathbf{f} = \mathbf{f}(\mathbf{x}) = f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}) \in K[\mathbf{x}]$. Let $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ be the Jacobian of the system (14.5) as defined in Lecture **??**. Define

$$\begin{array}{rcl} \mathbf{a}_0 & \stackrel{\text{def}}{=} & \mathbf{f}(\mathbf{0}) \\ \mathbf{a}_{k+1} & \stackrel{\text{def}}{=} & \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_k)^* \mathbf{a}_k. \end{array}$$

Theorem 14.3 *For sufficiently large finite N, the n-vector $\mathbf{a}_N$ is the least solution to* (14.5)*. Moreover, this solution is uniform over all homomorphic images of K.*

Below we will derive an explicit single-exponential bound on N as a function of n.

Proof. We will prove the first statement of the theorem; the uniformity property will follow by the same considerations as in the proof of Theorem 14.2. The proof proceeds by induction on n. The basis $n = 1$ was given in Theorem 14.2.

Now suppose $n \geq 2$. Partition n as $m + (n - m)$ where $1 \leq m < n$. For an n-vector $\mathbf{b} = b_1, \ldots, b_n$, write

$$\begin{aligned} \mathbf{b}^{\blacksquare\square} &\stackrel{\text{def}}{=} b_1, \ldots, b_m \\ \mathbf{b}^{\square\blacksquare} &\stackrel{\text{def}}{=} b_{m+1}, \ldots, b_n, \end{aligned}$$

and for an $n \times n$ matrix M, write

$$M^{\blacksquare\blacksquare}, \quad M^{\square\square}, \quad M^{\blacksquare\square}, \quad M^{\square\blacksquare}$$

for the upper left $m \times m$, upper right $m \times (n - m)$, lower left $(n - m) \times m$, and lower right $(n - m) \times (n - m)$ submatrices of M, respectively. To simplify notation, define

$$\begin{aligned} \mathbf{y} &\stackrel{\text{def}}{=} \mathbf{x}^{\blacksquare}, & \gg &\stackrel{\text{def}}{=} \mathbf{f}^{\blacksquare}, \\ \mathbf{z} &\stackrel{\text{def}}{=} \mathbf{x}^{\square}, & \mathbf{h} &\stackrel{\text{def}}{=} \mathbf{f}^{\square}. \end{aligned}$$

In this notation, we can rewrite (14.5) as

$$\begin{aligned} \gg(\mathbf{y}, \mathbf{z}) &\leq \mathbf{y} \\ \mathbf{h}(\mathbf{y}, \mathbf{z}) &\leq \mathbf{z}. \end{aligned} \tag{14.6}$$

Also,

$$\begin{aligned} \frac{\partial \gg}{\partial \mathbf{y}} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\blacksquare\blacksquare}, & \frac{\partial \gg}{\partial \mathbf{z}} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\square\square}, \\ \frac{\partial \mathbf{h}}{\partial \mathbf{y}} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\blacksquare\square}, & \frac{\partial \mathbf{h}}{\partial \mathbf{z}} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\square\blacksquare}. \end{aligned}$$

Now define

$$\begin{aligned} \mathbf{c}_0(\mathbf{y}) &\stackrel{\text{def}}{=} \mathbf{h}(\mathbf{y}, \mathbf{0}) \\ \mathbf{c}_{k+1}(\mathbf{y}) &\stackrel{\text{def}}{=} \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y}, \mathbf{c}_k(\mathbf{y}))^* \mathbf{c}_k(\mathbf{y}) \end{aligned}$$

By the induction hypthesis, there exists a P such that $\mathbf{c}_P(\mathbf{y})$ is the least solution to the system

$$\mathbf{h}(\mathbf{y}, \mathbf{z}) \leq \mathbf{z}$$

uniformly in $\mathbf{y}$. Define

$$\begin{aligned}
\widehat{\mathbf{g}}(\mathbf{y}) &\stackrel{\text{def}}{=} \gg(\mathbf{y}, \mathbf{c}_P(\mathbf{y})) \\
\mathbf{b}_0 &\stackrel{\text{def}}{=} \widehat{\mathbf{g}}(\mathbf{0}) \\
\mathbf{b}_{k+1} &\stackrel{\text{def}}{=} \tfrac{\partial \widehat{\mathbf{g}}}{\partial \mathbf{y}}(\mathbf{b}_k)^* \mathbf{b}_k.
\end{aligned}$$

Again by the induction hypthesis, there exists an M such that $\mathbf{b}_M$ is the least solution to the system

$$\widehat{\mathbf{g}}(\mathbf{y}) \leq \mathbf{y}.$$

By the uniformity of the solutions, we have that

$$\begin{aligned}
\gg(\mathbf{b}_M, \mathbf{c}_P(\mathbf{b}_M)) &\leq \mathbf{b}_M \\
\mathbf{h}(\mathbf{b}_M, \mathbf{c}_P(\mathbf{b}_M)) &\leq \mathbf{c}_P(\mathbf{b}_M),
\end{aligned}$$

and $\mathbf{b}_M, \mathbf{c}_P(\mathbf{b}_M)$ is the least solution to (14.5). Moreover, this is the least solution uniformly over all homomorphic images.

Our task now is to show that for sufficiently large N,

$$\mathbf{b}_M = \mathbf{a}_N^{\blacksquare\square}, \qquad \mathbf{c}_P(\mathbf{b}_M) = \mathbf{a}_N^{\square\blacksquare}. \tag{14.7}$$

The inequalities $\geq$ follow from the fact that if $\mathbf{u}$ is any solution to (14.5), then $\mathbf{a}_k \leq \mathbf{u}$ for all k. This can be shown by induction on k. Certainly

$$\mathbf{a}_0 = \mathbf{f}(\mathbf{0}) \leq \mathbf{f}(\mathbf{u}) \leq \mathbf{u},$$

and by Theorem 13.5 of Lecture **??**,

$$\tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{u})\mathbf{u} \leq \mathbf{u},$$

from which it follows that

$$\begin{aligned}
\mathbf{a}_{k+1} &= \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_k)^* \mathbf{a}_k \\
&\leq \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{u})^* \mathbf{u} \\
&\leq \mathbf{u}.
\end{aligned}$$

Now we establish a series of inequalities from which the forward inequalities $\leq$ of (14.7) will follow. First,

$$\mathbf{a}_k \leq \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_k)^* \mathbf{a}_k = \mathbf{a}_{k+1},$$

thus $\mathbf{a}_i \leq \mathbf{a}_j$, $i \leq j$, and similarly for $\mathbf{b}_i$ and $\mathbf{c}_i$.

Now we show that

$$\gg(\mathbf{a}_k) \leq \mathbf{a}_{k+1}^{\blacksquare\square} \qquad \mathbf{h}(\mathbf{a}_k) \leq \mathbf{a}_{k+1}^{\square\blacksquare}. \tag{14.8}$$

By Theorem refthm-Taylor1 of Lecture **??**,

$$\begin{aligned}
\gg(\mathbf{a}_k) &= \gg(\mathbf{0}) + \tfrac{\partial \gg}{\partial \mathbf{x}}(\mathbf{a}_k)\mathbf{a}_k \\
&= \mathbf{f}(\mathbf{0})^{\blacksquare} + \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_k)^{\blacksquare\blacksquare}\mathbf{a}_k \\
&= \mathbf{a}_0^{\blacksquare} + (\tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_k)\mathbf{a}_k)^{\blacksquare} \\
&\leq (\mathbf{a}_k + \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_k)^*\mathbf{a}_k)^{\blacksquare} \\
&= \mathbf{a}_{k+1}^{\blacksquare}.
\end{aligned}$$

The second inequality of (14.8) follows from a similar argument.

Now we show by induction on j that

$$\mathbf{c}_j(\mathbf{a}_k^{\blacksquare}) \leq \mathbf{a}_{j+k+1}^{\square}. \tag{14.9}$$

For the basis,

$$\begin{aligned}
\mathbf{c}_0(\mathbf{a}_k^{\blacksquare}) &= \mathbf{h}(\mathbf{a}_k^{\blacksquare}, \mathbf{0}) \\
&\leq \mathbf{h}(\mathbf{a}_k) \\
&\leq \mathbf{a}_{k+1}^{\blacksquare}.
\end{aligned}$$

For the induction step,

$$\begin{aligned}
\mathbf{c}_{j+1}(\mathbf{a}_k^{\blacksquare}) &= \tfrac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{a}_k^{\blacksquare}, \mathbf{c}_j(\mathbf{a}_k^{\blacksquare}))^*\mathbf{c}_j(\mathbf{a}_k^{\blacksquare}) \\
&\leq \tfrac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{a}_k^{\blacksquare}, \mathbf{a}_{j+k+1}^{\blacksquare})^*\mathbf{a}_{j+k+1}^{\square} \\
&\leq \tfrac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{a}_{j+k+1})^*\mathbf{a}_{j+k+1}^{\blacksquare} \\
&= (\tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_{j+k+1})^{\square\blacksquare})^*\mathbf{a}_{j+k+1}^{\blacksquare} \\
&\leq (\tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_{j+k+1})^*)^{\square\blacksquare}\mathbf{a}_{j+k+1}^{\blacksquare} \\
&\leq (\tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_{j+k+1})^*\mathbf{a}_{j+k+1})^{\blacksquare} \\
&= \mathbf{a}_{j+k+2}^{\blacksquare}.
\end{aligned}$$

Now we show by induction on k that when $m = n - 1$, that is, for $|z| = 1$,

$$\tfrac{\partial \mathbf{c}_{k+1}}{\partial \mathbf{y}}(\mathbf{y}) \leq \tfrac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y}, \mathbf{c}_k(\mathbf{y}))^*\tfrac{\partial \mathbf{h}}{\partial \mathbf{y}}(\mathbf{y}, \mathbf{c}_k(\mathbf{y})). \tag{14.10}$$

The corresponding result for $m < n-1$ would require some specialized notation even to state. The proof for $m = n-1$ is considerably simpler, so we henceforth restrict ourselves to that case.

First we note that

$$\frac{\partial \mathbf{c}_0}{\partial \mathbf{y}}(\mathbf{y}) \;=\; \frac{\partial}{\partial \mathbf{y}}(\mathbf{h}(\mathbf{y},\mathbf{0})) \;=\; \frac{\partial \mathbf{h}}{\partial \mathbf{y}}(\mathbf{y},\mathbf{0}). \tag{14.11}$$

Then

$$\begin{aligned}
&\frac{\partial \mathbf{c}_{k+1}}{\partial \mathbf{y}}(\mathbf{y}) \\
&\quad= \frac{\partial}{\partial \mathbf{y}}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\mathbf{c}_k(\mathbf{y})\right) \\
&\quad= \mathbf{c}_k(\mathbf{y})\frac{\partial}{\partial \mathbf{y}}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\right) \\
&\qquad\quad + \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y}) \\
&\quad= \mathbf{c}_k(\mathbf{y})\frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial}{\partial \mathbf{y}}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))\right) \\
&\qquad\quad + \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y}) \\
&\quad= \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\mathbf{c}_k(\mathbf{y})\frac{\partial}{\partial \mathbf{y}}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))\right) \\
&\qquad\quad + \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y}) \\
&\quad\le \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial}{\partial \mathbf{y}}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))\mathbf{c}_k(\mathbf{y})\right) \\
&\qquad\quad + \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y}) \\
&\quad\le \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial}{\partial \mathbf{y}}(\mathbf{h}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))) \\
&\qquad\quad + \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y}) \\
&\quad= \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\left(\frac{\partial \mathbf{h}}{\partial \mathbf{y}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))\right. \\
&\qquad\quad + \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))\frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y})\Big) \\
&\qquad\quad + \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y}) \\
&\quad= \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\left(\frac{\partial \mathbf{h}}{\partial \mathbf{y}}(\mathbf{y},\mathbf{c}_k(\mathbf{y})) + \frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y})\right),
\end{aligned} \tag{14.12}$$

so it suffices to show

$$\frac{\partial \mathbf{c}_k}{\partial \mathbf{y}}(\mathbf{y}) \;\le\; \frac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{y},\mathbf{c}_k(\mathbf{y}))^*\frac{\partial \mathbf{h}}{\partial \mathbf{y}}(\mathbf{y},\mathbf{c}_k(\mathbf{y})).$$

For $k = 0$, this is immediate from (14.11). For $k > 0$, this follows from (14.12) and the induction hypothesis.

Now we show by induction on k that

$$\mathbf{b}_k \;\le\; \mathbf{a}^{\blacksquare}_{(k+1)(P+2)}. \tag{14.13}$$

For the basis,

$$
\begin{aligned}
\mathbf{b}_0 &= \widehat{\mathbf{g}}(\mathbf{0}) \\
&= \gg(\mathbf{0}, \mathbf{c}_P(\mathbf{0})) \\
&\leq \gg(\mathbf{0}, \mathbf{c}_P(\mathbf{a}_0^{\blacksquare})) \\
&\leq \gg(\mathbf{0}, \mathbf{a}_{P+1}^{\blacksquare}) \\
&\leq \gg(\mathbf{a}_{P+1}) \\
&\leq \mathbf{a}_{P+2}^{\blacksquare}.
\end{aligned}
$$

For the induction step,

$$
\begin{aligned}
&\mathbf{b}_{k+1}\\
&\quad = \quad \tfrac{\partial \widehat{\mathbf{g}}}{\partial \mathbf{y}}(\mathbf{b}_k)^* \mathbf{b}_k\\
&\quad = \quad \tfrac{\partial}{\partial \mathbf{y}}(\gg(\mathbf{y}, \mathbf{c}_P(\mathbf{y})))^* \mathbf{y} \mid_{\mathbf{y}=\mathbf{b}_k}\\
&\quad = \quad (\tfrac{\partial \gg}{\partial \mathbf{y}}(\mathbf{y}, \mathbf{c}_P(\mathbf{y}))\\
&\qquad\quad + \tfrac{\partial \gg}{\partial \mathbf{z}}(\mathbf{y}, \mathbf{c}_P(\mathbf{y}))\tfrac{\partial \mathbf{c}_P}{\partial \mathbf{y}}(\mathbf{y}))^* \mathbf{y} \mid_{\mathbf{y}=\mathbf{b}_k}\\
&\quad = \quad (\tfrac{\partial \gg}{\partial \mathbf{y}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))\\
&\qquad\quad + \tfrac{\partial \gg}{\partial \mathbf{z}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))\tfrac{\partial \mathbf{c}_P}{\partial \mathbf{y}}(\mathbf{b}_k))^* \mathbf{b}_k\\
&\quad \leq \quad (\tfrac{\partial \gg}{\partial \mathbf{y}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))\\
&\qquad\quad + \tfrac{\partial \gg}{\partial \mathbf{z}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))\tfrac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{b}_k, \mathbf{c}_{P-1}(\mathbf{b}_k))^*\\
&\qquad\qquad \cdot \tfrac{\partial \mathbf{h}}{\partial \mathbf{y}}(\mathbf{b}_k, \mathbf{c}_{P-1}(\mathbf{b}_k)))^* \mathbf{b}_k\\
&\quad \leq \quad (\tfrac{\partial \gg}{\partial \mathbf{y}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))\\
&\qquad\quad + \tfrac{\partial \gg}{\partial \mathbf{z}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))\tfrac{\partial \mathbf{h}}{\partial \mathbf{z}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))^*\\
&\qquad\qquad \cdot \tfrac{\partial \mathbf{h}}{\partial \mathbf{y}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k)))^* \mathbf{b}_k\\
&\quad = \quad (\tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))^{\blacksquare\square}\\
&\qquad\quad + \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))^{\square\blacksquare}\tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))^{\square\blacksquare *}\\
&\qquad\qquad \cdot \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))^{\blacksquare\blacksquare})^* \mathbf{b}_k\\
&\quad \leq \quad \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{b}_k, \mathbf{c}_P(\mathbf{b}_k))^{*\blacksquare\square} \mathbf{b}_k\\
&\quad \leq \quad \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}^{\blacksquare}_{(k+1)(P+2)}, \mathbf{c}_P(\mathbf{a}^{\blacksquare}_{(k+1)(P+2)}))^{*\blacksquare\square} \mathbf{a}^{\blacksquare}_{(k+1)(P+2)}\\
&\quad \leq \quad \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}^{\blacksquare}_{(k+1)(P+2)}, \mathbf{a}^{\square}_{(k+2)(P+2)-1})^{*\blacksquare\square} \mathbf{a}^{\blacksquare}_{(k+1)(P+2)}\\
&\quad \leq \quad \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_{(k+2)(P+2)-1})^{*\blacksquare\square} \mathbf{a}^{\square}_{(k+2)(P+2)-1}\\
&\quad \leq \quad (\tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_{(k+2)(P+2)-1})^* \mathbf{a}_{(k+2)(P+2)-1})^{\square}\\
&\quad = \quad \mathbf{a}^{\blacksquare}_{(k+2)(P+2)}.
\end{aligned}
$$

It follows from (14.9) and (14.13) that

$$
\begin{aligned}
\mathbf{b}_M &\leq \mathbf{a}^{\blacksquare}_{(M+1)(P+2)},\\
\mathbf{c}_P(\mathbf{b}_M) &\leq \mathbf{a}^{\square}_{(M+2)(P+2)-1}.
\end{aligned}
$$

Taking $m = n - 1$, Theorem 14.2 says that $P = 1$ suffices. Thus the N in the statement of the theorem is bounded by $(M+2)(P+2) - 1 = 3M + 5$. This gives the following recurrence for N as a function of n:

$$\begin{aligned} N(1) &= 1 \\ N(n+1) &= 3N(n) + 5 \end{aligned}$$

with solution $N(n) = (7 \cdot 3^n - 5)/2$. □

Lecture 15

Kleene Algebra with Tests

In semantics and logics of programs, Kleene algebra forms an essential component of Propositional Dynamic Logic (PDL) [36], in which it is mixed with Boolean algebra and modal logic to give a simple yet powerful system for modeling and reasoning about computation at the propositional level.

Syntactically, PDL is a two-sorted logic consisting of *programs* and *propositions* defined by mutual induction. A test $\varphi?$ can be formed from any proposition φ; intuitively, $\varphi?$ acts as a guard that succeeds with no side effects in states satisfying φ and fails or aborts in states not satisfying φ. Semantically, programs are modeled as binary relations on a set of states, and $\varphi?$ is interpreted as the subset of the identity relation consisting of all pairs (s, s) such that φ is true in state s.

From a practical point of view, many simple program manipulations, such as loop unwinding and basic safety analysis, do not require the full power of PDL, but can be carried out in a purely equational subsystem using the axioms of Kleene algebra. However, tests are an essential ingredient, since they are needed to model conventional programming constructs such as conditionals and `while` loops. We define here a variant of Kleene algebra, called *Kleene algebra with tests* (KAT), for reasoning equationally with these constructs. KAT was introduced in [66, 68].

KAT has been applied successfully in a number of low-level verification tasks involving communication protocols, basic safety analysis, con-

currency control, and local compiler optimizations [4, 25, 26, 74]. A useful feature of KAT in this regard is its ability to accommodate certain basic equational assumptions regarding the interaction of atomic instructions and tests. This feature makes KAT ideal for reasoning about the correctness of low-level code transformations.

In this lecture we introduce the system and illustrate its use by giving a complete equational proof of a classical folk theorem [45, 86] which states that every `while` program can be simulated by another `while` program with at most one `while` loop. The approach we take is that of [86], who gives a set of local transformations that allow every `while` program to be transformed systematically to one with at most one `while` loop. For each such transformation, we can give a purely equational proof of correctness.

Definition of Kleene Algebra with Tests

A *Kleene algebra with tests* is a two-sorted algebra

$$(K, B, +, \cdot, ^*, 0, 1, ^-)$$

where $^-$ is a unary operator defined only on B, such that

- $(K, +, \cdot, ^*, 0, 1)$ is a Kleene algebra,
- $(B, +, \cdot, ^-, 0, 1)$ is a Boolean algebra, and
- $(B, +, \cdot, 0, 1)$ is a sub-semiring of $(K, +, \cdot, 0, 1)$.

The elements of B are called *tests*. We reserve the letters $p, q, r, \ldots$ for arbitrary elements of K and $a, b, c, \ldots$ for tests. In PDL, a test would be written $b?$, but since we are using different symbols for tests we can omit the ?.

The axioms for Kleene algebra were given in an earlier lecture. The axioms for Boolean algebra are just the laws of propositional logic. Expressed in our language with $+$ for join (disjunction, "or"), $\cdot$ for meet (conjunction, "and"), and $^-$ for negation (complement, "not"), they are

$$\begin{array}{llll}
x + (y + z) = (x + y) + z & x(yz) = (xy)z & x + y = y + x & xy = yx \\
x(y + z) = xy + xz & x + yz = (x + y)(x + z) & x + 0 = x & x1 = x \\
x0 = 0 & x + 1 = 1 & x + x = x & xx = x \\
x\overline{x} = 0 & x + \overline{x} = 1 & \overline{x}\,\overline{y} = \overline{x + y} & \overline{x} + \overline{y} = \overline{xy} \\
\overline{\overline{x}} = x. & & &
\end{array}$$

The definition of a KAT is deceptively simple, but actually carries a lot of information in a concise package. Note the overloading of the operators $+, \cdot, 0, 1$, each of which plays two roles: applied to arbitrary elements of K,

they refer to nondeterministic choice, composition, fail, and skip, respectively; and applied to tests, they take on the additional meaning of Boolean disjunction, conjunction, falsity, and truth, respectively. These two usages do not conflict—for example, sequential testing of b and c is the same as testing their conjunction—and their coexistence admits considerable economy of expression.

It follows immediately from the definition that $b \leq 1$ for all $b \in B$ (this is an axiom of Boolean algebra). It is tempting to define tests in an arbitrary Kleene algebra to be the set $\{p \in K \mid p \leq 1\}$, and this is the approach taken by [23]. This approach makes some sense in algebras of binary relations [91, **?**], but it does not work in all Kleene algebras. For example, it rules out the tropical Kleene algebra described in Lecture **??**. In this algebra, $p \leq 1$ for all p, but the idempotence law $pp = p$ fails, so the set $\{p \in K \mid p \leq 1\}$ does not form a Boolean algebra. In our approach, every Kleene algebra extends trivially to a Kleene algebra with tests by taking the two-element Boolean algebra $\{0, 1\}$. Of course, there are more interesting models as well.

However, there is a more serious issue. Even in algebras of binary relations, this approach forces us to consider all elements $p \leq 1$ as tests, including conditions that in practice would not normally be considered testable. For example, there may be programs p whose input/output relations have no side effects—that is, $p \leq 1$—but the associated test succeeds iff p halts, which in general is undecidable. We intend tests to be viewed as simple predicates that are easily recognizable as such and that are immediately decidable in a given state (and whose complements are therefore also immediately decidable). Having an explicit Boolean subalgebra allows this.

While Programs

We will work with a Pascal-like programming language with sequential composition p `;` q, a conditional test `if` b `then` p `else` q, and a looping construct `while` b `do` p. Programs built inductively from atomic programs and tests using these constructs are called *while programs*.

We occasionally omit the `else` clause of a conditional test. This can be considered an abbreviation for a conditional test with the dummy `else` clause 1 (true).

These constructs are modeled in Kleene algebra with tests as follows:

$$\begin{aligned} p\,;q &= pq \\ \texttt{if } b \texttt{ then } p \texttt{ else } q &= bp + \overline{b}q \\ \texttt{if } b \texttt{ then } p &= bp + \overline{b} \\ \texttt{while } b \texttt{ do } p &= (bp)^*\overline{b}. \end{aligned}$$

Conditions

We will also be reasoning in the presence of extra assumptions. Recall that Kleene algebra and KAT are Horn theories, and these extra assumptions take the form of equational premises to the left of the implication symbol in a Horn formala.

There are several common forms of premises that arise in applications. The most common form is a commutativity condition $pq = qp$ or $pb = bp$. A commutativity condition between two atomic programs p and q expresses the fact that the two operations represented by p and q do not interfere with each other and may be done in either order. For example, if p and q are the assignments $x := 3$ and $y := 4$ respectively, then they do not affect each other. A commutativity condition $pb = bp$, where p is a program and b is a test, indicates that the execution of p does not affect the truth value of b. For example, if p is the assignment $x := 3$ and b is the test $y = 4$, then execution of p has no effect on the truth value of b.

Another common assumption comes in the form of a pair of equations $p = pb$ and $bp = b$. The first equation indicates that the execution of p causes b to become true, thus testing b after executing p is always redundant. The second equation indicates that if the sole purpose of p is to make b true, then there is no need to execute p if b is already true. For example, if p is the assignment $x := 3$ and b is the test $x = 3$, then these conditions hold.

This is useful for example when we want to eliminate a redundant assignment. We can use the first equation $p = pb$ to generate the condition b in a KAT expression, then use commutativity conditions to move b around until it comes up against another p, then use the second equation $bp = b$ to eliminate the second occurrence of p, then move the b back again and eliminate it by applying $p = pb$ in the opposite direction.

It stands to reason that if p does not affect b, then neither should it affect $\overline{b}$. This is indeed the case. Thus $pb = bp$ should be equivalent to $p\overline{b} = \overline{b}p$. More generally,

Lemma 15.1 *In any Kleene algebra with tests, the following are equivalent:*

(i) $bp = pc$

(ii) $\overline{b}p = p\overline{c}$

(iii) $bp\overline{c} + \overline{b}pc = 0$.

Proof. By symmetry, it suffices to show the equivalence of (i) and (iii). Assuming (i), we have

$$bp\overline{c} + \overline{b}pc = pc\overline{c} + \overline{b}bp = p0 + 0p = 0.$$

Conversely, assuming (iii), we have $bp\overline{c} = \overline{b}pc = 0$. Then

$$bp = bp(c + \overline{c}) = bpc + bp\overline{c} = bpc + 0 = bpc + \overline{b}pc = (b + \overline{b})pc = pc.$$

□

Of course, any pair of tests commute; that is, $bc = cb$. This is an axiom of Boolean algebra.

A Folk Theorem

One can give a purely equational proof, using KAT and commutativity conditions, of a classical result: Every `while` program can be simulated by a `while` program with at most one `while` loop. This theorem is the subject of a treatise on folk theorems by Harel [45], who notes that it is commonly but erroneously attributed to Böhm and Jacopini [16] and who argues with some justification that it was known to Kleene. The version as stated here is originally due to Mirkowska [86], who gives a set of local transformations that allow every `while` program to be transformed systematically to one with at most one `while` loop. We consider a similar set of local transformations and give a purely equational proof of correctness for each. This result illustrates the use of Kleene algebra with tests and commutativity conditions in program equivalence proofs.

It is a commonly held belief that this result has no purely schematic (that is, propositional, uninterpreted) proof [45]. The proofs of [48, 86], as reported in [45], use extra variables to remember certain values at certain points in the program, either program counter values or the values of tests. Since having to remember these values seems unavoidable, one might infer that the only recourse is to introduce extra variables, along with an explicit assignment mechanism for assigning values to them. Thus, as the argument goes, proofs of this theorem cannot be purely propositional.

In a sense, this is true. However, in KAT, if we need to preserve the value of a test b across an action p with which b is not guaranteed to commute, we can introduce a new test c and commutativity condition $cp = pc$, which intuitively says that the computation of p does not affect the value

of c. Then we insert in an appropriate place the program $s\ ;\ bc + \overline{b}\overline{c}$, where s is a new atomic program. Intuitively, we regard s as assigning the value of b to some new Boolean variable that is tested in c, although there is no explicit mechanism for doing this; s is just an uninterpreted atomic program symbol. However, the guard $bc + \overline{b}\overline{c}$ (equivalently, $b \Leftrightarrow c$; in the language of `while` programs, `if` b `then` c `else` $\overline{c}$) ensures that b and c have the same Boolean value just after execution of s.

To illustrate this technique, consider the simple program

$$\texttt{if}\ b\ \texttt{then}\ \{p\ ;\ q\}\ \texttt{else}\ \{p\ ;\ r\} \tag{15.1}$$

If the value of b were preserved by p, then we could rewrite this program more simply as

$$p\ ;\ \texttt{if}\ b\ \texttt{then}\ q\ \texttt{else}\ r \tag{15.2}$$

Formally, the assumption that the value of b is preserved by p takes the form of the commutativity condition $bp = pb$. By Lemma 15.1, we also have $\overline{b}p = p\overline{b}$. Expressed in the language of Kleene algebra, the equivalence of (15.1) and (15.2) under the assumption $bp = pb$ becomes the universal Horn formula

$$bp = pb \ \Rightarrow\ bpq + \overline{b}pr = p(bq + \overline{b}r).$$

The identity on the right-hand side can be established by simple equational reasoning:

$$\begin{aligned} p(bq + \overline{b}r) &= pbq + p\overline{b}r && \text{by distributivity} \\ &= bpq + \overline{b}pr && \text{by the assumption } bp = pb \text{ and its consequence } \overline{b}p = p\overline{b}. \end{aligned}$$

But what if b is not preserved by p? Here we can introduce a new atomic test c and atomic program s along with the commutativity condition $pc = cp$, intuitively modeling the idea that c tests a variable that is not modified by p, and insert the program $s\ ;\ bc + \overline{b}\overline{c}$.

We can now give a purely equational proof of the equivalence of the two programs

$$\begin{array}{ll} s\ ;\ bc + \overline{b}\overline{c}\ ; & s\ ;\ bc + \overline{b}\overline{c}\ ; \\ \texttt{if}\ b\ \texttt{then}\ \{p\ ;\ q\}\ \texttt{else}\ \{p\ ;\ r\} & p\ ;\ \texttt{if}\ c\ \texttt{then}\ q\ \texttt{else}\ r \end{array}$$

using the axioms of Kleene algebra with tests and the commutativity condition $pc = cp$. In fact, we can even do away with the atomic program s: if the two programs are equivalent without the precomputation s, then they are certainly equivalent with it.

Removing the leading s and expressing the two programs in the language of Kleene algebra, the equivalence problem becomes

$$(bc + \overline{b}\overline{c})(bpq + \overline{b}pr) = (bc + \overline{b}\overline{c})p(cq + \overline{c}r). \tag{15.3}$$

Using the distributive laws and the laws of Boolean algebra, we can simplify the left-hand side of (15.3) as follows:

$$\begin{aligned}(bc + \overline{b}\overline{c})(bpq + \overline{b}pr) &= bcbpq + \overline{b}\overline{c}bpq + bc\overline{b}pr + \overline{b}\overline{c}\overline{b}pr \\ &= bcpq + \overline{b}\overline{c}pr.\end{aligned}$$

The right-hand side of (15.3) simplifies in a similar fashion to the same expression:

$$\begin{aligned}(bc + \overline{b}\overline{c})p(cq + \overline{c}r) &= bcpcq + \overline{b}\overline{c}pcq + bcp\overline{c}r + \overline{b}\overline{c}p\overline{c}r \\ &= bccpq + \overline{b}\overline{c}cpq + bc\overline{c}pr + \overline{b}\overline{c}\overline{c}pr \\ &= bcpq + \overline{b}\overline{c}pr.\end{aligned}$$

Here the commutativity assumption is used in the second step.

Normal Form

Let us show now that any program can be transformed to a program in a special form, which we call *normal form*. A program is in *normal form* if it is of the form

p ; `while` b `do` q

where p and q are `while`-free. The subprogram p is called the *precomputation* of the normal form.

Now we can show that every `while` program, suitably augmented with finitely many new subprograms of the form s ; $bc + \overline{b}\overline{c}$, is equivalent to a `while` program in normal form, reasoning in Kleene algebra with tests under certain commutativity assumptions $cp = pc$.

This can be proved by induction on the structure of the program. Each programming construct accounts for one case in the inductive proof. For each case, we can give a transformation that moves an inner `while` loop to the outside and an equational proof of its correctness. The inductive construction is performed from the inside out; that is, smaller subprograms first.

We first show how to deal with a conditional test containing programs in normal form in its `then` and `else` clauses. Consider the program

`if` b `then` $\{p_1$; `while` d_1 `do` $q_1\}$
`else` $\{p_2$; `while` d_2 `do` $q_2\}$.

We introduce a new atomic program s and test c and assume that c commutes with p_1, p_2, q_1, and q_2.

Under these assumptions, we show that the programs

$$\begin{array}{l} s\ ;\ bc + \overline{b}\overline{c}\ ; \\ \texttt{if}\ b\ \texttt{then}\ \{p_1\ ;\ \texttt{while}\ d_1\ \texttt{do}\ q_1\} \\ \qquad \texttt{else}\ \{p_2\ ;\ \texttt{while}\ d_2\ \texttt{do}\ q_2\} \end{array} \tag{15.4}$$

and

$$\begin{array}{l} s\ ;\ bc + \overline{b}\overline{c}\ ; \\ \texttt{if}\ c\ \texttt{then}\ p_1\ \texttt{else}\ p_2\ ; \\ \texttt{while}\ cd_1 + \overline{c}d_2\ \texttt{do} \\ \quad \texttt{if}\ c\ \texttt{then}\ q_1\ \texttt{else}\ q_2 \end{array} \tag{15.5}$$

are equivalent. Note that if the two programs in the `then` and `else` clauses of (15.4) are in normal form, then (15.5) is in normal form. As previously mentioned, we can do away with the precomputation s.

Removing s and rewriting (15.4) in the language of Kleene algebra, we obtain

$$(bc + \overline{b}\overline{c})(bp_1(d_1q_1)^*\overline{d}_1 + \overline{b}p_2(d_2q_2)^*\overline{d}_2)$$

which reduces by distributivity and Boolean algebra to

$$bcp_1(d_1q_1)^*\overline{d}_1 + \overline{b}\overline{c}p_2(d_2q_2)^*\overline{d}_2. \tag{15.6}$$

Similarly, (15.5) becomes

$$(bc + \overline{b}\overline{c})(cp_1 + \overline{c}p_2)((cd_1 + \overline{c}d_2)(cq_1 + \overline{c}q_2))^*\overline{cd_1 + \overline{c}d_2}. \tag{15.7}$$

The subexpression $\overline{cd_1 + \overline{c}d_2}$ of (15.7) is equivalent by propositional reasoning to $c\overline{d}_1 + \overline{c}\overline{d}_2$. Here we have used the familiar propositional equivalence

$$cd_1 + \overline{c}d_2 = (\overline{c} + d_1)(c + d_2)$$

and a De Morgan law. The other subexpressions of (15.7) can be simplified using distributivity and Boolean algebra to obtain

$$(bcp_1 + \overline{b}\overline{c}p_2)(cd_1q_1 + \overline{c}d_2q_2)^*(c\overline{d}_1 + \overline{c}\overline{d}_2).$$

Using distributivity, this can be broken up into the sum of four terms:

$$bcp_1(cd_1q_1 + \overline{c}d_2q_2)^*c\overline{d}_1 \tag{15.8}$$

$$+\ bcp_1(cd_1q_1 + \overline{c}d_2q_2)^*\overline{c}\overline{d}_2 + \overline{b}\overline{c}p_2(cd_1q_1 + \overline{c}d_2q_2)^*c\overline{d}_1 \tag{15.9}$$

$$+\ \overline{b}\overline{c}p_2(cd_1q_1 + \overline{c}d_2q_2)^*\overline{c}\overline{d}_2. \tag{15.10}$$

Under the commutativity assumptions, the middle two terms (15.9) vanish; and for the remaining two terms (15.8) and (15.10), we have

$$bcp_1(cd_1q_1 + \overline{c}d_2q_2)^*c\overline{d}_1 = bcp_1(ccd_1q_1 + c\overline{c}d_2q_2)^*\overline{d}_1 = bcp_1(d_1q_1)^*\overline{d}_1$$
$$\overline{b}\overline{c}p_2(cd_1q_1 + \overline{c}d_2q_2)^*\overline{c}\overline{d}_2 = \overline{b}\overline{c}p_2(\overline{c}cd_1q_1 + \overline{c}\overline{c}d_2q_2)^*\overline{d}_2 = \overline{b}\overline{c}p_2(d_2q_2)^*\overline{d}_2.$$

The sum of these two terms is exactly (15.6).

Nested Loops

We next consider two nested `while` loops. This construction is particularly interesting in that no commutativity conditions (thus no extra variables) are needed, in contrast to the corresponding transformations of [48, 86], as reported in [45].

It can be shown that the program

```
while b do {
  p ; while c do q
}
```

is equivalent to the program

```
if b then {
  p ;
  while b + c do
    if c then q else p
}
```

or in the language of Kleene algebra,

$$(bp(cq)^*\overline{c})^*\overline{b} = bp((b+c)(cq+\overline{c}p))^*\overline{b+c} + \overline{b}. \tag{15.11}$$

The $\overline{b}$ on the far right-hand side is for the nonexistent `else` clause of the outermost conditional of the second program.

This construction transforms a pair of nested `while` loops to a single `while` loop inside a conditional test. After this transformation, the `while` loop can be taken outside the conditional using the previous transformation (this part does require a commutativity condition). A dummy normal form such as `1 ; while 0 do 1` can be supplied for the missing `else` clause.

We leave the proof of this equivalence as an exercise. Not surprisingly, the key property used in the proof is the denesting property $(p^*q)^*p^* = (p+q)^*$, which equates a regular expression of *-depth two with another of *-depth one.

Eliminating Postcomputations

We wish to show that a postcomputation occurring after a `while` loop can be absorbed into the `while` loop. Consider a program of the form

$$\{\texttt{while}\ b\ \texttt{do}\ p\}\ ;\ q. \tag{15.12}$$

We can assume without loss of generality that b, the test of the `while` loop, commutes with the postcomputation q. If not, introduce a new test c that commutes with q along with an atomic program s that intuitively sets the value of c to b, and insert $s\ ;\ bc + \overline{b}\overline{c}$ before the loop and in the loop after the body. We then claim that the two programs

$$s\ ;\ bc + \overline{b}\overline{c}\ ;\ \texttt{while}\ b\ \texttt{do}\ \{p\ ;\ s\ ;\ bc + \overline{b}\overline{c}\}\ ;\ q \tag{15.13}$$

$$s\ ;\ bc + \overline{b}\overline{c}\ ;\ \texttt{while}\ c\ \texttt{do}\ \{p\ ;\ s\ ;\ bc + \overline{b}\overline{c}\}\ ;\ q \tag{15.14}$$

are equivalent. This allows us to replace (15.13) with (15.14), for which the commutativity assumption holds.

For purposes of arguing that (15.13) and (15.14) are equivalent, we can omit all occurrences of s. The leading occurrences can be omitted as argued above, and the occurrences inside the `while` loops can be assumed to be part of p. The two trailing occurrences of q can be omitted as well. After making these modifications and writing the programs in the language of KAT, we need to show

$$(bc + \overline{b}\overline{c})(bp(bc + \overline{b}\overline{c}))^*\overline{b} = (bc + \overline{b}\overline{c})(cp(bc + \overline{b}\overline{c}))^*\overline{c}.$$

Using the sliding rule $p(qp)^* = (pq)^*p$ on both sides, this becomes

$$((bc + \overline{b}\overline{c})bp)^*(bc + \overline{b}\overline{c})\overline{b} = ((bc + \overline{b}\overline{c})cp)^*(bc + \overline{b}\overline{c})\overline{c}.$$

By distributivity and Boolean algebra, both sides reduce easily to $(bcp)^*\overline{b}\overline{c}$.

Under the assumption that b and q commute, we can show that (15.12) is equivalent to the program

$$\begin{array}{l}\texttt{if}\ \overline{b}\ \texttt{then}\ q\\ \quad\texttt{else while}\ b\ \texttt{do}\ \{\\ \quad\quad p\ ;\ \texttt{if}\ \overline{b}\ \texttt{then}\ q\\ \quad\}\end{array} \tag{15.15}$$

Note that if p and q are `while` free, then (15.15) consists of a program in normal form inside a conditional, which can be transformed to normal form using the transformation above.

We leave this equivalence as an exercise.

Composition

The composition of two programs in normal form

p_1 ; `while` b_1 `do` q_1 ;
p_2 ; `while` b_2 `do` q_2

can be transformed to a single program in normal form. First, we use the result of the previous section to absorb the `while`-free program p_2 into the first `while` loop. We can also ignore p_1, since it can be absorbed into the precomputation of the resulting normal form when we are done. It therefore suffices to show how to transform a program

`while` b `do` p ;
`while` c `do` q

to normal form, where p and q are `while`-free.

As already argued, we can assume without loss of generality that the test b commutes with the program q by introducing a new test if necessary. Since b also commutes with c by Boolean algebra, by a homework exercise we have that b commutes with the entire second `while` loop. This allows us to use the construction of the previous section to absorb the second `while` loop into the first. The resulting program looks like

```
if b̄ then while c do q
    else while b do {                                   (15.16)
        p ; if b̄ then while c do q
    }
```

Here we have just substituted `while` c `do` q for q in (15.15). At this point we can apply the conditional test transformation to the inner subprogram

`if` $\overline{b}$ `then while` c `do` q

using a dummy normal form for the omitted `else` clause, giving two nested loops in the `else` clause of (15.16); then denest the loops in the `else` clause of (15.16); finally, apply the conditional test transformation to the entire resulting program, yielding a program in normal form.

The transformations described above give a systematic method for moving `while` loops outside of any other programming construct. By applying these transformations inductively from the innermost loops outward, we can transform any program into a program in normal form.

None of these arguments needed an explicit assignment mechanism to Boolean variables, but only a dummy program s which does something

unspecified (and which more often than not can be omitted in the actual proof) and a guard of the form $bc + \overline{b}\overline{c}$ that says that b and c somehow obtained the same Boolean value. Of course, in a real implementation, s might assign the value of the test b to some new Boolean variable that is tested in c, but this does not play a role in equational proofs. The point is that it is not significant exactly how Boolean values are preserved across computations, but rather that they *can be* preserved; and for the purposes of formal verification, this fact is captured by a commutativity assumption.

Lecture 16

Models of KAT

In this lecture we show that the equational theories of KAT, KAT* (the star-continuous Kleene algebras with tests), and relational KATs coincide. We also introduce a family of language-theoretic models consisting of regular sets of *guarded strings*, which play the same role in KAT that the regular sets play in Kleene algebra. These results are from [75].

The Language of Kleene Algebra with Tests

Let P and B be disjoint finite sets of symbols. Elements of P are called *primitive actions* and elements of B are called *primitive tests*. *Action terms* and *Boolean terms* are defined inductively:

- any primitive action p is an action term
- any primitive test b is a Boolean term
- 0 and 1 are Boolean terms
- if p and q are action terms, then so are $p + q$, pq, and p^* (suitably parenthesized if necessary)
- if b and c are Boolean terms, then so are $b + c$, bc, and $\overline{b}$ (suitably parenthesized if necessary)

- any Boolean term is an action term.

Expressed as a grammar,

$$p ::= p \in \mathsf{P} \mid p + q \mid pq \mid p^* \mid b$$
$$b ::= a \in \mathsf{B} \mid 0 \mid 1 \mid b + c \mid bc \mid \overline{b}.$$

The set of all terms over P and B is denoted $\mathsf{Exp}\,\mathsf{P},\mathsf{B}$. The set of all Boolean terms over B is denoted T_{B}.

An *interpretation* over a Kleene algebra with tests K is any homomorphism (function commuting with the distinguished operations and constants) defined on $\mathsf{Exp}\,\mathsf{P},\mathsf{B}$ and taking values in K such that the Boolean terms are mapped to elements of the distinguished Boolean subalgebra.

If K is a Kleene algebra with tests and I is an interpretation over K, we write $K, I \vDash \varphi$ if the formula φ holds in K under the interpretation I according to the usual semantics of first-order logic. We write $\mathsf{KAT} \vDash \varphi$ (respectively, $\mathsf{KAT}^* \vDash \varphi$) if the formula φ is a logical consequence of the axioms of KAT (respectively, KAT*). The only formulas we consider are equations or equational implications (universal Horn formulas).

We write $\mathsf{KAT} \vDash \varphi$ if the formula φ is a logical consequence of KAT, that is, if φ holds under all interpretations over Kleene algebras with tests. We write $\mathsf{KAT}^* \vDash \varphi$ if φ holds under all interpretations over star-continuous Kleene algebras with tests.

Guarded Strings

Let P and B be disjoint finite sets of symbols. Our language-theoretic model of Kleene algebras with tests is based on the idea of *guarded strings* over P and B. Guarded strings were first studied in [55].

We obtain a guarded string from a string $x \in \mathsf{P}^*$ by inserting *atoms* interstitially among the symbols of x. An *atom* is a Boolean expression representing an atom (minimal nonzero element) of the free Boolean algebra on generators B.

Formally, an *atom* of $\mathsf{B} = \{b_1, \ldots, b_k\}$ is a product of literals $c_1 \cdots c_k$, where each $c_i \in \{b_i, \overline{b}_i\}$. This assumes an arbitrary but fixed order $b_1 < b_2 < \cdots < b_k$ on B; for technical reasons, we require the literals in an atom to occur in this order. There are exactly 2^k atoms, and they are in one-to-one correspondence with the truth assignmens to B. We denote atoms of B by $\alpha, \beta, \alpha_0, \ldots$. The set of all atoms of B is denoted At_{B}, or just At when B is understood. The set At will turn out to be the multiplicative identity of our language-theoretic model $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$.

If $b \in \mathsf{B}$ and α is an atom of B, we write $\alpha \leq b$ if b occurs positively in α and $\alpha \leq \overline{b}$ if b occurs negatively in α. This notation is consistent with the natural order in the free Boolean algebra generated by B.

Intuitively, the action symbols can be thought of as atomic instructions to be executed and atoms as conditions that must be satisfied at some point in the computation. If $\alpha \leq c_i$, then α asserts that c_i holds (and $\overline{c}_i$ fails) at that point in the computation.

A *guarded string* over P and B is any element of $(\mathsf{At} \cdot \mathsf{P})^* \cdot \mathsf{At}$; that is, any string

$$\alpha_0 p_1 \alpha_1 p_2 \alpha_2 \cdots \alpha_{n-1} p_n \alpha_n, \ n \geq 0,$$

where each $\alpha_i \in \mathsf{At}$ and each $p_i \in \mathsf{P}$. Note that every guarded string x begins and ends with an atom, which we call $\mathsf{first}\, x$ and $\mathsf{last}\, x$, respectively. Thus if x is the guarded string above, then $\mathsf{first}\, x = \alpha_0$ and $\mathsf{last}\, x = \alpha_n$. In the case $n = 0$, x just consists of a single atom α_0, and $\mathsf{first}\, x = \mathsf{last}\, x = \alpha_0$.

The set of all guarded strings over P and B is denoted $\mathsf{GS}_{\mathsf{P},\mathsf{B}}$, or just GS when P and B are understood.

Let $\overline{\mathsf{B}} = \{\overline{b} \mid b \in \mathsf{B}\}$. We denote strings in $(\mathsf{P} \cup \mathsf{B} \cup \overline{\mathsf{B}})^*$, including guarded strings, by the letters $x, y, z, x_1, \ldots$.

The analog of concatenation for guarded strings is *fusion product*, also sometimes called *coalesced product*. It is a *partial* binary operation $\diamond$ on GS defined as follows:

$$x\alpha \diamond \beta y = \begin{cases} x\alpha y, & \text{if } \alpha = \beta, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

That is, if $\mathsf{last}\, x = \mathsf{first}\, y$, then the fusion product $x \diamond y$ exists and is equal to the string obtained by concatenating x and y, but writing the common atom $\mathsf{last}\, x = \mathsf{first}\, y$ only once between them; otherwise, $x \diamond y$ is undefined.

For example, if $\mathsf{B} = \{b, c\}$ and $\mathsf{P} = \{p, q\}$, then

$$bcp\overline{b}c \diamond \overline{b}cq\overline{b}\overline{c} = bcp\overline{b}cq\overline{b}\overline{c}.$$

The operation $\diamond$ is associative in the sense that $(x \diamond y) \diamond z$ is defined iff $x \diamond (y \diamond z)$ is defined, and if both are defined, then they are equal.

If $A, B \subseteq \mathsf{GS}$, define

$$A \cdot B \stackrel{\text{def}}{=} \{x \diamond y \mid x \in A,\ y \in B,\ x \diamond y \text{ exists}\}.$$

We will tend to elide the $\cdot$ and write just AB. Thus AB consists of all existing fusion products of guarded strings in A with guarded strings in B. For example, if $\mathsf{B} = \{b, c\}$, $\mathsf{P} = \{p, q\}$, and

$$A = \{bcp\overline{b}\overline{c}, \overline{b}\overline{c}, bcqb\overline{c}\} \qquad\qquad B = \{\overline{b}\overline{c}pbc, \overline{b}\overline{c}, b\overline{c}qbc\},$$

then

$$AB = \{bcp\overline{b}\overline{c}pbc, bcp\overline{b}\overline{c}, \overline{b}\overline{c}pbc, \overline{b}\overline{c}, bcqb\overline{c}qbc\}.$$

Note that, whereas the operation $\diamond$ on guarded strings is a partial operation, the set-theoretic product $\cdot$ on sets of guarded strings defined in terms of $\diamond$ is a total operation. If there are no existing fusion products of strings from A and B, then $AB = \varnothing$. It is not difficult to show that $\cdot$ is associative, that it distributes over union, and that it has two-sided identity At.

For $A \subseteq \mathsf{GS}$, define inductively

$$A^0 \stackrel{\text{def}}{=} \mathsf{At} \qquad\qquad A^{n+1} \stackrel{\text{def}}{=} AA^n.$$

The asterate operation for sets of guarded strings is defined by

$$A^* \stackrel{\text{def}}{=} \bigcup_{n\geq 0} A^n.$$

Let $^-$ denote set complementation in At. That is, if $A \subseteq \mathsf{At}$, then $\overline{A} = \mathsf{At} - A$.

We now define a language-theoretic model $\mathsf{Reg\,P,B}$ based on guarded strings. The elements of $\mathsf{Reg\,P,B}$ will be the regular sets of guarded strings over P and B (although we have not yet defined *regular* in this context). We will also give a standard interpretation $G : \mathsf{Exp\,P,B} \to \mathsf{Reg\,P,B}$ analogous to the standard interpretation of regular expressions as regular sets.

Consider the structure

$$(2^{\mathsf{GS}}, 2^{\mathsf{At}}, \cup, \cdot, ^*, ^-, \varnothing, \mathsf{At}),$$

which we denote briefly by 2^{GS}. It is quite straightforward to verify that this is a star-continuous Kleene algebra with tests; that is, it is a model of KAT^*. The Boolean algebra axioms hold for 2^{At} because it is a set-theoretic Boolean algebra.

The star-continuity condition follows immediately from the definition of * and the distributivity of $\cdot$ over arbitrary union. Since

$$B^* = \bigcup_{n\geq 0} B^n,$$

we have that

$$AB^*C = A(\bigcup_{n\geq 0} B^n)C = \bigcup_{n\geq 0} AB^nC.$$

Both of these expressions denote the set

$$\bigcup_{n\geq 0} \{x \diamond y \diamond z \mid x \in A,\ y \in B^n,\ z \in C\}.$$

For $p \in \mathsf{P}$ and $b \in \mathsf{B}$, define

$$G(p) \stackrel{\text{def}}{=} \{\alpha p \beta \mid \alpha, \beta \in \mathsf{At}\} \qquad G(b) \stackrel{\text{def}}{=} \{\alpha \in \mathsf{At} \mid \alpha \leq b\}. \qquad (16.1)$$

The structure $\mathsf{Reg}\,\mathsf{P}, \mathsf{B}$ is defined to be the subalgebra of 2^{GS} generated by the elements $G(p)$ for $p \in \mathsf{P}$ and $G(b)$ for $b \in \mathsf{B}$. Elements of $\mathsf{Reg}\,\mathsf{P}, \mathsf{B}$ are called *regular sets*.

Standard Interpretation

The map G defined on primitive actions and primitive tests in (16.1) extends uniquely to a homomorphism $G : \mathsf{Exp}\,\mathsf{P}, \mathsf{B} \to \mathsf{Reg}\,\mathsf{P}, \mathsf{B}$:

$$G(p+q) \stackrel{\text{def}}{=} G(p) \cup G(q) \qquad G(1) \stackrel{\text{def}}{=} \mathsf{At} \qquad G(\overline{b}) \stackrel{\text{def}}{=} \mathsf{At} - G(b)$$
$$G(pq) \stackrel{\text{def}}{=} G(p)G(q) \qquad G(0) \stackrel{\text{def}}{=} \varnothing \qquad G(p^*) \stackrel{\text{def}}{=} G(p)^*.$$

The map G is called the *standard interpretation* over $\mathsf{Reg}\,\mathsf{P}, \mathsf{B}$.

Relational Models

Relational Kleene algebras with tests are interesting because they closely model our intuition about programs. In a relational model, the elements of K are binary relations and $\cdot$ is interpreted as relational composition. Elements of the Boolean subalgebra are subsets of the identity relation.

Formally, a *relational Kleene algebra with tests* on a set X is any structure

$$(K, B, \cup, ;, ^*, ^-, \varnothing, \mathsf{id})$$

such that

$$(K, \cup, ;, ^*, \varnothing, \mathsf{id})$$

is a relational Kleene algebra, that is, K is a family of binary relations on X, $;$ is ordinary relational composition, * is reflexive transitive closure, and id is the identity relation on X; and

$$(B, \cup, ;, ^-, \varnothing, \mathsf{id})$$

is a Boolean algebra of subsets of the identity relation id (not necessarily the whole powerset).

All relational Kleene algebras with tests are star-continuous. We write $\mathsf{REL} \models \varphi$ if the formula φ holds in all relational Kleene algebras in the usual sense of first-order logic.

Completeness of KAT* over Reg P, B

Now we show that an equation $p = q$ is a theorem of star-continuous Kleene algebra with tests iff it holds under the standard interpretation G over $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$, where P and B contain all primitive action and test symbols, respectively, appearing in p and q. Thus $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ is the free Kleene algebra with tests on generators P and B. In the next lecture, we will strengthen these results by removing the assumption of star-continuity.

Theorem 16.1 *Let $p, q \in \mathsf{Exp}\,\mathsf{P},\mathsf{B}$. Then*

$$\mathsf{KAT}^* \vDash p = q \Leftrightarrow G(p) = G(q).$$

Equivalently, $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ *is the free star-continuous Kleene algebra with tests on generators* P *and* B*.*

The forward implication is easy, since $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ is a star-continuous Kleene algebra. The converse is a consequence of the following lemma.

Lemma 16.2 *For any star-continuous Kleene algebra with tests K, interpretation $I : \mathsf{Exp}\,\mathsf{P},\mathsf{B} \to K$, and $p, q, r \in \mathsf{Exp}\,\mathsf{P},\mathsf{B}$,*

$$I(pqr) = \sup_{x \in G(q)} I(pxr)$$

where the supremum is with respect to the natural order in K. In particular,

$$I(q) = \sup_{x \in G(q)} I(x).$$

This result is analogous to the same result for Kleene algebras proved in Lecture **??** and the proof is similar. Note that the star-continuity axiom is a special case.

We are most interested in the second statement, but there is a slight subtlety that requires the stronger first statement as the induction hypothesis. In addition to the existence of the supremum, the more general statement provides a kind of infinite distributivity law over existing suprema. The need for this arises mainly in the induction case for $\cdot$.

Proof of Lemma 16.2. We proceed by induction on the structure of q. The basis consists of cases for primitive tests, primitive actions, 0 and 1. We argue the case for primitive actions and primitive tests explicitly.

For a primitive action $q \in \mathsf{P}$, recall that $G(q) = \{\alpha q \beta \mid \alpha, \beta \in \mathsf{At}\}$. Then

$$\begin{aligned} I(pqr) &= I(p)I(1)I(q)I(1)I(r) = \sup\{I(p)I(\alpha)I(q)I(\beta)I(r) \mid \alpha, \beta \in \mathsf{At}\} \\ &= \sup\{I(p\alpha q\beta r) \mid \alpha, \beta \in \mathsf{At}\} = \sup\{I(pxr) \mid x \in G(q)\}. \end{aligned}$$

Finite distributivity was used in the second step.

For a primitive test $b \in B$, recall that $G(b) = \{\alpha \mid \alpha \leq b\}$. Then

$$\begin{aligned} I(pbr) &= I(p)I(b)I(r) = \sup\{I(p)I(\alpha)I(r) \mid \alpha \leq b\} \\ &= \sup\{I(p\alpha r) \mid \alpha \leq b\} = \sup\{I(pxr) \mid x \in G(b)\}. \end{aligned}$$

Again, finite distributivity was used in the second step.

The induction step consists of cases for $+$, $\cdot$, *, and $^-$. The cases other than $\cdot$ and $^-$ are the same as in the proof of Theorem **??** of Lecture **??**.

For the case $\cdot$, recall that

$$G(qq') = G(q) \cdot G(q') = \{y \diamond z \mid y \in G(q),\ z \in G(q'),\ y \diamond z \text{ exists}\} = \{y\alpha z \mid y\alpha \in G(q),\ \alpha z \in G(q')\}.$$

Applying the induction hypothesis twice,

$$\begin{aligned} I(pqq'r) &= \sup\{I(pqvr) \mid v \in G(q')\} \\ &= \sup\{\sup\{I(puvr) \mid u \in G(q)\} \mid v \in G(q')\} \\ &= \sup\{I(puvr) \mid u \in G(q),\ v \in G(q')\}. \end{aligned}$$

The last step follows from a purely lattice-theoretic argument: if all the suprema in question on the left hand side exist, then the supremum on the right hand side exists and the two sides are equal.

Now

$$\begin{aligned} &\sup\{I(puvr) \mid u \in G(q),\ v \in G(q')\} \\ &\quad = \sup\{I(py\alpha\beta zr) \mid y\alpha \in G(q),\ \beta z \in G(q')\} \\ &\quad = \sup\{I(py\alpha\alpha zr) \mid y\alpha \in G(q),\ \alpha z \in G(q')\} \qquad (16.2) \\ &\quad = \sup\{I(py\alpha zr) \mid y\alpha \in G(q),\ \alpha z \in G(q')\} \\ &\quad = \sup\{I(pxr) \mid x \in G(qq')\}. \end{aligned}$$

The justification for step (16.2) is that if $\alpha \neq \beta$, then the product in K is 0 and does not contribute to the supremum.

For the case $^-$, recall that

$$G(\bar{b}) = \mathsf{At} - G(b) = \{\alpha \mid \alpha \not\leq b\} = \{\alpha \mid \alpha \leq \bar{b}\}.$$

Then

$$I(p\bar{b}r) = \sup\{I(p\alpha r) \mid \alpha \leq \bar{b}\} = \sup\{I(p\alpha r) \mid \alpha \in G(\bar{b})\}.$$

□

Proof of Theorem 16.1. If $\mathsf{KAT}^* \models p = q$ then $G(p) = G(q)$, since $\mathsf{Reg}\,\mathsf{P}, \mathsf{B}$ is a star-continuous Kleene algebra with tests. Conversely, if $G(p) = G(q)$, then by Lemma 16.2, for any star-continuous Kleene algebra with tests K and any interpretation I over K, $I(p) = I(q)$. Therefore $\mathsf{KAT}^* \models p = q$. □

Completeness over Relational Models

Finally we show completeness of KAT* over relational interpretations. It will suffice to construct a relational model isomorphic to $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$. This construction is similar to a construction we have seen before for Kleene algebra in Lecture **??** for regular sets.

For A any set of guarded strings, define

$$h(A) \stackrel{\text{def}}{=} \{(x, x \diamond y) \mid x \in \mathsf{GS},\ y \in A,\ x \diamond y \text{ exists}\}.$$

Lemma 16.3 *The language-theoretic model* 2^{GS} *and its submodel* $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ *are isomorphic to relational models.*

Proof. We show that the function $h : 2^{\mathsf{GS}} \to 2^{\mathsf{GS}\times\mathsf{GS}}$ defined above embeds 2^{GS} isomorphically onto a subalgebra of the Kleene algebra of all binary relations on GS.

It is straightforward to verify that h is a homomorphism:

$$h(A \cup B) = h(A) \cup h(B)$$

$$\begin{aligned} h(AB) &= \{(z, z \diamond r) \mid z \in \mathsf{GS},\ r \in AB,\ z \diamond r \text{ exists}\} \\ &= \{(z, z \diamond p \diamond q) \mid z \in \mathsf{GS},\ p \in A,\ q \in B,\ z \diamond p \diamond q \text{ exists}\} \\ &= \{(z, z \diamond p) \mid z \in \mathsf{GS},\ p \in A,\ z \diamond p \text{ exists}\} \mathbin{;} \{(z \diamond p, z \diamond p \diamond q) \mid z \in \mathsf{GS},\ p \in A,\ q \in B,\ z \diamond p \diamond q \text{ e} \\ &= \{(z, z \diamond p) \mid z \in \mathsf{GS},\ p \in A,\ z \diamond p \text{ exists}\} \mathbin{;} \{(y, y \diamond q) \mid y \in \mathsf{GS},\ q \in B,\ y \diamond q \text{ exists}\} \\ &= h(A) \mathbin{;} h(B) \end{aligned}$$

$$h(A^*) = h(\bigcup_{n\geq 0} A^n) = \bigcup_{n\geq 0} h(A)^n = h(A)^*$$

$$h(\mathsf{At}) = \{(x, x \diamond \alpha) \mid x \in \mathsf{GS},\ \alpha \in \mathsf{At},\ x \diamond \alpha \text{ exists}\} = \{(x, x) \mid x \in \mathsf{GS}\} = \mathsf{id} \quad h(0) = \varnothing$$

$$\begin{aligned} h(\overline{B}) &= h(\{\alpha \mid \alpha \notin B\}) \\ &= \{(x, x \diamond \alpha) \mid \alpha \notin B,\ x \diamond \alpha \text{ exists}\} \\ &= \{(y, y) \mid \mathsf{last}\, y \notin B\} \\ &= \mathsf{id} - \{(y, y) \mid \mathsf{last}\, y \in B\} \\ &= \mathsf{id} - h(B). \end{aligned}$$

The function h is injective, since A can be uniquely recovered from $h(A)$:

$$A = \{y \mid \exists \alpha\ (\alpha, y) \in h(A)\}.$$

The submodel $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ is perforce isomorphic to a relational model on GS, namely the image of $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ under h. □

Combining Theorem 16.1, Lemma 16.3, and the fact that all relational models are star-continuous Kleene algebras with tests, we have

Theorem 16.4 *Let* REL *denote the class of all relational Kleene algebras with tests. Let* $p, q \in \mathsf{Exp}\,\mathsf{P}, \mathsf{B}$. *The following are equivalent:*

(i) $\mathsf{KAT}^* \models p = q$

(ii) $G(p) = G(q)$

(iii) $\mathsf{REL} \models p = q$.

In the next lecture we will remove the assumption of star-continuity and show that the statement $\mathsf{KAT} \models p = q$ can be added to this list. Thus KAT is complete for the equational theory of relational models and $\mathsf{Reg}\,\mathsf{P}, \mathsf{B}$ forms the free KAT on generators P and B. This result is analogous to the completeness result of Lecture **??**, which states that the regular sets over a finite alphabet P form the free Kleene algebra on generators P.

Lecture 17

Completeness of KAT

In this lecture we show that the equational theories of the Kleene algebras with tests and the star-continuous Kleene algebras with tests coincide. Combined with the results of the previous lecture, this says that KAT is complete for the equational theory of relational models and $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ forms the free KAT on generators P and B. This result is analogous to the completeness result of Lecture **??**, which states that the regular sets over a finite alphabet P form the free Kleene algebra on generators P. The results of this lecture are from [75].

Theorem 17.1 *Let* REL *denote the class of all relational Kleene algebras with tests. Let* $p, q \in \mathsf{Exp}\,\mathsf{P},\mathsf{B}$. *The following are equivalent:*

(i) $\mathsf{KAT} \models p = q$

(ii) $\mathsf{KAT}^* \models p = q$

(iii) $G(p) = G(q)$

(iv) $\mathsf{REL} \models p = q$.

The statements (ii)–(iv) were shown to be equivalent in Theorem **??** of Lecture **??**. Here we have added (i) to the list. Thus, for the purpose of deriving identities, the infinitary star-continuity condition provides no extra power over the finitary axiomatization KAT. However, it does entail

more Horn formulas (equational implications). Note that $\mathsf{KAT} \models p = q$ iff $\mathsf{KAT} \vdash p = q$ and $\mathsf{KAT}^* \models p = q$ iff $\mathsf{KAT}^* \vdash p = q$ by the completeness of equational logic.

One possible approach might be to modify the completeness proof of Lecture **??** for KA to handle tests. We take a different approach here, showing that every term p can be transformed into a KAT-equivalent term $\widehat{p}$ such that $G(\widehat{p})$, the set of guarded strings represented by $\widehat{p}$, is the same as $R(\widehat{p})$, the set of strings represented by $\widehat{p}$ under the ordinary interpretation of regular expressions. The Boolean algebra axioms are not needed in equivalence proofs involving such terms, so we can apply the completeness result of Lecture **??** directly. This idea is ultimately due to Kaplan [55].

Consider the set $\overline{\mathsf{B}} = \{\overline{b} \mid b \in \mathsf{B}\}$, the set of negated atomic tests. We can view $\overline{\mathsf{B}}$ as a separate set of primitive symbols disjoint from B and P. Using the DeMorgan laws and the law $\overline{\overline{b}} = b$ of Boolean algebra, every term p can be transformed to a KAT-equivalent term p' in which $\overline{\ }$ is applied only to primitive test symbols, thus we can view p' as a regular expression over the alphabet $\mathsf{P} \cup \mathsf{B} \cup \overline{\mathsf{B}}$. As such, it represents a set of strings

$$R(p') \subseteq (\mathsf{P} \cup \mathsf{B} \cup \overline{\mathsf{B}})^*$$

under the standard interpretation R of regular expressions as regular sets.

In general, the sets $R(p')$ and $G(p')$ may differ. For example, $R(q) = \{q\}$ for primitive action q, but $G(q) = \{\alpha q \beta \mid \alpha, \beta \in \mathsf{At}\}$.

Our main task will be to show how to further transform p' to another KAT-equivalent string $\widehat{p}$ such that all elements of $R(\widehat{p})$ are guarded strings and $R(\widehat{p}) = G(\widehat{p})$. We can then use the completeness result of [64], since p and q will be KAT-equivalent iff $\widehat{p}$ and $\widehat{q}$ are equivalent as regular expressions over $\mathsf{P} \cup \mathsf{B} \cup \overline{\mathsf{B}}$; that is, if they can be proved equivalent in pure Kleene algebra.

Example 17.2 *Consider the two terms*

$$p = (q + b + \overline{b})^* br \qquad\qquad \widehat{p} = (bq + \overline{b}q)^* br(b + \overline{b}),$$

where $\mathsf{P} = \{q, r\}$ *and* $\mathsf{B} = \{b\}$. *There are certainly strings in* $R(p)$, $qq\overline{b}bbqbr$ *for example, that are not guarded strings. However,* p *and* $\widehat{p}$ *represent the same set of guarded strings under the interpretation* G, *and all strings in* $R(\widehat{p})$ *are guarded strings; that is,* $G(p) = G(\widehat{p}) = R(\widehat{p})$.

In our inductive proof, it will be helpful to maintain terms in the following special form. Call a term *externally guarded* if it is of the form α or $\alpha q \beta$, where α and β are atoms of B. For an externally guarded term $\alpha q \beta$, let $\mathsf{first}\, p = \alpha$ and $\mathsf{last}\, p = \beta$. For an externally guarded term α, define

first p = last $p = \alpha$. Define a special multiplication operation $\diamond$ on externally guarded terms as follows:

$$r\alpha \diamond \beta s \stackrel{\text{def}}{=} \begin{cases} r\alpha s, & \text{if } \alpha = \beta, \\ 0, & \text{if } \alpha \neq \beta. \end{cases}$$

This is much like fusion product on guarded strings as defined in Lecture **??**, except that for incompatible pairs of guarded strings, fusion product is undefined, whereas here $\diamond$ is defined and has value 0.

For any two externally guarded terms q and r, $q \diamond r$ is externally guarded, and $q \diamond r = qr$ is a theorem of KAT; in particular,

$$G(q \diamond r) = G(q) \cdot G(r) = G(qr).$$

If $\sum_i q_i$ and $\sum_j r_j$ are sums of zero or more externally guarded terms, define

$$(\sum_i q_i) \diamond (\sum_j r_j) \stackrel{\text{def}}{=} \sum_{i,j} q_i \diamond r_j.$$

As above, for any two sums q and r of externally guarded terms, $q \diamond r = qr$ is a theorem of KAT; in particular,

$$G(q \diamond r) = G(q) \cdot G(r) = G(qr),$$

and $q \diamond r$ is a sum of externally guarded terms.

Lemma 17.3 *For every term p, there is a term $\widehat{p}$ such that*

(i) $\mathsf{KAT} \vDash p = \widehat{p}$

(ii) $R(\widehat{p}) = G(\widehat{p})$

(iii) *$\widehat{p}$ is a sum of zero or more externally guarded terms.*

Proof. As argued above, we can assume without loss of generality that all occurrences of $^-$ in p are applied to primitive tests only, thus we may view p as a term over the alphabet $\mathsf{P} \cup \mathsf{B} \cup \overline{\mathsf{B}}$.

We define $\widehat{p}$ by induction on the structure of p. For the basis, take

$$\widehat{p} \stackrel{\text{def}}{=} \sum_{\alpha,\beta \in \mathsf{At}} \alpha p \beta, \quad p \in \mathsf{P} \qquad\qquad \widehat{1} \stackrel{\text{def}}{=} \sum_{\alpha \in \mathsf{At}} \alpha$$

$$\widehat{b} \stackrel{\text{def}}{=} \sum_{\alpha \leq b} \alpha, \quad b \in \mathsf{B} \cup \overline{\mathsf{B}} \qquad\qquad \widehat{0} \stackrel{\text{def}}{=} 0.$$

In each of these cases, it is straightforward to verify (i), (ii), and (iii).

For the induction step, suppose we have terms $\widehat{p}$ and $\widehat{q}$ satisfying (ii) and (iii). We take

$$\widehat{p+q} \stackrel{\text{def}}{=} \widehat{p} + \widehat{q} \qquad\qquad \widehat{pq} \stackrel{\text{def}}{=} \widehat{p} \diamond \widehat{q}.$$

These constructions are easily shown to satisfy (i), (ii), and (iii).

It remains to construct $\widehat{p^*}$. We proceed by induction on the number of externally guarded terms in the sum p. For the basis, we define

$$\begin{aligned}
\widehat{0^*} &\stackrel{\text{def}}{=} \widehat{1} \\
\widehat{\alpha^*} &\stackrel{\text{def}}{=} \widehat{1} \\
\widehat{(\alpha q \beta)^*} &\stackrel{\text{def}}{=} \begin{cases} \widehat{1} + \alpha q \beta, & \text{if } \alpha \neq \beta, \\ \widehat{1} + \alpha q (\alpha q)^* \alpha, & \text{if } \alpha = \beta. \end{cases}
\end{aligned} \tag{17.1}$$

For the induction step, let $p = q + r$, where r is an externally guarded term and q is a sum of externally guarded terms, one fewer in number than in p. By the induction hypothesis, we can construct

$$\widehat{q^*} = \sum_i \alpha_i q_i \beta_i$$

with the desired properties. Suppose that $\mathsf{first}\, r = \alpha$. Then $\mathsf{KAT} \vDash r = \alpha r$. Moreover, the following equations are provable in KAT:

$$r\widehat{q^*}\alpha = r(\sum_i \alpha_i q_i \beta_i)\alpha = \sum_i (r \diamond \alpha_i q_i \beta_i \diamond \alpha) = \sum_{\substack{\mathsf{last}\, r = \alpha_i \\ \beta_i = \alpha}} r q_i \alpha = r(\sum_{\substack{\mathsf{last}\, r = \alpha_i \\ \beta_i = \alpha}} q_i)\alpha,$$

and the expression on the right-hand side is externally guarded and satisfies (ii). We can therefore apply (17.1) to this expression, yielding an expression q' equivalent to $(r\widehat{q^*}\alpha)^*$ and satisfying (ii) and (iii).

Now reasoning in KAT,

$$\begin{aligned}
p^* &= (q + r)^* \\
&= q^*(rq^*)^* && \text{by the denesting rule} \\
&= q^* + q^* r q^* (rq^*)^* && \text{by unwinding and distributivity} \\
&= q^* + q^* r q^* (\alpha r q^*)^* \\
&= q^* + q^* (r q^* \alpha)^* r q^* && \text{by the sliding rule} \\
&= \widehat{q^*} + \widehat{q^*} \diamond q' \diamond r \diamond \widehat{q^*},
\end{aligned}$$

which is of the desired form. □

The next theorem shows that the equational theories of the Kleene algebras with tests and the star-continuous Kleene algebras with tests coincide.

Theorem 17.4 $\mathsf{KAT} \models p = q \Leftrightarrow G(p) = G(q)$.

Proof. The forward implication is immediate, since $\mathsf{Reg}\,\mathsf{P}, \mathsf{B}$ is a Kleene algebra with tests.

For the reverse implication, suppose $G(p) = G(q)$. By Lemma 17.3,

$$\mathsf{KAT} \vdash p = \widehat{p} \qquad R(\widehat{p}) = G(\widehat{p}) \qquad \mathsf{KAT} \vdash q = \widehat{q} \qquad R(\widehat{q}) = G(\widehat{q}).$$

By the soundness of the KAT axioms,

$$R(\widehat{p}) = G(\widehat{p}) = G(p) = G(q) = G(\widehat{q}) = R(\widehat{q}).$$

By the completeness theorem for KA (Theorem **??**, Lecture **??**), $\mathsf{KA} \vdash \widehat{p} = \widehat{q}$. Combining this with the proofs $\mathsf{KAT} \vdash p = \widehat{p}$ and $\mathsf{KAT} \vdash q = \widehat{q}$, we have $\mathsf{KAT} \models p = q$. □

Since we have shown that the equational theories of the Kleene algebras with tests and the star-continuous Kleene algebras with tests coincide, we can henceforth write $\models p = q$ unambiguously in place of $\mathsf{KAT}^* \models p = q$ or $\mathsf{KAT} \models p = q$.

Decidability

Once we have Theorem 17.1, the decidability of the equational theory of Kleene algebra with tests follows almost immediately from a simple reduction to Propositional Dynamic Logic (PDL). Any term in the language of KAT is a program of PDL (after replacing Boolean terms b with PDL tests $b?$), and it is known that two such terms p and q represent the same binary relation in all relational structures iff

$$\mathsf{PDL} \models \langle p\rangle c \Leftrightarrow \langle q\rangle c,$$

where c is a new primitive proposition symbol [36] (see [46]). By Theorems 17.1 and 17.4, this is tantamount to deciding KAT-equivalence.

PDL is known to be exponential time complete [36, 99], thus the equational theory of KAT is decidable in no more than exponential time. It is at least PSPACE-hard, since the equational theory of Kleene algebra is [113]. We will show in Lecture **??** by different methods that the equational theory of KAT is PSPACE-complete.

Lecture 18

KAT and Hoare Logic

In this lecture and the next we show that KAT subsumes propositional Hoare logic (PHL). Thus the specialized syntax and deductive apparatus of Hoare logic are inessential and can be replaced by simple equational reasoning. Moreover, all relationally valid Hoare-style inference rules are derivable in KAT (this is false for PHL). The results of this lecture are from [70, 75]. In a future lecture we will show that deciding the relational validity of such rules is PSPACE-complete.

Hoare logic, introduced by C. A. R. Hoare in 1969 [49], was the first formal system for the specification and verification of well-structured programs. This pioneering work initiated the field of program correctness and inspired hundreds of technical articles [22, 31, 32]. For this achievement among others, Hoare received the Turing Award in 1980. A comprehensive introduction to Hoare logic can be found in [32].

Hoare logic uses a specialized syntax involving *partial correctness assertions* (PCAs) of the form $\{b\}\,p\,\{c\}$ and a deductive apparatus consisting of a system of specialized rules of inference. Under certain conditions, these rules are relatively complete [31]; essentially, the propositional fragment of the logic can be used to reduce partial correctness assertions to static assertions about the underlying domain of computation.

The propositional fragment of Hoare logic, called *propositional Hoare logic* (PHL), is subsumed by KAT. The reduction transforms PCAs to ordinary equations and the specialized rules of inference to equational impli-

cations (universal Horn formulas). The transformed rules are all derivable in KAT by pure equational reasoning. More generally, all Hoare-style inference rules of the form

$$\frac{\{b_1\}\,p_1\,\{c_1\} \qquad \dots \qquad \{b_n\}\,p_n\,\{c_n\}}{\{b\}\,p\,\{c\}} \tag{18.1}$$

that are valid over relational models are derivable in KAT; this is trivially false for PHL.

Encoding of While Programs and Partial Correctness Assertions

The encoding of the `while` programming constructs using the regular operators and tests originated with propositional dynamic logic (PDL) [36]. KAT is strictly less expressive than PDL, but is simpler and purely equational, and is only PSPACE-complete, whereas PDL is EXPTIME-complete. In addition, PDL interpretations are restricted to relational models.

Halpern and Reif [43] prove PSPACE-completeness of strict deterministic PDL, but neither the upper nor the lower bound of the KAT PSPACE-completeness result follows from theirs. Not only are PDL semantics restricted to relational models, but the arguments of [43] depend on an additional nonalgebraic restriction: the relations interpreting atomic programs must be single-valued. Without this restriction, even if only `while` programs are allowed, PDL is EXPTIME-hard. In contrast, KAT imposes no such restrictions.

Hoare Logic

A common choice of programming language in Hoare logic is the language of `while` programs. The first-order version of this language contains a simple assignment $x := e$, conditional test `if` b `then` p `else` q, sequential composition $p\,;\,q$, and a looping construct `while` b `do` p.

The encoding of the `while` program constructs in KAT is as in PDL [36]:

$$p\,;\,q \stackrel{\text{def}}{=} pq \tag{18.2}$$

$$\texttt{if}\ b\ \texttt{then}\ p\ \texttt{else}\ q \stackrel{\text{def}}{=} bp + \overline{b}q \tag{18.3}$$

$$\texttt{while}\ b\ \texttt{do}\ p \stackrel{\text{def}}{=} (bp)^*\overline{b}. \tag{18.4}$$

The basic assertion of Hoare logic is the *partial correctness assertion* (PCA)

$$\{b\}\,p\,\{c\}, \tag{18.5}$$

where b and c are formulas and p is a program. Intuitively, this statement asserts that whenever b holds before the execution of the program p, then if and when p halts, c is guaranteed to hold of the output state. It does not assert that p must halt.

For applications in program verification, the standard interpretation would be a Kleene algebra of binary relations on a set and the Boolean algebra of subsets of the identity relation.

Semantically, programs p in Hoare logic and dynamic logic (DL) are usually interpreted as binary input/output relations p^M on a domain of computation M, and assertions are interpreted as subsets of M [31, 98]. The definition of the relation p^M is inductive on the structure of p; for example, $(p\ ;\ q)^M = p^M\ ;\ q^M$, the ordinary relational composition of the relations corresponding to p and q. The meaning of the PCA (18.5) is the same as the meaning of the DL formula $b \Rightarrow [p]c$, where $\Rightarrow$ is ordinary propositional implication and the modal construct $[p]c$ is interpreted in the model M as the set of states s such that for all $(s,t) \in p^M$, the output state t satisfies c.

Hoare logic provides a system of specialized rules for deriving valid PCAs, one rule for each programming construct. The verification process is inductive on the structure of programs. The traditional Hoare inference rules are:

Assignment rule

$$\{b[e/x]\}\, x := e\, \{b\} \tag{18.6}$$

Composition rule

$$\frac{\{b\}\, p\, \{c\} \qquad \{c\}\, q\, \{d\}}{\{b\}\, p\ ;\ q\, \{d\}} \tag{18.7}$$

Conditional rule

$$\frac{\{b \wedge c\}\, p\, \{d\} \qquad \{\neg b \wedge c\}\, q\, \{d\}}{\{c\}\, \texttt{if}\ b\ \texttt{then}\ p\ \texttt{else}\ q\, \{d\}} \tag{18.8}$$

While rule

$$\frac{\{b \wedge c\}\, p\, \{c\}}{\{c\}\, \texttt{while}\ b\ \texttt{do}\ p\, \{\neg b \wedge c\}} \tag{18.9}$$

Weakening rule

$$\frac{b' \Rightarrow b \qquad \{b\}\, p\, \{c\} \qquad c \Rightarrow c'}{\{b'\}\, p\, \{c'\}} \tag{18.10}$$

Cook [31] showed that these rules are complete relative to first-order number theory when interpreted over the structure of arithmetic $\mathbb{N}$.

Propositional Hoare logic (PHL) consists of atomic proposition and program symbols, the usual propositional connectives, `while` program constructs, and PCAs built from these. Atomic programs are interpreted as binary relations on a set M and atomic propositions are interpreted as subsets of M. The deduction system of PHL consists of the composition, conditional, while, and weakening rules (18.7)–(18.10) and propositional logic. The assignment rule (18.6) is omitted, since there is no first-order relational structure over which to interpret program variables; in practice, its role is played by PCAs over atomic programs that are postulated as assumptions.

In PHL, we are concerned with the problem of determining the validity of rules of the form

$$\frac{\{b_1\}\, p_1\, \{c_1\} \qquad \dots \qquad \{b_n\}\, p_n\, \{c_n\}}{\{b\}\, p\, \{c\}} \tag{18.11}$$

over relational interpretations. The premises $\{b_i\}\, p_i\, \{c_i\}$ take the place of the assignment rule (18.6) and are an essential part of the formulation.

Encoding Hoare Logic in KAT

The propositional Hoare rules can be derived as theorems of KAT. The PCA $\{b\}\, p\, \{c\}$ is encoded in KAT in any one of the following four equivalent ways:

Lemma 18.1 *The following are equivalent:*

(i) $bp\bar{c} = 0$

(ii) $bp = bpc$

(iii) $bp \leq bpc$

(iv) $bp \leq pc$

Proof. Miscellaneous Exercise **??**. □

Intuitively, the formulation (i) says that the program p with preguard b and postguard $\bar{c}$ has no halting execution, and the formulation (ii) says that testing c after executing bp is always redundant.

Using the encoding of `while` programs (18.2)–(18.4) and Lemma 18.1(ii), the Hoare rules (18.7)–(18.10) take the following form:

Composition rule $bp = bpc \wedge cq = cqd \Rightarrow bpq = bpqd$ (18.12)

Conditional rule $bcp = bcpd \wedge \overline{b}cq = \overline{b}cqd \Rightarrow c(bp + \overline{b}q) = c(bp + \overline{b}q)d$ (18.13)

While rule $bcp = bcpc \Rightarrow c(bp)^*\overline{b} = c(bp)^*\overline{b}\,\overline{b}c$ (18.14)

Weakening rule $b' \leq b \wedge bp = bpc \wedge c \leq c' \Rightarrow b'p = b'pc'.$ (18.15)

These implications are to be interpreted as universal Horn formulas; that is, the variables are implicitly universally quantified. To establish the adequacy of the translation, we show that the KAT encodings (18.12)–(18.15) of the Hoare rules (18.7)–(18.10) are theorems of KAT.

Theorem 18.2 *The universal Horn formulas* (18.12)–(18.15) *are theorems of* KAT.

Proof. First we derive the composition rule (18.12). Assuming the premises

$$bp = bpc \tag{18.16}$$
$$cq = cqd, \tag{18.17}$$

we have

$$\begin{aligned} bpq &= bpcq && \text{by (18.16)} \\ &= bpcqd && \text{by (18.17)} \\ &= bpqd && \text{by (18.16).} \end{aligned}$$

Thus the implication (18.12) holds.

For the conditional rule (18.13), assume the premises

$$bcp = bcpd \tag{18.18}$$
$$\overline{b}cq = \overline{b}cqd. \tag{18.19}$$

Then

$$\begin{aligned} c(bp + \overline{b}q) &= cbp + c\overline{b}q && \text{by distributivity} \\ &= bcp + \overline{b}cq && \text{by commutativity of tests} \\ &= bcpd + \overline{b}cqd && \text{by (18.18) and (18.19)} \\ &= cbpd + c\overline{b}qd && \text{by commutativity of tests} \\ &= c(bp + \overline{b}q)d && \text{by distributivity.} \end{aligned}$$

For the `while` rule (18.14), by trivial simplifications it suffices to show

$$cbp \leq cbpc \Rightarrow c(bp)^* \leq c(bp)^*c.$$

Assume

$$cbp \leq cbpc. \tag{18.20}$$

By an axiom of KA, we need only show

$$c + c(bp)^*cbp \leq c(bp)^*c.$$

But

$$\begin{aligned}
c + c(bp)^*cbp &\leq c + c(bp)^*cbpc && \text{by (18.20) and monotonicity}\\
&\leq c1c + c(bp)^*cbpc && \text{by Boolean algebra}\\
&\leq c(1 + (bp)^*cbp)c && \text{by distributivity}\\
&\leq c(1 + (bp)^*bp)c && \text{by monotonicity}\\
&\leq c(bp)^*c && \text{by unwinding.}
\end{aligned}$$

Finally, for the weakening rule (18.15), we can rewrite the rule as

$$b' \leq b \wedge bp\overline{c} = 0 \wedge \overline{c}' \leq \overline{c} \Rightarrow b'p\overline{c}' = 0,$$

which follows immediately from the monotonicity of multiplication. □

Lecture 19

Completeness of KAT for the Hoare Theory of Relational Models

Theorem 18.2 of Lecture 18 says that for any proof rule of PHL, or more generally, for any rule of the form

$$\frac{\{b_1\}\, p_1\, \{c_1\} \qquad \dots \qquad \{b_n\}\, p_n\, \{c_n\}}{\{b\}\, p\, \{c\}}$$

derivable in PHL, the corresponding equational implication (universal Horn formula)

$$b_1 p_1 \bar{c}_1 = 0 \wedge \cdots \wedge b_n p_n \bar{c}_n = 0 \Rightarrow b p \bar{c} = 0 \tag{19.1}$$

is a theorem of KAT. In this lecture we strengthen this result to show (Corollary 19.2) that *all* universal Horn formulas of the form

$$r_1 = 0 \wedge \cdots \wedge r_n = 0 \Rightarrow p = q \tag{19.2}$$

that are relationally valid (true in all relational models) are theorems of KAT; in other words, KAT is complete for universal Horn formulas of the form (19.2) over relational interpretations. This result subsumes Theorem 18.2 of Lecture 18, since the Hoare rules are relationally valid. Corollary

19.2 is trivially false for PHL; for example, the rule

$$\frac{\{c\}\,\texttt{if } b \texttt{ then } p \texttt{ else } p\,\{c\}}{\{c\}\,p\,\{c\}}$$

cannot be proved in PHL for the simple reason that the Hoare rules only increase the length of programs. The results of this lecture are from [70].

To prove this result we generalize Cohen's theorem (Lecture **??**) in two ways: to handle tests and to show completeness over relational models. The deductive completeness of KAT over relationally valid formulas of the form (19.2) will follow as a corollary.

Let $\mathsf{Exp}\,\mathsf{P},\mathsf{B}$ denote the set of terms of the language of KAT over primitive propositions $\mathsf{P} = \{p_1,\ldots,p_m\}$ and primitive tests $\mathsf{B} = \{b_1,\ldots,b_k\}$. Let $r_1,\ldots,r_n,p,q \in \mathsf{Exp}\,\mathsf{P},\mathsf{B}$. Let $\top$ be the *universal expression*

$$\top = (p_1 + \cdots + p_m)^*$$

and let $r = \sum_i r_i$. The formula (19.2) is equivalent to $r = 0 \Rightarrow p = q$.

Recall the definition of the algebra $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ of regular sets of guarded strings over P,B and the standard interpretation $G : \mathsf{Exp}\,\mathsf{P},\mathsf{B} \to \mathsf{Reg}\,\mathsf{P},\mathsf{B}$ from Lecture **??**. We showed in Lecture **??** that $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$ is the free KAT on generators P,B in the sense that for any terms $s,t \in \mathsf{Exp}\,\mathsf{P},\mathsf{B}$,

$$\vDash s = t \Leftrightarrow G(s) = G(t). \tag{19.3}$$

Note that $G(\top) = \mathsf{GS}$, the set of all guarded strings over P,B.

Theorem 19.1 *The following four conditions are equivalent:*

(i) $\mathsf{KAT} \vDash r = 0 \Rightarrow p = q$

(ii) $\mathsf{KAT}^* \vDash r = 0 \Rightarrow p = q$

(iii) $\mathsf{REL} \vDash r = 0 \Rightarrow p = q$

(iv) $\vDash\ p + \top r \top = q + \top r \top$.

It does not matter whether (iv) is preceded by KAT, KAT^*, or REL, since the equational theories of these classes coincide, as shown in Lecture **??**.

Proof. Since $\mathsf{REL} \subseteq \mathsf{KAT}^* \subseteq \mathsf{KAT}$, the implications (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) hold trivially. Also, it is clear that

$$\mathsf{KAT} \vDash p + \top r \top = q + \top r \top \Rightarrow (r = 0 \Rightarrow p = q),$$

therefore (iv) $\Rightarrow$ (i) as well. It thus remains to show that (iii) $\Rightarrow$ (iv). Writing equations as pairs of inequalities, we wish to show

$$\text{if} \quad \mathsf{REL} \vDash r = 0 \Rightarrow p \le q \quad \text{then} \quad \vDash p \le q + \top r \top. \tag{19.4}$$

To show (19.4), we construct a relational interpretation M on states $G(\top) - G(\top r \top)$, where G is the standard interpretation of expressions as sets of guarded strings. Note that if $x, y, z \in G(\top)$ such that $x \diamond y \diamond z \in G(\top) - G(\top r \top)$, then $y \in G(\top) - G(\top r \top)$. If $G(\top) \subseteq G(\top r \top)$, then we are done, since in that case $G(p) \subseteq G(\top) \subseteq G(\top r \top)$ and the right-hand side of (19.4) follows immediately from (19.3). Similarly, if $G(1) \subseteq G(\top r \top)$, then $G(\top) \subseteq G(u \top r \top) \subseteq G(\top r \top)$ and the same argument applies. We can therefore assume without loss of generality that both $G(\top) - G(\top r \top)$ and $G(1) - G(\top r \top)$ are nonempty.

The atomic symbols are interpreted in M as follows:

$$M(p) \stackrel{\text{def}}{=} \{(x, xp\beta) \mid xp\beta \in G(\top) - G(\top r \top)\}, \quad p \in \mathsf{P}$$

$$M(b) \stackrel{\text{def}}{=} \{(x, x) \mid x \in G(\top) - G(\top r \top),\ \mathsf{last}\, x \leq b\}, \quad b \in \mathsf{B}.$$

The interpretations $M(p)$ of KAT expressions p as binary relations are defined inductively in the standard way for relational models.

We now show that for any $e \in \mathsf{Exp}\,\mathsf{P}, \mathsf{B}$,

$$M(e) = \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(e)\} \tag{19.5}$$

by induction on the structure of e. For primitive programs p and tests b,

$$\begin{aligned} M(p) &= \{(x, xp\beta) \mid xp\beta \in G(\top) - G(\top r \top)\} \\ &= \{(x, x \diamond \alpha p \beta) \mid x \diamond \alpha p \beta \in G(\top) - G(\top r \top)\} \\ &= \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(p)\}, \\ M(b) &= \{(x, x) \mid x \in G(\top) - G(\top r \top),\ \mathsf{last}\, x \in G(b)\} \\ &= \{(x, x \diamond \beta) \mid x \diamond \beta \in G(\top) - G(\top r \top),\ \beta \in G(b)\} \\ &= \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(b)\}. \end{aligned}$$

For the constants 0 and 1, we have

$$\begin{aligned} M(0) &= \varnothing = \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(0)\}, \\ M(1) &= \{(x, x) \mid x \in G(\top) - G(\top r \top)\} = \{(x, x \diamond \beta) \mid x \diamond \beta \in G(\top) - G(\top r \top),\ \beta \in G(1)\}. \end{aligned}$$

For compound expressions,

$$\begin{aligned} M(s + t) &= M(s) \cup M(t) \\ &= \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(s)\} \cup \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \\ &= \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(s) \cup G(t)\} \\ &= \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(s + t)\} \end{aligned}$$

$$\begin{aligned} M(st) &= M(s)\ ;\ M(t) \\ &= \{(x, x \diamond z) \mid x \diamond z \in G(\top) - G(\top r \top),\ z \in G(s)\}\ ;\ \{(y, y \diamond w) \mid y \diamond w \in G(\top) - G(\top r \top),\ w \in \\ &= \{(x, x \diamond z \diamond w) \mid x \diamond z \diamond w \in G(\top) - G(\top r \top),\ z \in G(s),\ w \in G(t)\} \\ &= \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(st)\} \end{aligned}$$

$$M(t^*) = \bigcup_n M(t^n) = \bigcup_n \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(t^n)\}$$
$$= \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in \bigcup_n G(t^n)\}$$
$$= \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(t^*)\}.$$

We now show (19.4). Suppose the left-hand side holds. By (19.5),

$$M(r) \stackrel{\text{def}}{=} \{(x, x \diamond y) \mid x \diamond y \in G(\top) - G(\top r \top),\ y \in G(r\} = \varnothing.$$

By the left-hand side of (19.4), $M(p) \subseteq M(q)$. In particular, for any $x \in G(p) - G(\top r \top)$, $(\mathsf{first}\, x, x) \in M(p)$, therefore $(\mathsf{first}\, x, x) \in M(q)$ as well, thus $x \in G(q) - G(\top r \top)$. But this says $G(p) - G(\top r \top) = G(q) - G(\top r \top)$, thus $G(p) \subseteq G(q) \cup G(\top r \top) = G(q + \top r \top)$. It follows from (19.3) that the right-hand side of (19.4) holds. □

Corollary 19.2 KAT *is deductively complete for formulas of the form* (19.2) *over relational models.*

Proof. If the formula (19.2) is valid over relational models, then by Theorem 19.1, (iv) holds. Since KAT is complete for valid equations,

$$\mathsf{KAT} \vdash p + \top r \top = q + \top r \top.$$

But clearly

$$\mathsf{KAT} \vdash p + \top r \top = q + \top r \top \wedge r = 0 \Rightarrow p = q,$$

therefore

$$\mathsf{KAT} \vdash r = 0 \Rightarrow p = q.$$

□

19 From Elim2

19.1 Ideals

The elimination of hypotheses of the form $q = 0$ rests on the concept of an *ideal*. An *ideal* in an idempotent semiring K is a nonempty subset $I \subseteq K$ such that

(i) if $x, y \in I$, then $x + y \in I$

(ii) if $x \in I$ and $r \in K$, then $xr \in I$ and $rx \in I$

(iii) if $x \leq y$ and $y \in I$, then $x \in I$.

It follows that $0 \in I$. If desired, we might also postulate $1 \notin I$ to rule out the degenerate case $I = K$. This definition is in slight contrast to ideals in rings, where there is no analogue of (iii).

For $A \subseteq K$, define

$$\langle A \rangle \stackrel{\text{def}}{=} \{y \mid \exists n\, \exists a_1, \ldots, a_n \in A\, \exists u, v \in K\, y \leq u(a_1 + \cdots + a_n)v\}. \quad (19.6)$$

Lemma 19.1 *$\langle A \rangle$ is an ideal containing A, and is the smallest such ideal.*

Proof. Surely $A \subseteq \langle A \rangle$, since for $a \in A$, we can take $n = 1$, $y = a_1 = a$, and $u = v = 1$ in (19.6).

To show $\langle A \rangle$ is an ideal, we must show that it is closed under the operations (i)–(iii). For (i), if

$$y_1 \leq u_1(a_1 + \cdots + a_n)v_1 \qquad y_2 \leq u_2(b_1 + \cdots + b_m)v_2$$

with $a_1, \ldots, a_n, b_1, \ldots, b_m \in A$, then

$$\begin{aligned} y_1 + y_2 &\leq u_1(a_1 + \cdots + a_n)v_1 + u_2(b_1 + \cdots + b_m)v_2 \\ &\leq (u_1 + u_2)(a_1 + \cdots + a_n)(v_1 + v_2) + (u_1 + u_2)(b_1 + \cdots + b_m)(v_1 + v_2) \\ &\leq (u_1 + u_2)(a_1 + \cdots + a_n + b_1 + \cdots + b_m)(v_1 + v_2). \end{aligned}$$

For (ii), if $y \leq u(a_1 + \cdots + a_n)v$ and $r \in K$, then

$$ry \leq ru(a_1 + \cdots + a_n)v \qquad yr \leq u(a_1 + \cdots + a_n)vr.$$

Finally, for (iii), if $x \leq y \leq u(a_1 + \cdots + a_n)v$, then $x \leq u(a_1 + \cdots + a_n)v$.

To show that $\langle A \rangle$ is the smallest ideal containing A, we only need to show that all ideals that contain A also contain $\langle A \rangle$. If I is an ideal containing A, then since I is closed under (i), it contains all elements $a_1 + \cdots + a_n$ for $a_1, \ldots, a_n \in A$. Since it is closed under (ii), it must contain all $u(a_1 + \cdots + a_n)v$ for $a_1, \ldots, a_n \in A$ and $u, v \in K$. Finally, since it is closed under (iii), it must contain all y such that $y \leq u(a_1 + \cdots + a_n)v$, $a_1, \ldots, a_n \in A$, and $u, v \in K$. But this is all of $\langle A \rangle$. □

Given an ideal I, define

$$x \lesssim y \stackrel{\text{def}}{\iff} \exists z \in I\, x \leq y + z \qquad x \approx y \stackrel{\text{def}}{\iff} x \lesssim y \wedge y \lesssim x. \quad (19.7)$$

Alternatively and with the same effect, we could define

$$x \approx y \stackrel{\text{def}}{\iff} \exists z \in I\, x + z = y + z \qquad x \lesssim y \stackrel{\text{def}}{\iff} x + y \approx y. \quad (19.8)$$

Lemma 19.2 *Let $h : K_1 \to K_2$ be any semiring homomorphism between idempotent semirings. Then the kernel of h,*

$$\ker h \stackrel{\text{def}}{=} \{s \mid h(s) = 0\},$$

is an ideal. Conversely, any ideal is the kernel of a semiring epimorphism.

Proof. Let $h : K_1 \to K_2$ be a semiring homomorphism. We argue that $\ker h$ satisfies the properties (i)–(iii) in the definition of ideals. For (i), if $h(x) = h(y) = 0$, then $h(x+y) = h(x) + h(y) = 0 + 0 = 0$, since h is a homomorphism. Similarly, for (ii), if $h(x) = 0$ and $r \in K_1$ is any other element, Then $h(xr) = h(x)h(r) = 0 \cdot h(r) = 0$ and $h(rx) = h(r)h(x) = h(r) \cdot 0 = 0$. Finally, for (iii), if $x \leq y$ and $h(y) = 0$, then $x + y = y$, so $h(x) = h(x) + 0 = h(x) + h(y) = h(x + y) = h(y) = 0$.

For the other direction, consider the relations $\approx$ and $\lesssim$ defined in (19.7) and (19.8). One can show easily that the order $\lesssim$ is a preorder (reflexive and transitive) and $\approx$ is an equivalence relation. Denote by K/I the quotient of K modulo $\approx$ and let $[\cdot] : K \to K/I$ be the canonical map. The relation $\lesssim$ is well-defined on K/I:

$$x \approx y \lesssim z \Rightarrow x \lesssim y \lesssim z \Rightarrow x \lesssim z,$$

and is a partial order (reflexive, transitive, and antisymmetric), and $I = [0]$:

$$x \approx 0 \Leftrightarrow x \lesssim 0 \Leftrightarrow \exists z \in I\ x \leq z \Leftrightarrow x \in I.$$

Now we wish to show that $\approx$ is a congruence with respect to addition and multiplication; that is,

$$x \approx y \Rightarrow x + z \approx y + z \qquad\qquad y \approx y' \Rightarrow xyz \approx xy'z.$$

These are quite easy to verify:

$$\begin{aligned} x \lesssim y &\Rightarrow \exists w \in I\ x \leq y + w \\ &\Rightarrow \exists w \in I\ x \leq y + w \\ &\Rightarrow \exists w \in I\ x + z \leq y + z + w \\ &\Rightarrow x + z \lesssim y + z, \end{aligned}$$

and similarly $y \lesssim x \Rightarrow y + z \lesssim x + z$, therefore

$$x \approx y \Rightarrow x \lesssim y \text{ and } y \lesssim x \Rightarrow x + z \lesssim y + z \text{ and } y + z \lesssim x + z \Rightarrow x + z \approx y + z.$$

Thus the operations are well defined on $\approx$-classes and K/I is an idempotent semiring, and the canonical map $x \mapsto [x]$ is a semiring epimorphism $[\cdot] : K \to K/I$ with kernel I. □

Quite fortuitously, and more than a little surprisingly, if K is a KAT, then the congruence $\approx$ defined in (19.7) turns out to be a KAT-congruence, and the quotient K/I is a KAT.

Lemma 19.3 *If K is a* KAT*, then the relation $\approx$ is a* KAT*-congruence and K/I is a* KAT*. If $I = \langle A \rangle$, then K/I, $[\cdot]$ is initial among all homomorphic images of K in which the image of A vanishes; that is, given any homomorphism $h : K \to K'$ such that $h(a) = 0$ for all $a \in A$, there exists a homomorphism $h' : K/I \to K'$ such that $h = h' \circ [\cdot]$.*

Proof. We must first show that $\approx$ is a congruence with respect to * and $^-$ in order to verify that those operators are well defined on K/I. Note that, since I is closed under addition, to verify $x \approx y$ it suffices to find $z, w \in I$ such that $x + z = y + w$, as this implies that $x + z + w = y + z + w$ and $z + w \in I$.

For *, we wish to show that if $x \approx y$ then $x^* \approx y^*$. Reasoning in KAT, we have

$$(x+z)^* \;=\; x^*(zx^*)^* \;=\; x^* + x^*zx^*(zx^*)^*, \tag{19.9}$$

and similarly for $(y+z)^*$. If $x + z = y + z$ with $z \in I$, then $(x+z)^* = (y+z)^*$, thus by (19.9),

$$x^* + x^*zx^*(zx^*)^* \;=\; y^* + y^*zy^*(zy^*)^*,$$

and both $x^*zx^*(zx^*)^*$ and $y^*zy^*(zy^*)^*$ are in I, therefore $x^* \approx y^*$.

For negation, we show that the ideal I behaves like a Boolean algebra ideal on Boolean elements; that is, $c \approx d$ iff $c\bar{d} + \bar{c}d \in I$. Suppose $c + z = d + z$ with $z \in I$. Multiplying on the left by $\bar{c}$, we have $\bar{c}z = \bar{c}d + \bar{c}z$, so $\bar{c}d \in I$. Similarly, multiplying on the right by $\bar{d}$ gives $c\bar{d} \in I$, therefore $c\bar{d} + \bar{c}d \in I$. Conversely, if $c\bar{d} + \bar{c}d \in I$, then $c + c\bar{d} + \bar{c}d = c + d = d + c\bar{d} + \bar{c}d$, so $c \approx d$. By the symmetry of $c\bar{d} + \bar{c}d$, $c \approx d$ implies that $\bar{c} \approx \bar{d}$, thus $\approx$ is a congruence with respect to negation.

We have shown that $\approx$ is a congruence with respect to all the KAT operations, thus the KAT operations are well defined on the quotient K/I and the canonical map $x \mapsto [x]$ is a homomorphism. However, we have yet to show that K/I is a KAT. We know that it satisfies all equations, because the epimorphism $[\cdot]$ preserves all equations, and K is a KAT. However, we must also verify that the equational implications

$$ax \le x \Rightarrow a^*x \le x \qquad\qquad xa \le x \Rightarrow xa^* \le x$$

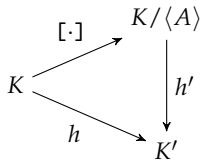
hold modulo $\approx$ as well. We show the former; the latter follows from symmetry.

To show that $ax \leq x \Rightarrow a^*x \leq x$ holds modulo $\approx$, we must show that if $ax \lesssim x$, then $a^*x \lesssim x$. If $ax \lesssim x$, then $ax \leq x + z$ for some $z \in I$. Reasoning in KAT,

$$a(x + a^*z) \;=\; ax + aa^*z \;\leq\; x + z + aa^*z \;=\; x + a^*z.$$

Applying the same rule in K, we have $a^*(x + a^*z) \leq x + a^*z$, thus $a^*x \leq x + a^*z$. Since $a^*z \in I$, $a^*x \lesssim x$.

Finally, to argue the last statement of the lemma, we wish to show that any homomorphism $h : K \to K'$ under which A vanishes factors through $K/\langle A\rangle$ via the canonical homomorphism $[\cdot]$. In other words, if $h(a) = 0$ for all $a \in A$, then there exists a homomorphism $h' : K/\langle A\rangle \to K'$ such that $h = h' \circ [\cdot]$.



We need only observe that the condition that A vanishes under h means that $\langle A\rangle$ is contained in $\ker h$, thus if $x \approx y$ then $h(x) = h(y)$, so h is well defined on $\approx$ classes and reduces to a map $h' : K/\langle A\rangle \to K'$. □

We have shown that if K is a KAT, then for any ideal I, the quotient K/I is a KAT, the canonical map $[\cdot] : K \to K/I$ is an epimorphism, and $I = [0]$. Moreover, K/I is initial among all homomorphic images of K such that $I = [0]$.

Another unusual fact is that, unlike the case of groups or rings, the ideal $[0]$ does not uniquely determine the homomorphic image up to isomorphism. For example, consider the free KA on one generator a. Modulo the inequality $a \leq 1$, the resulting algebra is isomorphic to the *tropical semiring* used in shortest path algorithms [] (Miscellaneous Exercise **??**); but the kernel of the canonical map to this algebra is $\{0\}$, the same as the identity.

If the KAT K has a top element $\top$, then Horn formulas

$$s_1 = 0 \wedge \cdots \wedge s_n = 0 \;\Rightarrow\; s = t \tag{19.10}$$

reduce to singles equations in K. For instance, if K is finitely generated with generators $p_1, \ldots, p_k$, as is the case with all finitely generated free algebras such as Reg P, B, we can take $\top = (p_1 + \cdots + p_k)^*$. To reduce the Horn formula (19.10) to an equation, let $z = \top(s_1 + \cdots + s_n)\top$. Then z is the maximum element of the ideal generated by $s_1, \ldots, s_n$. Thus (19.10) is equivalent to the equation $s + z = t + z$. This can also be viewed as selecting a canonical element $s + z$ from the congruence class of s, but in this case the map $s \mapsto s + z$ is not a homomorphism.

Lecture 20

Complexity of Kleene Algebra with Tests

In this lecture we show that KAT, like KA, is PSPACE-complete. Thus KAT, while considerably more expressive than KA, is no more difficult to decide. The results of this lecture are from [28, 75].

We have shown (Lecture **??**) that the Hoare theory of KAT (Horn formulas with premises of the form $r = 0$) reduces efficiently to the equational theory. We have also argued (Lecture **??**) that the equational theories of KAT and KAT* (star-continuous KAT) coincide, and that these theories are complete over certain language-theoretic and relational models.

Our PSPACE algorithm makes use of $\mathsf{Reg}\,\mathsf{P},\mathsf{B}$, the free language-theoretic model involving sets of guarded strings introduced in Lecture **??**, and matrices over Kleene algebras with tests.

We will show later that star-continuous KA in the presence of extra commutativity conditions of the form $pq = qp$, even for primitive p and q, is undecidable. This was observed by Cohen [24]. In fact, the universal Horn theory of KAT* is Π^1_1-complete [71].

Matrix Algebras

Let K, B be a Kleene algebra with tests. As argued in Lecture **??**, the structure

$$\mathrm{Mat}(n, K, B) = (\mathrm{Mat}(n, K), \Delta(n, B), +, \cdot, ^*, ^-, 0_n, I_n)$$

again forms a Kleene algebra with tests, where $\mathrm{Mat}(n, K)$ denotes the family of $n \times n$ matrices over K, $\Delta(n, B)$ denotes the family of $n \times n$ diagonal matrices over B, the operations $+$ and $\cdot$ are the usual operations of matrix addition and multiplication, respectively, 0_n is the $n \times n$ zero matrix, and I_n the $n \times n$ identity matrix. The operation * on matrices is defined inductively:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^* = \begin{bmatrix} (A + BD^*C)^* & (A + BD^*C)^*BD^* \\ (D + CA^*B)^*CA^* & (D + CA^*B)^* \end{bmatrix} \tag{20.1}$$

The operation $^-$ on $\Delta(n, B)$ just complements the diagonal elements, leaving the off-diagonal elements 0.

KAT Homomorphisms and Finitary Algebras

Definition 20.1 *Let K, B and K', B' be Kleene algebras with tests. A* KAT-homomorphism *$h : K, B \to K', B'$ is a Kleene algebra homomorphism $h : K \to K'$ whose restriction to B is a Boolean algebra homomorphism $h : B \to B'$.*

Lemma 20.2 *Let $h : K, B \to K', B'$ be a* KAT*-homomorphism, and let $H : \mathrm{Mat}(n, K) \to \mathrm{Mat}(n, K')$ be its componentwise extension to matrices. Then H is a* KAT*-homomorphism $\mathrm{Mat}(n, K, B) \to \mathrm{Mat}(n, K', B')$.*

Proof. By definition, $H(E)_{ij} = h(E_{ij})$. It is immediate that $H(E + F) = H(E) + H(F)$, $H(EF) = H(E)H(F)$, $H(0) = 0$, and $H(I) = I$. For *, we can use the inductive definition (20.1) to give a straightforward inductive proof that $H(E^*) = H(E)^*$. Finally, for $E \in \Delta(n, B)$,

$$H(\overline{E})_{ii} = h(\overline{E}_{ii}) = \overline{h(E_{ii})} = \overline{H(E)}_{ii}, \qquad H(\overline{E})_{ij} = 0 = \overline{H(E)}_{ij},\ i \neq j,$$

so $H(\overline{E}) = \overline{H(E)}$. □

A KA or KAT is called *finitary* if for all $a \in K$ there exists an $m \geq 0$ such that $a^* = (1 + a)^m$. Any finite algebra is finitary, and any finitary algebra is star-continuous.

Lemma 20.3 *If K is finitary, then so is $\mathrm{Mat}(n, K)$.*

Proof. We proceed by induction on n. For the basis, the algebras K and $\mathrm{Mat}(1, K)$ are isomorphic, so there is nothing to prove. Now suppose $n \geq 2$. Break up $E \in \mathrm{Mat}(n, K)$ arbitrarily into submatrices

$$E = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

where A and D are square. Using the denesting rule,

$$\begin{aligned}
\begin{bmatrix} A & B \\ C & D \end{bmatrix}^* &= \left(\begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix} + \begin{bmatrix} 0 & B \\ C & 0 \end{bmatrix} \right)^* \\
&= \left(\begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix}^* \begin{bmatrix} 0 & B \\ C & 0 \end{bmatrix} \right)^* \begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix}^* \\
&= \left(\begin{bmatrix} A^* & 0 \\ 0 & D^* \end{bmatrix} \begin{bmatrix} 0 & B \\ C & 0 \end{bmatrix} \right)^* \begin{bmatrix} A^* & 0 \\ 0 & D^* \end{bmatrix} \\
&= \begin{bmatrix} 0 & A^*B \\ D^*C & 0 \end{bmatrix}^* \begin{bmatrix} A^* & 0 \\ 0 & D^* \end{bmatrix}
\end{aligned}$$

By the induction hypothesis, there exists an $m \geq 0$ such that

$$\begin{bmatrix} A^* & 0 \\ 0 & D^* \end{bmatrix} = \begin{bmatrix} (I+A)^m & 0 \\ 0 & (I+D)^m \end{bmatrix} = \begin{bmatrix} I+A & 0 \\ 0 & I+D \end{bmatrix}^m \leq (I+E)^m$$

Also, using the KA theorem $x^* = (xx)^*(1+x)$,

$$\begin{bmatrix} 0 & A^*B \\ D^*C & 0 \end{bmatrix}^* = \begin{bmatrix} A^*BD^*C & 0 \\ 0 & D^*CA^*B \end{bmatrix}^* \left(\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} 0 & A^*B \\ D^*C & 0 \end{bmatrix} \right)$$

By the induction hypothesis, there exists a $k \geq 0$ such that

$$\begin{aligned}
\begin{bmatrix} A^*BD^*C & 0 \\ 0 & D^*CA^*B \end{bmatrix}^* &= \begin{bmatrix} (A^*BD^*C)^* & 0 \\ 0 & (D^*CA^*B)^* \end{bmatrix} \\
&= \begin{bmatrix} (I+A^*BD^*C)^k & 0 \\ 0 & (I+D^*CA^*B)^k \end{bmatrix} \\
&= \begin{bmatrix} I+A^*BD^*C & 0 \\ 0 & I+D^*CA^*B \end{bmatrix}^k \\
&= \left(\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} A^* & 0 \\ 0 & D^* \end{bmatrix} \begin{bmatrix} 0 & B \\ C & 0 \end{bmatrix} \begin{bmatrix} A^* & 0 \\ 0 & D^* \end{bmatrix} \begin{bmatrix} 0 & B \\ C & 0 \end{bmatrix} \right)^k \\
&\leq (I + (I+E)^m E (I+E)^m E)^k \\
&\leq (I+E)^{2k(m+1)}
\end{aligned}$$

Similarly,

$$\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} 0 & A^*B \\ D^*C & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} A^* & 0 \\ 0 & D^* \end{bmatrix} \begin{bmatrix} 0 & B \\ C & 0 \end{bmatrix} \leq I + (I+E)^m E \leq (I+E)^{m+1}$$

Putting these all together, we have

$$E^* \leq (I+E)^{2km+2k+2m+1}.$$

□

Matrices over a Boolean Algebra

In this section we establish some special properties of matrices over a Boolean algebra that will prove useful in the subsequent development.

If B is the distinguished Boolean algebra of a Kleene algebra with tests K, then the algebra $\mathrm{Mat}(n, B)$ is a subalgebra of $\mathrm{Mat}(n, K)$. (Note that it is not the distinguished Boolean algebra of $\mathrm{Mat}(n, K)$; in fact, it is not even a Boolean algebra in general). The algebra $\Delta(n, B)$ of diagonal matrices over B is the distinguished Boolean algebra of both $\mathrm{Mat}(n, K)$ and $\mathrm{Mat}(n, B)$.

Since $b^* = 1$ for any $b \in B$, it follows immediately from Lemma 20.3 that $\mathrm{Mat}(n, B)$ is finitary. In fact, it can be established by combinatorial means that if $A \in \mathrm{Mat}(n, B)$, then $A^* = (I + A)^{n-1}$, but we will not need this tighter bound.

Let B denote the free Boolean algebra on generators B. Given a matrix $J \in \mathrm{Mat}(n, B)$ and an atom α, let J_α be the 0,1-matrix

$$(J_\alpha)_{ij} = \begin{cases} 1, & \text{if } \alpha \leq J_{ij}, \\ 0, & \text{otherwise.} \end{cases}$$

Lemma 20.4

$$\alpha \leq (J^*)_{ij} \Leftrightarrow (J_\alpha^*)_{ij} = 1.$$

In particular, one can determine whether $\alpha \leq (J^*)_{ij}$ in linear time.

Proof. The first statement is a direct application of Lemma 20.2, using the Boolean homomorphism $h_\alpha : B \to \{0, 1\}$ defined by

$$h_\alpha(b) = \begin{cases} 1, & \text{if } \alpha \leq b, \\ 0, & \text{otherwise.} \end{cases}$$

Then $J_\alpha = H_\alpha(J)$, where H_α is the componentwise extension of h_α to matrices. The condition to be proved is equivalent to the statement $H_\alpha(J^*) = H_\alpha(J)^*$.

The entries of J_α can be determined by testing whether $\alpha \leq b$, which essentially amounts to evaluating a Boolean expression on a given truth assignment. The matrix J_α is a 0,1 matrix, so $(J_\alpha^*)_{ij}$ can be determined in linear time by depth first search on the corresponding directed graph. The entire matrix J_α^* can be computed efficiently using any standard transitive closure algorithm. □

Matrix Representation of Terms

We eventually want to give an algorithm for deciding whether $\mathsf{KAT} \models p = q$. By Theorem 17.4 of Lecture **??**, it suffices to decide whether $G(p) =$

$G(q)$, where G is the canonical interpretation $G : \mathsf{Exp}\,\mathsf{P},\mathsf{B} \to \mathsf{Reg}\,\mathsf{P},\mathsf{B}$, as defined in Lecture **??**.

One possible approach, exploited in Lecture **??**, is to construct from $p \in \mathsf{Exp}\,\mathsf{P},\mathsf{B}$ a regular expression $\widehat{p} \in \mathsf{Exp}\,\mathsf{P},\mathsf{B}$ such that

$$G(p) = R(\widehat{p}). \tag{20.2}$$

Then deciding whether $G(p) = G(q)$ reduces to deciding whether $R(\widehat{p}) = R(\widehat{q})$, which we know how to do in PSPACE.

Unfortunately, the construction of $\widehat{p}$ from p as given in Lecture **??** involves an exponential blowup, which the following example shows to be unavoidable. Suppose $k = 2m$. Consider the expression

$$p = (b_1 b_{m+1} + \overline{b}_1 \overline{b}_{m+1})(b_2 b_{m+2} + \overline{b}_2 \overline{b}_{m+2}) \cdots (b_m b_k + \overline{b}_m \overline{b}_k)$$

This expression represents the set of atoms in which the i^{th} and $m + i^{\text{th}}$ literal have the same parity. Any nondeterministic finite automaton accepting $G(p)$ must store in its state the first half of the string so that it can verify that the second half is correct. Therefore the automaton must have at least 2^m states. Since the translation between regular expressions and nondeterministic automata is linear, any regular expression $\widehat{p}$ such that $R(\widehat{p}) = G(p)$ must be exponentially longer than p.

To circumvent this exponential blowup, we work with a matrix representation of expressions. The construction of Kleene's theorem as given in Lecture **??** produces a matrix $P \in \mathrm{Mat}(n, \mathcal{F}_{\mathsf{P},\mathsf{B}})$ with small entries and 0,1-vectors u, v of length n such that

$$R(p) = R(u^T P^* v), \tag{20.3}$$

where n is approximately the size of p and $\mathcal{F}_{\mathsf{P},\mathsf{B}}$ is the free Kleene algebra with tests on generators P and B. The construction of P is by induction on the structure of p, and corresponds to the combinatorial construction of an automaton from a regular expression as found for example in [50, 67]. The matrix P is the transition matrix of the automaton equivalent to the regular expression p over the input alphabet $\mathsf{P} \cup \mathsf{B} \cup \overline{\mathsf{B}}$. The vectors u and v specify the start and final states of the automaton, respectively. The elements of P are 0, 1, and sums of primitive symbols. This construction is given in its entirety in Lecture **??**, so we do not repeat it here.

Since the entries of P are sums of primitive symbols, we can write $P = J + A$, where the entries of J are sums of elements of $\mathsf{B} \cup \overline{\mathsf{B}}$ and the entries of A are sums of elements of P. Using the denesting rule of KA, we can then write

$$P^* = (J^* A)^* J^*$$

This form is particularly well suited to the treatment of guarded strings $\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m$, the guards α_i being handled by J^* and the symbols p_i by A.

We extend the definition of J_α above to general matrices. For $p \in \mathsf{P}$, define the 0-1 matrix

$$(A_p)_{ij} = \begin{cases} 1, & \text{if } p \leq A_{ij}, \\ 0, & \text{otherwise.} \end{cases}$$

Lemma 20.5

$$\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m \in G(u^T (J^* A)^* J^* v) \;\Leftrightarrow\; u^T J^*_{\alpha_0} A_{p_1} J^*_{\alpha_1} \cdots J^*_{\alpha_{m-1}} A_{p_m} J^*_{\alpha_m} v = 1.$$

Proof. Because of the restricted form of the entries in J^* and A, all guarded strings in $G(u^T (J^* A)^k J^* v)$ are of the form $\alpha_0 p_1 \alpha_1 \cdots \alpha_{k-1} p_k \alpha_k$. Since

$$G(u^T (J^* A)^* J^* v) = \bigcup_{k \geq 0} G(u^T (J^* A)^k J^* v)\,,$$

we have that

$$\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m \in G(u^T (J^* A)^* J^* v) \;\Leftrightarrow\; \alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m \in G(u^T (J^* A)^m J^* v).$$

Furthermore, by the definition of matrix multiplication, this occurs iff there exist $s_0, t_0, s_1, t_1, \ldots, s_m, t_m$ such that

- $u_{s_0} = 1$
- $\alpha_i \leq (J^*)_{s_i t_i}, 0 \leq i \leq m$
- $p_i \leq A_{t_{i-1} s_i}, 1 \leq i \leq m$
- $v_{t_m} = 1$.

By Lemma 20.4 and the definition of A_{p_i}, this occurs iff there exist s_0, t_0, $s_1, t_1, \ldots, s_m, t_m$ such that

- $u_{s_0} = 1$
- $(J^*_{\alpha_i})_{s_i t_i} = 1, 0 \leq i \leq m$
- $(A_{p_i})_{t_{i-1} s_i} = 1, 1 \leq i \leq m$
- $v_{t_m} = 1$.

By the definition of Boolean matrix multiplication, this occurs iff

$$u^T J^*_{\alpha_0} A_{p_1} J^*_{\alpha_1} \cdots J^*_{\alpha_{m-1}} A_{p_m} J^*_{\alpha_m} v = 1.$$

□

A PSPACE Algorithm

Now we give a PSPACE algorithm for deciding whether KAT $\models p \leq q$, or equivalently by Theorem 17.4 of Lecture **??**, whether $G(p) \subseteq G(q)$. The algorithm will nondeterministically guess a guarded string

$$\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m \in G(p) - G(q).$$

We first produce the matrices u, P, v and y, Q, z such that

$$R(p) = R(u^T P^* v) \qquad R(q) = R(y^T Q^* z).$$

By the fact that $G(p) = \bigcup_{x \in R(p)} G(x)$ proved in Lecture **??**, we also have

$$G(p) = G(u^T P^* v) \qquad G(q) = G(y^T Q^* z).$$

Writing $P = J + A$ and $Q = K + B$ where the entries of J and K are sums of elements of $\mathsf{B} \cup \overline{\mathsf{B}}$ and the entries of A and B are sums of elements of P, we have

$$G(p) = G(u^T (J^* A) J^* v) \qquad G(q) = G(y^T (K^* B) K^* z).$$

By Lemma 20.5, it suffices to guess $\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m$ such that

$$u^T J^*_{\alpha_0} A_{p_1} J^*_{\alpha_1} \cdots J^*_{\alpha_{m-1}} A_{p_m} J^*_{\alpha_m} v = 1 \quad y^T K^*_{\alpha_0} B_{p_1} K^*_{\alpha_1} \cdots K^*_{\alpha_{m-1}} B_{p_m} K^*_{\alpha_m} z = 0.$$

We first guess α_0 and calculate J_{α_0} and K_{α_0} and their reflexive transitive closures $J^*_{\alpha_0}$ and $K^*_{\alpha_0}$, then calculate the 0-1 row vectors

$$u_0^T = u^T J^*_{\alpha_0} \qquad y_0^T = y^T K^*_{\alpha_0}.$$

At stage i, say we have calculated u_i and y_i such that

$$u_i^T = u^T J^*_{\alpha_0} A_{p_1} J^*_{\alpha_1} \cdots J^*_{\alpha_{i-1}} A_{p_i} J^*_{\alpha_i} \quad y_i^T = y^T K^*_{\alpha_0} B_{p_1} K^*_{\alpha_1} \cdots K^*_{\alpha_{i-1}} B_{p_i} K^*_{\alpha_i}.$$

We guess p_{i+1} and calculate $A_{p_{i+1}}$ and $B_{p_{i+1}}$, then guess α_{i+1} and calculate $J_{\alpha_{i+1}}$ and $K_{\alpha_{i+1}}$ and their reflexive transitive closures $J^*_{\alpha_{i+1}}$ and $K^*_{\alpha_{i+1}}$. We then calculate

$$u_{i+1}^T = u_i^T A_{p_{i+1}} J_{\alpha_{i+1}} \qquad y_{i+1}^T = y_i^T B_{p_{i+1}} K^*_{\alpha_{i+1}}.$$

It follows inductively that for all $m \geq 0$,

$$u_m^T = u^T J^*_{\alpha_0} A_{p_1} J^*_{\alpha_1} \cdots J^*_{\alpha_{m-1}} A_{p_m} J^*_{\alpha_m} \quad y_m^T = y^T K^*_{\alpha_0} B_{p_1} K^*_{\alpha_1} \cdots K^*_{\alpha_{m-1}} B_{p_m} K^*_{\alpha_m}.$$

We halt and accept if at any point $u_m^T v = 1$ and $y_m^T z = 0$.

The correctness of this algorithm follows from Lemma 20.5. It uses at most polynomial space, since in each stage of the computation only the vectors u_i and y_i need be remembered.

The algorithm can be made deterministic using Savitch's Theorem (see [50]). The problem is PSPACE-hard, as shown in Lecture **??**. It is not necessary to worry about termination or the length of the computation; Savitch's theorem takes care of that. We have thus shown

Theorem 20.6 *The equational theory of* KAT *is PSPACE-complete.*

In the next lecture, we will show that PHL is also PSPACE-hard.

Lecture 21

Complexity of PHL

In this lecture we show the PSPACE-hardness of propositional Hoare logic (PHL). This result was first proved in [70] by a direct encoding of arbitrary polynomial-space Turing machines. The proof given here is a simpler proof from [27] encoding the universality problem for nondeterministic finite automata, a well-known PSPACE-complete problem [38] (see Lecture **??**).

Recall from Lecture **??** that the deduction rules of PHL consist of the usual composition, conditional, while, and weakening rules of Hoare logic, as well as the and- and or-rule

$$\frac{\{c\}\,p\,\{d\},\ c \in C}{\{\bigvee C\}\,p\,\{d\}} \qquad\qquad \frac{\{b\}\,p\,\{c\},\ c \in C}{\{b\}\,p\,\{\bigwedge C\}}$$

for any finite set C of propositions. The and- and or-rule are not part of the traditional formulation [5] but are necessary for completeness [76]. The assignment axiom is meaningless in PHL and is omitted. We are interested in the validity of rules of the form

$$\frac{\{b_1\}\,p_1\,\{c_1\} \qquad \ldots \qquad \{b_n\}\,p_n\,\{c_n\}}{\{b\}\,p\,\{c\}} \tag{21.1}$$

interpreted as universal Horn sentences over relational models.

We consider two related decision problems. Given a rule of the form (21.1):

(i) Is it relationally valid? That is, is it true in all relational models?

(ii) Is it derivable in PHL?

We show that both of these problems are PSPACE-hard by a single reduction from the universality problem for nondeterministic finite automata: Given such an automaton M over input alphabet $\{0,1\}$ with states S, nondeterministic transition function $\Delta : S \times \{0,1\} \to 2^S$, start states $I \subseteq S$, and final states $F \subseteq S$, does M accept all strings?

Intuitively, our construction simulates the *subset construction* for forming a deterministic automaton from M (see for example [50, 67]). Typically, one thinks of pebbles placed on the states of M and moved according to the transition rules of M. The pebbles keep track of all the states that M could possibly be in after scanning a prefix of the input string. We start at time 0 with pebbles on all the start states of M. Now say we have a set A of states of M occupied by pebbles at time t. If the next input symbol is a, at time $t+1$ we place a pebble on each state reachable under input symbol a from a state in A.

To encode this in PHL, let a_u be a primitive test for each state $u \in S$. Let b be another primitive test and p a primitive action. Let

$$\text{START} \stackrel{\text{def}}{=} \bigwedge_{u \in I} a_u \qquad\qquad \text{FINAL} \stackrel{\text{def}}{=} \bigvee_{u \in F} a_u.$$

Consider the rule

$$\frac{\{a_u \wedge b\}\, p\, \{a_v\},\ \text{all } v \in \Delta(u,1) \qquad \{a_u \wedge \neg b\}\, p\, \{a_v\},\ \text{all } v \in \Delta(u,0)}{\{\text{START}\}\ \texttt{while}\ \text{FINAL}\ \texttt{do}\ p\, \{0\}} \tag{21.2}$$

Note that this rule is linear in the size of the description of the automaton.

Intuitively, a_u says that state u is occupied by a pebble, and b (respectively, $\neg b$) says that the next input symbol is 1 (respectively, 0). The program p says to place pebbles on *at least* all states reachable from a currently pebbled state under the next input symbol according to the transition rules of M. The formula START says that all start states are pebbled, and FINAL says that at least one final state is pebbled. Other subexpressions of (21.2) have the following intuitive meanings:

$\{a_u \wedge b\}\, p\, \{a_v\}$	"If state u is pebbled at time t, and if the next input symbol is 1, then there must be a pebble on state v at time $t+1$."
`while` FINAL `do` p	"Continue updating the pebble positions as long as there is a final state occupied by a pebble."

The formula (21.2) says intuitively that if all start states are initially pebbled, and if in each step the pebbles are moved such that *at least* those states that are reachable under the current input symbol from a currently pebbled state are pebbled in the next step, then there is always a pebble on a final state.

Now we proceed to the formal proof of the correctness of this construction.

Theorem 21.1 *The following are equivalent:*

(i) *The rule* (21.2) *is relationally valid.*

(ii) *The rule* (21.2) *is derivable in* PHL.

(iii) *The automaton M accepts all strings.*

Proof. We show (ii) $\Rightarrow$ (i) $\Rightarrow$ (iii) $\Rightarrow$ (ii). The first implication (ii) $\Rightarrow$ (i) is immediate from the soundness of PHL over relational models.

For the second implication (i) $\Rightarrow$ (iii), let $x \in \{0,1\}^*$ be any input string. Build a relational model of PHL as follows: the elements are the prefixes of x; the formula b is true at y if $x = yz$ and the first symbol of z is 1; the formula a_u is true at y if $u \in \Delta(I, y)$, that is, if the state u is reachable under input string y from a start state of M; and the program p is the relation consisting of all pairs (y, z) for $|z| = |y| + 1$. An easy argument shows that all premises of (21.2) hold in this model. By (i), the conclusion holds, thus x satisfies FINAL, so there is a final state reachable from a start state of M under input string x.

Finally, for the last implication (iii) $\Rightarrow$ (ii), we prove (21.2) in PHL. Let

$$R \stackrel{\text{def}}{=} \{\Delta(I, x) \mid x \in \{0,1\}^*\} \qquad\qquad \varphi \stackrel{\text{def}}{=} \bigvee_{A \in R} \bigwedge_{s \in A} a_s.$$

The set R is just the set of reachable states of the subset automaton. It follows in a straightforward way from the premises of (21.2) using the and-, or-, and weakening rules that φ is an invariant of p, or in other words $\{\varphi\}\, p\, \{\varphi\}$. Since START $\Rightarrow$ φ and $\varphi \Rightarrow$ FINAL, the latter being a consequence of (iii), the conclusion of (21.2) follows from the while rule and weakening. □

Lecture 22

Complexity of Reasoning under Assumptions

In this lecture and the next we study the complexity of reasoning in Kleene algebra and star-continuous Kleene algebra in the presence of extra equational assumptions E; that is, the complexity of deciding the validity of universal Horn formulas $E \Rightarrow s = t$, where E is a finite set of equations. We obtain various levels of complexity based on the form of the assumptions E. Our main results are: for star-continuous Kleene algebra, (i) if E contains only commutativity assumptions $pq = qp$, the problem is Π_1^0-complete; (ii) if E contains only monoid equations, the problem is Π_2^0-complete; and (iii) for arbitrary equations E, the problem is Π_1^1-complete. The last problem is the universal Horn theory of the star-continuous Kleene algebras. These results are from [69].

Reasoning with Assumptions

The equational theory of KA alone is PSPACE-complete, and this is as efficient as one could expect. As we have argued, however, in practice one often needs to reason in the presence of assumptions of various forms. For example, a *commutativity condition* $pq = qp$ models the fact that the programs p and q can be executed in either order with the same result. In the presence of tests, the commutativity condition $pb = bp$ models the fact that the execution of the program p does not affect the value of the test b.

Such assumptions are needed to reason about basic program transformations such as constant propagation and moving static computations out of loops.

As shown in Lecture **??**, assumptions of the form $pb = bp$ where b is a test do not increase the complexity of KAT. Unfortunately, slightly more general commutativity assumptions $pq = qp$, even for p and q atomic, may lead to undecidability. Cohen gave a direct proof of this fact encoding Post's Correspondence Problem. This result can also be shown to follow from a 1979 result of Berstel [10] with a little extra work; we will give this argument this below.

These considerations bring up the general question:

> *How hard is it to reason in Kleene algebra under equational assumptions?*

Equivalently and more formally,

> *What is the complexity of deciding the validity of universal Horn formulas of the form $E \Rightarrow s = t$, where E is a finite set of equations?*

Here "universal" refers to the fact that the atomic symbols of E, s, and t are implicitly universally quantified. This question is quite natural, since the axiomatization of KA is itself a universal Horn axiomatization.

The question becomes particularly interesting in the presence of star-continuity (KA*). Recall that a Kleene algebra is *star-continuous* if it satisfies the infinitary condition

$$pq^*r \quad = \quad \sup_{n \geq 0} pq^n r,$$

where the supremum is with respect to the natural order in the Kleene algebra. Not all Kleene algebras are star-continuous, but all known naturally occurring ones are. Moreover, although star-continuity often provides a convenient shortcut in equational proofs, there are no more equations provable with it than without it, as we have shown.

Because of these considerations, it has become common practice to adopt star-continuity as a matter of course. However, this is not without consequence: although the equational theories of KA and KA* coincide, their Horn theories do not.

The known results about the complexity of the Horn theories of KA and KA* are summarized in Table 1.

The results D and H apply to Kleene algebras with tests and were proved in Lecture **??**. The decision problems in the column labeled KA are all r.e. because of the finitary axiomatization of KA. The r.e.-hardness of A and B follows from the fact that these problems encode the word problem for finitely presented monoids, shown r.e.-hard independently

Table 1: Main Results

Form of assumptions	KA	KA*
unrestricted	A. Σ_1^0-complete	E. Π_1^1-complete
monoid equations	B. Σ_1^0-complete	F. Π_2^0-complete
$pq = qp$	C. *EXPSPACE*-hard	G. Π_1^0-complete
$pb = bp$	D. PSPACE-complete	H. PSPACE-complete

by Post and Markov in 1947 (see [33, Theorem 4.3, p. 98]). The EXPSPACE-hardness of C follows from the EXPSPACE-hardness of the word problem for commutative monoids [83]. It is not known whether C is decidable.

Perhaps the most remarkable of these results is E. This is the general question of the complexity of the universal Horn theory of the star-continuous Kleene algebras. This question is related to a conjecture of Conway [29, p. 103], who asked for an axiomatization of the universal Horn theory of the regular sets. The phrasing of Conway's conjecture is somewhat ambiguous, and a literal interpretation is relatively easy to refute [62].

That the universal Horn theory of KA* should be so highly complex is quite surprising in light of the relative simplicity of the axiomatization. There are a few examples of Π_1^1-completeness results in Propositional Dynamic Logic (PDL), but PDL is a relatively more sophisticated two-sorted system and takes significant advantage of a restricted semantics involving only relational models.

This lecture and the next assume a basic knowledge of complexity of abstract data types and recursion theoretic hierarchies. Good introductory references on these topics are [85] and [47], respectively.

Regular Sets over a Monoid

A cornerstone of our approach is the universality of Reg M, the regular subsets of an arbitrary monoid M, discussed earlier in Lecture **??**. Recall that any monoid homomorphism $h : M \to K$ from a monoid M to the multiplicative monoid of a star-continuous Kleene algebra K extends uniquely to a Kleene algebra homomorphism $\widehat{h} : \mathsf{Reg}\, M \to K$. In category-theoretic terms, the map $M \mapsto \mathsf{Reg}\, M$ constitutes a left adjoint to the forgetful functor taking a star-continuous Kleene algebra to its multiplicative monoid.

In practice, this property will allow us to restrict our attention to algebras of the form $\mathsf{Reg}\, \Sigma^*/E$ when dealing with universal Horn formulas

$E \Rightarrow s = t$, where E consists of monoid equations. Intuitively, we can think in terms of regular sets of equivalence classes of words modulo E.

More formally, let M be a monoid with identity 1_M. Recall that the powerset 2^M forms a natural star-continuous Kleene algebra under the operations

$$\begin{array}{rcl@{\qquad}rcl} A + B & \stackrel{\text{def}}{=} & A \cup B & 0 & \stackrel{\text{def}}{=} & \varnothing \\ AB & \stackrel{\text{def}}{=} & \{xy \mid x \in A,\ y \in B\} & 1 & \stackrel{\text{def}}{=} & \{1_M\} \\ A^* & \stackrel{\text{def}}{=} & \bigcup_{n \geq 0} A^n & & & \end{array}$$

The injection $\rho_M : x \mapsto \{x\}$ is a monoid homomorphism embedding M into the multiplicative monoid of 2^M.

Let $\mathsf{Reg}\, M$ denote the smallest Kleene subalgebra of 2^M containing the image of M under the map ρ_M. This is a star-continuous Kleene algebra and is called the *algebra of regular sets over* M.

The map $M \mapsto \mathsf{Reg}\, M$, along with the map that associates with every monoid homomorphism $h : M \to M'$ the Kleene algebra homomorphism $\mathsf{Reg}\, h : \mathsf{Reg}\, M \to \mathsf{Reg}\, M'$ defined by

$$\mathsf{Reg}\, h\ (A) \stackrel{\text{def}}{=} \{h(x) \mid x \in A\},$$

constitute a functor Reg from the category of monoids and monoid homomorphisms to the category KA^* of star-continuous Kleene algebras and Kleene algebra homomorphisms.

$$\begin{array}{ccc} M & \xrightarrow{\quad h \quad} & M' \\ {\scriptstyle \rho_M}\downarrow & & \downarrow{\scriptstyle \rho_{M'}} \\ \mathsf{Reg}\, M & \xrightarrow{\mathsf{Reg}\, h} & \mathsf{Reg}\, M' \end{array} \tag{22.1}$$

The functor Reg is the left adjoint of the forgetful functor that takes a star-continuous Kleene algebra to its multiplicative monoid. This implies that any monoid homomorphism $h : M \to K$ from a monoid M to the multiplicative monoid of a star-continuous Kleene algebra K extends uniquely through ρ_M to a Kleene algebra homomorphism $\widehat{h} : \mathsf{Reg}\, M \to K$:

$$\begin{array}{ccc} M & \xrightarrow{\quad h \quad} & K \\ {\scriptstyle \rho_M}\downarrow & \nearrow_{\widehat{h}} & \\ \mathsf{Reg}\, M & & \end{array} \tag{22.2}$$

The homomorphism $\widehat{h}$ is defined as follows:

$$\widehat{h}(A) \stackrel{\text{def}}{=} \sup \{h(x) \mid x \in A\}. \tag{22.3}$$

This makes sense for star-continuous Kleene algebras because of Theorem **??** of Lecture **??**, which says that *suprema of all definable subsets of a star-continuous Kleene algebra exist*. It does not work for Kleene algebras in general, since the supremum on the right-hand side of (22.3) may not exist.

Monoid Equations

Now we indicate how to take advantage of the universality property (22.2) to obtain the results F and G in Table 1.

Let Σ be a finite alphabet. Let E be a finite set of equations between words in Σ^*, the free monoid over Σ. Let s, t be regular expressions over Σ. Let Σ^*/E denote the quotient monoid. For $x \in \Sigma^*$, let $[x]$ denote the E-congruence class of x in Σ^*/E. The map $\iota : a \mapsto \{[a]\}$ constitutes an interpretation over the star-continuous Kleene algebra $\mathsf{Reg}\,\Sigma^*/E$, called the *standard interpretation*.

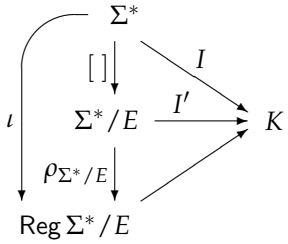
Lemma 22.1 *The following are equivalent:*

(i) $\mathsf{KA}^* \models E \Rightarrow s = t$*; that is, the Horn formula* $E \Rightarrow s = t$ *is true in all star-continuous Kleene algebras under all interpretations;*

(ii) $\mathsf{Reg}\,\Sigma^*/E, \iota \models s = t$.

Proof. It is easily verified that $\mathsf{Reg}\,\Sigma^*/E$ satisfies E under the standard interpretation ι. The implication (i) $\Rightarrow$ (ii) follows.

Conversely, for (ii) $\Rightarrow$ (i), let I be any interpretation into a star-continuous Kleene algebra K satisfying E. The monoid homomorphism $I : \Sigma^* \to K$ factors as $I = I' \circ [\,]$, where $I' : \Sigma^*/E \to K$. The universality property (22.2) then implies that I', hence I, factors through $\mathsf{Reg}\,\Sigma^*/E$.



Thus any equation true in $\mathsf{Reg}\,\Sigma^*/E$ under interpretation ι is also true in K under I. □

This result will allow us to restrict our attention to $\mathsf{Reg}\,\Sigma^*/E$ for the purpose of proving F and G in Table 1.

Encoding Turing Machines

The lower bound proofs for E, F, and G in Table 1 depend partially on encoding Turing machine computations as monoid equations. We follow the treatment of Davis [33].

Without loss of generality, we consider only deterministic Turing machines M that conform to the following restrictions.

- M has input alphabet $\{a\}$ and finite tape alphabet Γ containing a and a special blank symbol $\sqcup$ different from a. The alphabet Γ may contain other symbols as well.
- It has a finite set of states Q disjoint from Γ containing a start state s and one or more halt states distinct from s.
- There are no transitions into the start state s and no transitions out of any halt state. Thus, once M enters a halt state, it cannot proceed.
- It has a single two-way-infinite read-write tape, padded on the left and right by infinitely many blanks $\sqcup$.
- M never writes a blank symbol between two nonblank symbols.

Let $\vdash$, $\dashv$ be two special symbols that are not in Γ or Q. Let

$$\Delta \stackrel{\text{def}}{=} \Gamma \cup Q \cup \{\vdash, \dashv\}.$$

A *configuration* is a string in Δ^* of the form $\vdash xqy\dashv$, where $x, y \in \Gamma^*$ and $q \in Q$. Configurations describe instantaneous global descriptions of M in the course of some computation. In the configuration $\vdash xqy\dashv$, the current state is q, the tape currently contains xy surrounded by infinitely many blanks $\sqcup$ on either side, and the machine is scanning the first symbol of y. If y is null, then the machine is assumed to be scanning the blank symbol immediately to the right of x, although that blank symbol need not be explicitly represented in the configuration.

The symbols $\vdash$ and $\dashv$ are *not* part of M's tape alphabet, but only a device to mark the ends of configurations and to create extra blank symbols to the right and left of the input if required; more on this below.

Each transition of M is of the form $(p, a) \to (b, d, q)$, which means, "when in state p scanning symbol a, print b, move the tape head one cell in direction $d \in \{\text{left}, \text{right}\}$, and enter state q."

Now consider the following equations on Δ^*:

(E1) for each transition $(p, a) \to (b, \text{right}, q)$ of M, the equation $pa = bq$;

(E2) for each transition $(p, a) \to (b, \text{left}, q)$ of M and each $c \in \Gamma$, the equation $cpa = qcb$;

(E3) the equations $\vdash = \vdash \sqcup$ and $\dashv = \sqcup \dashv$.

Equations (E3) allow us to create extra blank symbols to the left and right of the input any time we need them and to destroy them if we do not.

For $x, y \in \Delta^*$, we write $x \approx y$ if x and y are congruent modulo (E1)–(E3), and we write $x \sim y$ if x and y are congruent modulo (E3) only.

Lemma 22.2 *If $x, y \in \Gamma$ and t is a halt state, then*

$$\vdash xsy \dashv \;\approx\; \vdash ztw \dashv \quad\Leftrightarrow\quad \vdash xsy \dashv \xrightarrow[M]{*} \vdash ztw \dashv. \tag{22.4}$$

Proof. See [33, Theorem 4.3, p. 98]. The chief concern is that monoid equations are reversible, whereas computations are not; thus it is conceivable that the left-hand side of (22.4) holds by some complicated sequence of substitutions modeling a zigzagging forwards-and-backwards computation even when the right-hand side of (22.4) does not. It can be shown that since M is deterministic and there are no transitions out of state t, this cannot happen. □

Main Results

Theorem 22.3 *The following complexity results hold for the problem of deciding whether a given Horn formula $E \Rightarrow s = t$ is true in all star-continuous Kleene algebras.*

(i) *When E consists of commutativity conditions (or for that matter, any monoid equations $x = y$ such that $|x| = |y|$), the problem is Π^0_1-complete.*

(ii) *When E consists of arbitrary monoid equations $x = y$, the problem is Π^0_2-complete.*

Proof. Using Lemma 22.1 and expressing an equation as the conjunction of two inequalities, we can reduce the problem to the conjunction of two instances of

$$\mathsf{Reg}\, \Sigma^*/E,\ \iota \;\models\; s \leq t. \tag{22.5}$$

The upper bounds for both (i) and (ii) are obtained by expressing (22.5) as a first-order formula with the appropriate quantifier prefix. Let $\equiv$ denote congruence modulo E on Σ^*. Applying (22.1) with $M = \Sigma^*$ and $M' = \Sigma^*/E$, (22.5) can be expressed

$$\forall x\ x \in \rho_{\Sigma^*}(s) \quad\Rightarrow\quad \exists y\ y \equiv x \wedge y \in \rho_{\Sigma^*}(t). \tag{22.6}$$

The predicates $x \in \rho_{\Sigma^*}(s)$ and $y \in \rho_{\Sigma^*}(t)$ are decidable, and efficiently so: this is just string matching with regular expressions. Thus the formula (22.6) is a Π^0_2 formula. Moreover, if all equations in E are length-preserving, then the existential subformula

$$\exists y\ y \equiv x \wedge y \in \rho_{\Sigma^*}(t)$$

is decidable, so (22.6) is equivalent to a Π^0_1 formula.

The lower bound for (i) uses the characterization of Lemma 22.1 and the result of Berstel [10] (see also [40, 66]) that (22.5) is undecidable. The reductions given in the cited references show that (22.5) is Π^0_1-hard. This result holds even when E consists only of commutativity conditions of the form $pq = qp$ for atomic p and q.

We prove the lower bound for (ii) by encoding the totality problem for Turing machines; that is, whether a given Turing machine halts on all inputs. Let M be a Turing machine of the form described above with a single halt state t. Assume without loss of generality that M erases its tape before halting. The totality problem is to decide whether

$$\vdash sa^n \dashv \quad \xrightarrow[M]{*} \quad \vdash t \dashv, \quad n \geq 0.$$

This is a well-known Π^0_2-complete problem. By Lemma 22.2, this is true iff

$$\mathsf{Reg}\,\Delta^*/E,\ \iota \quad \models \quad \vdash sa^n \dashv\ =\ \vdash t \dashv, \quad n \geq 0,$$

where E consists of equations (E1)–(E3). This is equivalent to

$$\mathsf{Reg}\,\Delta^*/E,\ \iota \quad \models \quad \vdash sa^n \dashv\ \leq\ \vdash t \dashv, \quad n \geq 0,$$

since $\{x\} \subseteq \{y\}$ iff $x = y$. By star-continuity, this is true iff

$$\mathsf{Reg}\,\Delta^*/E,\ \iota \quad \models \quad \vdash sa^* \dashv\ \leq\ \vdash t \dashv,$$

and by Lemma 22.1, this is true iff

$$\mathsf{KA}^* \quad \models \quad E\ \rightarrow\ \vdash sa^* \dashv\ \leq\ \vdash t \dashv.$$

□

Lecture 23

Π^1_1-Completeness of $\mathcal{H}$ KA*

In this lecture we prove that the universal Horn theory of the star-continuous Kleene algebras is Π^1_1-complete. This result is from [69].

Let $G = (\omega, R)$ be a recursive directed graph on vertices ω, the natural numbers. For $m \in \omega$, denote by $R(m)$ the set of R-successors of m:

$$R(m) \quad = \quad \{n \mid (m,n) \in R\}.$$

Let $\mathsf{WF} \subseteq \omega$ be the set of all m such that all R-paths out of m are finite. Alternatively, we could define WF as the least solution of the following recursive equation:

$$\mathsf{WF} \quad = \quad \{m \mid R(m) \subseteq \mathsf{WF}\}.$$

Let us call G *well-founded* if $0 \in \mathsf{WF}$; that is, if all R-paths out of 0 are finite.

A well-known Π^1_1-complete problem is:

> *Given a recursive graph (say by a total Turing machine accepting the set of encodings of edges $(m,n) \in R$), is it well-founded?*

We reduce this problem to $\mathcal{H}$ KA*, thereby showing that the latter problem is Π^1_1-hard.

By assumption, R is a recursive set, thus there is a total deterministic Turing machine M that decides whether $(m,n) \in R$. We can assume without loss of generality that M satisfies the restrictions on Turing machines imposed in the previous lecture and operates as follows.

In addition to its start state s, M has three halt states t, r, u. When started in configuration $\vdash a^m s a^n \dashv$, it first performs a check that the tape initially contains a contiguous string of a's surrounded by blanks and enters halt state u if not. It then determines whether $(m,n) \in R$. If so, it halts in configuration $\vdash a^n t \dashv$, and if not, it halts in configuration $\vdash r \dashv$. Thus

$$\vdash a^m s a^n \dashv \quad \xrightarrow[M]{*} \quad \begin{cases} \vdash a^n t \dashv, & \text{if } (m,n) \in R, \\ \vdash r \dashv, & \text{if } (m,n) \notin R. \end{cases}$$

By Lemma 22.2 of the previous lecture, we have

$$\begin{aligned} \vdash a^m s a^n \dashv \;\approx\; \vdash a^n t \dashv \quad &\Leftrightarrow \quad (m,n) \in R, \\ \vdash a^m s a^n \dashv \;\approx\; \vdash r \dashv \quad &\Leftrightarrow \quad (m,n) \notin R, \end{aligned}$$

where $\approx$ denotes congruence modulo equations (E1)–(E3) of the previous lecture.

Now consider the Kleene algebra equation

$$t \;\leq\; s a^*. \tag{23.1}$$

Let E be the set of equations (E1)–(E3) together with (23.1).

The following is our main lemma.

Lemma 23.1 *For all $m \geq 0$,*

$$\mathsf{KA}^* \;\models\; E \;\Rightarrow\; \vdash a^m t \dashv \;\leq\; \vdash r \dashv$$

if and only if $m \in \mathsf{WF}$.

Proof. The reverse implication ($\Leftarrow$) is proved by transfinite induction on the stages of the inductive definition of WF. Suppose that $m \in \mathsf{WF}$. Let $\tau : 2^\omega \to 2^\omega$ be the monotone map

$$\tau(A) \;=\; \{m \mid R(m) \subseteq A\}$$

and define

$$\begin{aligned} \tau^0(A) &= A \\ \tau^{\alpha+1}(A) &= \tau(\tau^\alpha(A)) \\ \tau^\lambda(A) &= \bigcup_{\alpha<\lambda} \tau^\alpha(A), \quad \lambda \text{ a limit ordinal.} \end{aligned}$$

Then

$$\mathsf{WF} \;=\; \bigcup_\alpha \tau^\alpha(\varnothing).$$

Let α be the smallest ordinal such that $m \in \tau^\alpha(\varnothing)$. Then α must be a successor ordinal $\beta+1$, therefore $m \in \tau(\tau^\beta(\varnothing))$, so $R(m) \subseteq \tau^\beta(\varnothing)$. By the induction hypothesis, if $n \in R(m)$, then

$$\mathsf{KA}^* \;\models\; E \;\Rightarrow\; \vdash a^n t \dashv \;\leq\; \vdash r \dashv,$$

and $\vdash a^m s a^n \dashv \;\approx\; \vdash a^n t \dashv$, therefore

$$\mathsf{KA}^* \;\models\; E \;\Rightarrow\; \vdash a^m s a^n \dashv \;\leq\; \vdash r \dashv.$$

For $n \notin R(m)$, $\vdash a^m s a^n \dashv \;\approx\; \vdash r \dashv$. Thus for all n,

$$\mathsf{KA}^* \;\models\; E \;\Rightarrow\; \vdash a^m s a^n \dashv \;\leq\; \vdash r \dashv.$$

By star-continuity,

$$\mathsf{KA}^* \;\models\; E \;\Rightarrow\; \vdash a^m s a^* \dashv \;\leq\; \vdash r \dashv,$$

and by (23.1),

$$\mathsf{KA}^* \;\models\; E \;\Rightarrow\; \vdash a^m t \dashv \;\leq\; \vdash r \dashv.$$

Conversely, for the forward implication ($\Rightarrow$), we construct a particular interpretation satisfying E in which for all $m \in \omega$, $\vdash a^m t \dashv \;\leq\; \vdash r \dashv$ implies $m \in \mathsf{WF}$.

For $A \subseteq \Delta^*$, define the monotone map

$$\sigma(A) \;=\; A \;\cup\; \{x \mid \exists y \in A\; x \approx y\} \;\cup\; \{utv \mid \forall n\; usa^n v \in A\}. \tag{23.2}$$

Call a subset of Δ^* *closed* if it is closed under the operation σ. The *closure* of A is the smallest closed set containing A and is denoted $\overline{A}$. Build a Kleene algebra consisting of the closed sets with operations

$$\begin{array}{rclcrcl} A \oplus B &=& \overline{A \cup B} & & 0 &=& \varnothing \\ A \odot B &=& \overline{AB} & & 1 &=& \{\varepsilon\}, \\ A^{\circledast} &=& \overline{\bigcup_n A^n} & & & & \end{array}$$

where ε is the null string and A^n is the n^{th} power of A under the operation $\odot$. It is not difficult to show that the family of closed sets forms a star-continuous Kleene algebra under these operations.

We show now that under the interpretation $a \mapsto \overline{\{a\}}$, the equations E are satisfied. For an equation $x = y$ of type (E1)–(E3), we need to show that $\overline{\{x\}} = \overline{\{y\}}$. It suffices to show that $x \in \overline{\{y\}}$ and $y \in \overline{\{x\}}$. But since $x \approx y$, this follows immediately from (23.2).

For the equation $t \leq sa^*$, we need to show that

$$t \in \overline{\{s\}} \odot \overline{\bigcup_n \overline{\{a\}}^n}.$$

It suffices to show $t \in \overline{\{sa^n \mid n \geq 0\}}$. Again, this follows immediately from (23.2).

Finally, we show that for $x \in \overline{\{\vdash r \dashv\}}$, either

(i) $x \xrightarrow[M]{*} \vdash r \dashv$;

(ii) $x \xrightarrow[M]{*} \vdash a^n t \dashv$ for some $n \in \mathsf{WF}$; or

(iii) $x \sim \vdash a^n t a^k \dashv$ for some $k \geq 1$.

The argument proceeds by transfinite induction on the inductive definition of closure:

$$\begin{aligned} \sigma^0(A) &= A \\ \sigma^{\alpha+1}(A) &= \sigma(\sigma^\alpha(A)) \\ \sigma^\lambda(A) &= \bigcup_{\alpha<\lambda} \sigma^\alpha(A), \quad \lambda \text{ a limit ordinal} \\ \overline{A} &= \bigcup_\alpha \sigma^\alpha(A). \end{aligned}$$

Let α be the least ordinal such that

$$x \in \sigma^\alpha(\{\vdash r \dashv\}).$$

Then α must be a successor ordinal $\beta + 1$, thus

$$x \in \sigma(\sigma^\beta(\{\vdash r \dashv\})).$$

There are two cases, one for each clause in the definition (23.2) of σ.

If there exists $y \in \sigma^\beta(\{\vdash r \dashv\})$ such that $x \approx y$, then by the induction hypothesis, y satisfies one of (i)–(iii), therefore so does x; the argument here is similar to [33, Theorem 4.3, p. 98].

Otherwise, $x = utv$ and

$$usa^n v \in \sigma^\beta(\{\vdash r \dashv\})$$

for all n. By the induction hypothesis, one of (i)–(iii) holds for each $usa^n v$. But (iii) is impossible because of the form of (E3). Moreover, by construction of M, each of (i) and (ii) implies that $u \sim \leq a^m$ and $v \sim a^k \dashv$ for

some k, m. Thus $x \sim \vdash a^m t a^k \dashv$. If $k \geq 1$, then x satisfies (iii). Otherwise, $x \sim \vdash a^m t \dashv$ and

$$\vdash a^m s a^n \dashv \quad \in \quad \sigma^\beta(\{\vdash r \dashv\})$$

for all n, therefore either (i) or (ii) holds for $\vdash a^m s a^n \dashv$. If (i), then $(m, n) \notin R$. If (ii), then $(m, n) \in R$ and $n \in \mathsf{WF}$. Thus $R(m) \subseteq \mathsf{WF}$ and $m \in \mathsf{WF}$. □

Theorem 23.2 *$\mathcal{H}\,\mathsf{KA}^*$ is Π^1_1-complete.*

Proof. Taking $m = 0$ in Lemma 23.1, we have

$$\mathsf{KA}^* \quad \models \quad E \;\Rightarrow \vdash t \dashv \;\leq\; \vdash r \dashv$$

if and only if G is well-founded. This gives the desired lower bound. The upper bound follows from the form of the infinitary axiomatization of star-continuous Kleene algebra; validity is equivalent to the existence of a well-founded proof tree. □

Lecture 24

Coalgebraic Theory of KA and KAT

Traditionally, KAT is axiomatized equationally [68]. In [21], Chen and Pucella develop a coalgebraic theory of KAT inspired by Rutten's coalgebraic theory of KA based on deterministic automata [105]. Remarking that "the known automata-theoretic presentation of KAT [72] does not lend itself to a coalgebraic theory," and that "the notion of derivative, essential to the coinduction proof principle in this context, is not readily definable for KAT expressions as defined in [68]," Chen and Pucella develop a new interpretation of KAT expressions and a corresponding automata theory differing from [72] in several respects. They give a coinductive proof principle and show how it can be used to establish equivalence of expressions. This gives an alternative to equational proofs using the standard axiomatization [68] or by minimization of deterministic automata [72].

The ability to generate equivalence proofs automatically has important implications for proof-carrying code. Chen and Pucella argue that the coalgebraic approach makes this possible, since proofs can be generated purely mechanically via repeated application of the Brzozowski derivative, whereas classical equational logic "requires creativity" [21]. This is not strictly true, as Worthington [118] has shown that equational proofs can also be generated automatically. However, it is fair to say that the coinductive approach does provide a more natural mechanism.

Still unresolved is the issue of proof complexity in the coinductive setting. Chen and Pucella claim that coinduction can give shorter proofs,

but they give no supporting evidence. Worthington's technique is known to require PSPACE and to produce exponential-size proofs in the worst case. This worst-case bound is unlikely to be significantly improved, as the equational theory of KAT is PSPACE-complete [28].

Chen and Pucella's treatment has a few technical shortcomings, as they themselves point out. In their words:

> The "path independence" of a mixed automaton gives any mixed automaton a certain form of redundancy. This redundancy persists in the definition of bisimulation... An open question is to cleanly eliminate this redundancy; a particular motivation for doing this would be to make proofs of expression equivalence as simple as possible. Along these lines, it would be of interest to develop other weaker notions of bisimulation that give rise to bisimulations; pseudo-bisimulations require a sort of "fixed variable ordering" that does not seem absolutely necessary...
>
> Another issue for future work would be to give a class of expressions wider than our mixed expressions for which there are readily understandable and applicable rules for computing derivatives. In particular, a methodology for computing derivatives of the KAT expressions defined by Kozen [68] would be nice to see. Intuitively, there seems to be a tradeoff between the expressiveness of the regular expression language and the simplicity of computing derivatives (in the context of KAT). Formal tools for understanding this tradeoff could potentially be quite useful. [21]

This paper addresses these issues. We develop a coalgebraic theory of KAT, which we call KCT, along the lines of [21, 105]. Our treatment includes a new definition of the Brzozowski derivative, but in the context of the original automata-theoretic formulation of KAT involving automata on guarded strings [72]. The syntactic form of the Brzozowski derivative applies to all KAT expressions as defined in [68]. The somewhat artificial concepts of path independence, fixed variable ordering, and pseudo-bisimulation do not arise in this setting. This treatment places KCT within the general coalgebraic framework described by Bonsangue, Rutten, and Silva [17, 18].

We also give a complexity analysis of the coinductive proof principle. We show that an efficient implementation is tantamount to the construction of nondeterministic automata from the given expressions by a Kleene construction, determinizing the two automata by a standard subset construction, and constructing a bisimulation on states of the resulting deterministic automata. It follows that coinductive equivalence proofs can be

generated automatically in PSPACE. This matches Worthington's bound [118] for equational proofs.

24 Automata on Guarded Strings

Automata on guarded strings (AGS), also known as automata with tests, were introduced in [72]. They are a generalization of ordinary finite-state automata to include tests. An ordinary automaton with null transitions is an AGS over the two-element Boolean algebra.

24.1 Guarded Strings

Recall that a *guarded string* over P, B is an alternating sequence

$$\alpha_0 p_1 \alpha_1 p_2 \cdots \alpha_{n-1} p_n \alpha_n,$$

where $p_i \in \mathsf{P}$ and the α_i are atoms (minimal nonzero elements) of the free Boolean algebra B generated by B. The set of atoms is denoted At. The elements of At can be regarded either as conjunctions of literals of B (elements of B or their negations) or as truth assignments to B. A *guarded string* is thus an element of $(\mathsf{At} \cdot \mathsf{P})^* \cdot \mathsf{At}$. The set of all guarded strings is denoted GS. Guarded strings represent the join-irreducible elements of the free KAT on generators P, B.

24.2 Nondeterministic Automata

A nondeterministic AGS consists of a labeled directed graph with two types of transitions, *action transitions* labeled with elements of P and *test transitions* labeled with elements of B. There is a set START of *start states* and a set ACCEPT of *accept states*.

An input to an AGS is a guarded string $\alpha_0 p_0 \alpha_1 \cdots \alpha_{n-1} p_{n-1} \alpha_n$. Intuitively, it operates as follows. It starts with a pebble on a nondeterministically chosen start state and its input head scanning α_0. In the course of the computation, say the pebble is occupying state s and the input head is scanning α_i. If $i < n$, it may read the next action symbol p_i from the input string and move the pebble to any nondeterministically chosen state t such that there is an action transition from s to t with label p_i. The input head is advanced past p_i in the input string and is now scanning α_{i+1}. Also while scanning α_i, it may slide the pebble along an enabled test transition at any time without advancing the input head. A test transition is *enabled* if $\alpha_i \leq b$, where b is the label of the transition. The automaton accepts if it is ever scanning α_n while the pebble is on an accept state. Thus

the automaton accepts a guarded string x if there is a directed path π from START to ACCEPT such that $x \leq e$, where e is the product of the labels of the edges along π.

Formally, a (nondeterministic) *automaton on guarded strings* (AGS) over P and B is a tuple

$$M = (Q, \Delta, \text{START}, \text{ACCEPT}),$$

where Q is a set of *states*, $\text{START} \subseteq Q$ are the *start states*, $\text{ACCEPT} \subseteq Q$ are the *accept states*, and Δ is the *transition function*

$$\Delta : (\mathsf{P} + \mathsf{At}) \to Q \to 2^Q,$$

where $+$ denotes disjoint (marked) union.

The definition of acceptance involves the Kleisli composition $\bullet$ and asterate † operations on maps $Q \to 2^Q$ defined by:

$$R \bullet S \stackrel{\text{def}}{=} s \mapsto \bigcup_{t \in S(s)} R(t) \qquad R^0 \stackrel{\text{def}}{=} s \mapsto \{s\}$$

$$R^\dagger \stackrel{\text{def}}{=} \bigcup_{n \geq 0} R^n \qquad R^{n+1} \stackrel{\text{def}}{=} R \bullet R^n.$$

The map Δ generates a map

$$\widehat{\Delta} : (\mathsf{P} + \mathsf{At}) \to Q \to 2^Q$$

defined by

$$\widehat{\Delta}_\alpha \stackrel{\text{def}}{=} \Delta_\alpha^\dagger \qquad \widehat{\Delta}_p \stackrel{\text{def}}{=} \Delta_p.$$

Intuitively, $\widehat{\Delta}_\alpha(s)$ accumulates all states accessible from s by a sequence of test transitions enabled under α. The map $\widehat{\Delta}$ extends further to a monoid homomorphism

$$\widehat{\Delta} : (\mathsf{P} + \mathsf{At})^* \to Q \to 2^Q$$

from the free monoid $(\mathsf{P} + \mathsf{At})^*$ to the monoid $Q \to 2^Q$ under Kleisli composition. Thus

$$\widehat{\Delta}_\varepsilon \stackrel{\text{def}}{=} s \mapsto \{s\} \qquad \widehat{\Delta}_{xy} \stackrel{\text{def}}{=} \widehat{\Delta}_y \bullet \widehat{\Delta}_x.$$

The guarded strings $\mathsf{GS} = (\mathsf{At} \cdot \mathsf{P})^* \cdot \mathsf{At}$ form a submonoid of $(\mathsf{P} + \mathsf{At})^*$. The automaton M *accepts* $x \in \mathsf{GS}$ if there exist $s \in \text{START}$ and $t \in \text{ACCEPT}$ such that $t \in \widehat{\Delta}_x(s)$. The set of guarded strings accepted by M is denoted $\mathsf{GS}(M)$.

24.3 Deterministic Automata

The definition of deterministic AGS here differs from that of [72] so as to conform to the coalgebraic structure to be introduced in §24, but the difference is inessential. In [72] the set of states of a deterministic AGS is separated into disjoint sets of *action states* and *test states*, whereas here we have elided the action states.

A *deterministic automaton on guarded strings* (AGS) over P and B is a structure

$$M \;=\; (Q,\, \delta,\, \varepsilon,\, \textsc{start}),$$

where Q is a set of *states*, $\textsc{start} \in Q$ is the *start state*, and

$$\delta : \mathsf{At} \cdot \mathsf{P} \;\to\; Q \;\to\; Q \qquad\qquad \varepsilon : \mathsf{At} \;\to\; Q \;\to\; 2$$

with components

$$\delta_{\alpha p} : Q \;\to\; Q \qquad\qquad \varepsilon_\alpha : Q \;\to\; 2$$

for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$. The components ε_α play the same role as the accept states in a nondeterministic automaton.

Define the function $Ł : Q \to \mathsf{GS} \to 2$ inductively as follows:

$$Ł(u)(\alpha) \stackrel{\text{def}}{=} \varepsilon_\alpha(u) \qquad\qquad Ł(u)(\alpha p y) \stackrel{\text{def}}{=} Ł(\delta_{\alpha p}(u))(y), \tag{24.1}$$

where $y \in \mathsf{GS}$, $\alpha \in \mathsf{At}$, and $p \in \mathsf{P}$. The machine is said to *accept* $x \in \mathsf{GS}$ if $Ł(\textsc{start})(x) = 1$. The set of guarded strings accepted by M is denoted $\mathsf{GS}(M)$. Identifying a subset of GS with its characteristic function $\mathsf{GS} \to 2$, we can write $\mathsf{GS}(M) = Ł(\textsc{start})$.

The map δ extends to a monoid homomorphism $\widehat{\delta} : (\mathsf{At} \cdot \mathsf{P})^* \to Q \to Q$ from the free monoid $(\mathsf{At} \cdot \mathsf{P})^*$ to the monoid $Q \to Q$ under functional composition.

24.4 Determinization

Determinization is effected by a subset construction similar to that for ordinary automata. Given a nondeterministic AGS

$$N \;=\; (Q,\, \Delta,\, \textsc{start},\, \textsc{accept}),$$

there is an equivalent deterministic AGS

$$M \;=\; (2^Q,\, \delta,\, \varepsilon,\, \textsc{start}),$$

where for $A \subseteq Q$,

$$\varepsilon_\alpha(A) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } \exists s \in A\ \exists t \in \textsc{accept}\ t \in \widehat{\Delta}_\alpha(s), \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_{\alpha p}(A) \stackrel{\text{def}}{=} \bigcup_{s \in A} \widehat{\Delta}_{\alpha p}(s).$$

One can show by a straightforward induction on the length of $x \in \mathsf{GS}$ that for all $A \subseteq Q$,

$$Ł(A)(x) \;=\; \begin{cases} 1, & \text{if } \exists s \in A\ \exists t \in \textsc{accept}\ t \in \widehat{\Delta}_x(s), \\ 0, & \text{otherwise;} \end{cases}$$

in particular,

$$Ł(\textsc{start})(x) = 1 \;\Leftrightarrow\; \exists s \in \textsc{start}\ \exists t \in \textsc{accept}\ t \in \widehat{\Delta}_x(s).$$

As these are exactly the acceptance criteria for M and N respectively, the two machines accept the same set of guarded strings.

24 Kleene Coalgebra with Tests (KCT)

A Kleene coalgebra with tests (KCT) is very much like Kleene coalgebra (KC) [105], but with the addition of Boolean tests. Formally, a *Kleene coalgebra with tests* (KCT) over P and B is a structure

$$M \;=\; (Q, \delta, \varepsilon),$$

where Q is a set of *states* and

$$\delta : \mathsf{At} \cdot \mathsf{P} \;\to\; Q \;\to\; Q \qquad\qquad \varepsilon : \mathsf{At} \;\to\; Q \;\to\; 2$$

for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$, exactly as in deterministic automata on guarded strings. Thus we can view a KCT as simply a deterministic AGS without a designated start state.

A KCT *morphism* $h : (Q, \delta, \varepsilon) \to (Q', \delta', \varepsilon')$ is a set map $h : Q \to Q'$ that commutes with δ, δ' and $\varepsilon, \varepsilon'$; that is,

$$\delta'_{\alpha p}(h(u)) \;=\; h(\delta_{\alpha p}(u)) \qquad\qquad \varepsilon'_\alpha(h(u)) \;=\; \varepsilon_\alpha(u).$$

We denote the category of KCTs and KCT morphisms over P and B also by KCT.

24.1 The Brzozowski Derivative

There is a natural KCT over P and B defined in terms of the Brzozowski derivative on sets of guarded strings. The traditional Brzozowski derivative [19] is a kind of residuation operator on sets of ordinary strings. The current form is quite similar, except that we extend the definition to accommodate tests.

We define two maps

$$D : \mathsf{At} \cdot \mathsf{P} \to 2^{\mathsf{GS}} \to 2^{\mathsf{GS}} \qquad E : \mathsf{At} \to 2^{\mathsf{GS}} \to 2,$$

where for $R \subseteq \mathsf{GS}$,

$$D_{\alpha p}(R) \stackrel{\text{def}}{=} \{y \in \mathsf{GS} \mid \alpha p y \in R\} \qquad E_\alpha(R) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } \alpha \in R, \\ 0, & \text{if } \alpha \notin R. \end{cases}$$

It is clear that the structure

$$\mathsf{Brz} \stackrel{\text{def}}{=} (2^{\mathsf{GS}}, D, E)$$

forms a KCT. Indeed, it is the final object in the category KCT: for any KCT $M = (Q, \delta, \varepsilon)$, the function $Ł : Q \to 2^{\mathsf{GS}}$ defined in (24.1) is the unique KCT morphism $Ł : M \to \mathsf{Brz}$.

24.2 The Brzozowski Derivative, Syntactic Form

As with Brzozowski's original formulation [19], there is also a syntactic form of the Brzozowski derivative defined on KAT expressions. Let Exp denote the set of KAT expressions over P and B. We define a family of derivative operators

$$D : \mathsf{At} \cdot \mathsf{P} \to \mathsf{Exp} \to \mathsf{Exp} \qquad E : \mathsf{At} \to \mathsf{Exp} \to 2$$

consisting of components

$$D_{\alpha p} : \mathsf{Exp} \to \mathsf{Exp} \qquad E_\alpha : \mathsf{Exp} \to 2$$

defined inductively as follows. For $\alpha \in \mathsf{At}$, $p, q \in \mathsf{P}$, and $b \in B$,

$$\begin{aligned} D_{\alpha p}(e_1 + e_2) &\stackrel{\text{def}}{=} D_{\alpha p}(e_1) + D_{\alpha p}(e_2) \\ D_{\alpha p}(e_1 e_2) &\stackrel{\text{def}}{=} D_{\alpha p}(e_1)\, e_2 + E_\alpha(e_1)\, D_{\alpha p}(e_2) \\ D_{\alpha p}(e^*) &\stackrel{\text{def}}{=} D_{\alpha p}(e)\, e^* \end{aligned} \qquad \begin{aligned} D_{\alpha p}(q) &\stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } p = q, \\ 0, & \text{otherwise,} \end{cases} \\ D_{\alpha p}(b) &\stackrel{\text{def}}{=} 0. \end{aligned}$$

$$\begin{aligned} E_\alpha(e_1 + e_2) &\stackrel{\text{def}}{=} E_\alpha(e_1) + E_\alpha(e_2) \\ E_\alpha(e_1 e_2) &\stackrel{\text{def}}{=} E_\alpha(e_1)\, E_\alpha(e_2) \\ E_\alpha(e^*) &\stackrel{\text{def}}{=} 1 \end{aligned} \qquad \begin{aligned} E_\alpha(b) &\stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } \alpha \leq b, \\ 0, & \text{otherwise,} \end{cases} \\ E_\alpha(q) &\stackrel{\text{def}}{=} 0. \end{aligned}$$

These operators on KAT expressions are collectively called the *syntactic Brzozowski derivative*.

The map E_α is just the evaluation morphism that for any KAT expression substitutes 0 for any $p \in \mathsf{P}$, 1 for any $b \in \mathsf{B}$ such that $\alpha \leq b$, and 0 for any $b \in \mathsf{B}$ such that $\alpha \leq \overline{b}$, then simplifies the resulting expression over the two-element Kleene algebra 2. It is easily shown that for any KAT expression e,

$$E_\alpha(e) \;=\; \begin{cases} 1, & \text{if } \alpha \leq e, \\ 0, & \text{if } \alpha \not\leq e \end{cases} \;=\; \begin{cases} 1, & \text{if } \alpha \in \mathsf{GS}(e), \\ 0, & \text{if } \alpha \notin \mathsf{GS}(e). \end{cases}$$

The structure

$$(\mathsf{Exp},\, D,\, E)$$

is a KCT in the sense of §24, thus there is a unique KCT morphism $Ł : \mathsf{Exp} \to \mathsf{Brz}$ to the final coalgebra Brz defined in (24.1). We will show that $Ł(e) = \mathsf{GS}(e)$, where GS is the traditional interpretation of KAT expressions mentioned in §**??**.

Lemma 24.1 *For all* $\alpha \in \mathsf{At}$, $p \in \mathsf{P}$, *and* $e, e' \in \mathsf{Exp}$,

$$\alpha p e' \leq e \;\Leftrightarrow\; e' \leq D_{\alpha p}(e).$$

Proof. For the forward implication,

$$D_{\alpha p}(\alpha p e') \;=\; D_{\alpha p}(\alpha) p e' + E_\alpha(\alpha) D_{\alpha p}(p) e' + E_\alpha(\alpha) E_\alpha(p) D_{\alpha p}(e') \;=\; e'.$$

By monotonicity of $D_{\alpha p}$,

$$\alpha p e' \leq e \;\Rightarrow\; e' = D_{\alpha p}(\alpha p e') \leq D_{\alpha p}(e).$$

For the reverse implication, it suffices to show $\alpha p D_{\alpha p}(e) \leq e$. We proceed by induction on the structure of e. For $p \in \mathsf{P}$,

$$\alpha p D_{\alpha p}(p) \;=\; \alpha p \;\leq\; p.$$

For the case $e_1 e_2$,

$$\begin{aligned} \alpha p D_{\alpha p}(e_1 e_2) \;&=\; \alpha p D_{\alpha p}(e_1) e_2 + \alpha p E_\alpha(e_1) D_{\alpha p}(e_2) \\ &=\; \alpha p D_{\alpha p}(e_1) e_2 + \alpha E_\alpha(e_1) \alpha p D_{\alpha p}(e_2) \\ &\leq\; e_1 e_2. \end{aligned}$$

For the case e^*,

$$\alpha p D_{\alpha p}(e^*) \;=\; \alpha p D_{\alpha p}(e) e^* \;\leq\; e e^* \;\leq\; e^*.$$

All other cases are equally straightforward. □

Theorem 24.2 *For all* KAT *expressions* e, $\mathsf{GS}(e) = Ł(e)$. *Thus the set accepted by the automaton* (Exp, D, E, e) *is* $\mathsf{GS}(e)$.

Proof. We wish to show that for all $x \in \mathsf{GS}$, $x \in \mathsf{GS}(e)$ iff $Ł(e)(x) = 1$. By the completeness theorem for KAT [75], we have $x \in \mathsf{GS}(e)$ iff $x \leq e$, so it suffices to show that $x \leq e$ iff $L(e)(x) = 1$. We proceed by induction on the length of x. The basis for x an atom α is immediate from the definition of E_α. For $x = \alpha p y$, by Lemma 24.1,

$$\alpha p y \leq e \Leftrightarrow \quad y \leq D_{\alpha p}(e) \quad \Leftrightarrow \quad Ł(D_{\alpha p}(e))(y) = 1 \quad \Leftrightarrow \quad Ł(e)(apy) = 1.$$

□

24 Completeness

24.1 Bisimulation on KCTs

A *bisimulation* between two KCTs $M = (Q, \delta, \varepsilon)$ and $M' = (Q', \delta', \varepsilon')$ is a binary relation $\equiv$ between Q and Q' such that if $s \in Q$, $t \in Q'$, and $s \equiv t$, then for all $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$,

(i) $\varepsilon_\alpha(s) = \varepsilon'_\alpha(t)$; and

(ii) $\delta_{\alpha p}(s) \equiv \delta'_{\alpha p}(t)$.

Lemma 24.1 *The relation*

$$s \mathrel{\widehat{\equiv}} t \stackrel{\text{def}}{\iff} Ł(s) = Ł(t)$$

is the unique maximal bisimulation between M *and* M'.

Proof. It is easily shown that $\mathrel{\widehat{\equiv}}$ satisfies (i) and (ii). Moreover, if $\equiv$ is any relation satisfying (i) and (ii), one can show by a straightforward inductive argument that $\equiv$ refines $\mathrel{\widehat{\equiv}}$, thus $\mathrel{\widehat{\equiv}}$ is the unique maximal relation satisfying (i) and (ii). □

An *autobisimulation* is a bisimulation between M and itself. Bisimulations are closed under relational composition and arbitrary union, and the identity relation is an autobisimulation. Thus the reflexive, symmetric, and transitive closure of an autobisimulation is again an autobisimulation. An autobisimulation that is so closed is called a KCT-*congruence*. KCT-congruences are exactly the kernels of KCT-morphisms.

A KCT is bisimilar to its quotient by any KCT-congruence under the map $\{(s, [s]) \mid s \in Q\}$, where $[s]$ is the KCT-congruence class of s. The quotient by the unique maximal autobisimulation is a sub-coalgebra of Brz, the final coalgebra.

24.2 Bisimulation on Deterministic Automata

For deterministic automata, we add an extra condition. A *bisimulation* between two deterministic AGS $M = (Q, \delta, \varepsilon, \text{START})$ and $M' = (Q', \delta', \varepsilon', \text{START}')$ is a bisimulation $\equiv$ between the underlying KCTs (Q, δ, ε) and $(Q', \delta', \varepsilon')$ such that $\text{START} \equiv \text{START}'$. Two automata are *bisimilar* if there exists a bisimulation between them.

Lemma 24.2 *M and M' are bisimilar iff $\mathsf{GS}(M) = \mathsf{GS}(M')$.*

Proof. Let $\widehat{\equiv}$ be the relation defined in the proof of Lemma 24.1. If $\mathsf{GS}(M) = \mathsf{GS}(M')$, then $Ł(\text{START}) = Ł(\text{START}')$ by the definition of acceptance, therefore $\text{START} \mathrel{\widehat{\equiv}} \text{START}'$. Then M and M' are bisimilar under $\widehat{\equiv}$.

Conversely, if there exists a bisimulation $\equiv$ between M and M', then $\text{START} \equiv \text{START}'$, and by Lemma 24.1, $\equiv$ refines $\widehat{\equiv}$, therefore $\text{START} \mathrel{\widehat{\equiv}} \text{START}'$. Thus $\widehat{\equiv}$ is a bisimulation of automata. □

The quotient of an automaton by its unique maximal autobisimulation gives the unique minimal equivalent automaton (ignoring inaccessible states).

Theorem 24.3 (Completeness) *The following are equivalent:*

(i) *the automata* (Exp, D, E, e) *and* (Exp, D, E, e') *are bisimilar;*

(ii) $Ł(e) = Ł(e')$*;*

(iii) $\mathsf{GS}(e) = \mathsf{GS}(e')$*;*

(iv) *e and e' are equivalent modulo the axioms of* KAT.

Proof. The equivalence of (i)–(iii) follows from Theorem 24.2 and Lemma 24.2. The equivalence of (iii) and (iv) are just the soundness and completeness of KAT for the guarded string model [75]. □

24 Complexity

Let Exp_e denote the subautomaton of (Exp, D, E, e) consisting of those expressions that are accessible from e; that is, those expressions of the form $\widehat{D}_x(e)$ for some $x \in (\mathsf{At} \cdot \mathsf{P})^*$. Theorem 24.3 by itself is not very useful as a deductive system or decision procedure for equivalence, because Exp_e is not a finite system in general. However, equivalent finite systems exist. In particular, by Theorem 24.3, KAT equivalence is the maximal congruence

on Exp. The quotient with respect to this relation, ignoring inaccessible states, gives the minimal deterministic AGS accepting $\mathsf{GS}(e)$, which is finite since $\mathsf{GS}(e)$ is regular.

Unfortunately, to construct this automaton directly, we would need an independent algorithm to decide KAT equivalence. However, we can obtain finite automata with finer congruences that are easier to decide than full KAT equivalence. Chen and Pucella [21] use equivalence modulo additive associativity, commutativity, and idempotence (ACI-equivalence). Here we consider equivalence modulo the axioms of idempotent commutative monoids for $+, 0$ and the axioms

$$1 \cdot x = x \qquad 0 \cdot x = 0 \qquad (x + y) \cdot z = xz + yz. \tag{24.2}$$

Multiplicative associativity is not assumed, nor is left distributivity. We might call structures satisfying these axioms *right presemirings*. We denote by $\approx$ the KAT-congruence on terms generated by these axioms. We will show that $\mathsf{Exp}_e/\!\approx$ has finitely many accessible classes. It is a coarser relation than ACI-equivalence, therefore has fewer classes, but is still easy to decide, as there are normal forms up to additive commutativity. Of course, it makes the most sense to use the coarsest relation possible that is easily decidable, because coarser relations give smaller automata.

Because there are only finitely many $\approx$-classes accessible from e, the quotient automaton $\mathsf{Exp}_e/\!\approx$ is finite, and we can use it to obtain finite coinductive equivalence proofs. More interestingly, we will also show that $\mathsf{Exp}_e/\!\approx$ is a homomorphic image of a deterministic automaton M_e obtained by creating a nondeterministic AGS N_e from the expression e by a Kleene construction, then determinizing N_e by a subset construction as described in §24.4. This characterization gives a bound on the size of $\mathsf{Exp}_e/\!\approx$, which we can then use to argue that coinductive equivalence proofs can be generated automatically in PSPACE.

Lemma 24.1 *The relation $\approx$ is a* KCT*-congruence on* Exp.

Proof. We must show that if $e \approx e'$, then $E_\alpha(e) = E_\alpha(e')$ and $D_{\alpha p}(e) \approx D_{\alpha p}(e')$. The first conclusion follows from Theorem 24.3 and the fact that $\approx$ refines KAT-equivalence.

For the additive axioms of idempotent commutative monoids, the second conclusion follows from the additivity of $D_{\alpha p}$.

For the axioms (24.2),

$$\begin{aligned}
D_{\alpha p}(1x) &= D_{\alpha p}(1)x + E_\alpha(1)D_{\alpha p}(x) = 0x + 1D_{\alpha p}(x) \approx D_{\alpha p}(x)\\
D_{\alpha p}(0x) &= D_{\alpha p}(0)x + E_\alpha(0)D_{\alpha p}(x) = 0x + 0D_{\alpha p}(x) \approx D_{\alpha p}(0)\\
D_{\alpha p}((x+y)z) &= (D_{\alpha p}(x) + D_{\alpha p}(y))z + (E_\alpha(x) + E_\alpha(y))D_{\alpha p}(z)\\
&\approx D_{\alpha p}(x)z + E_\alpha(x)D_{\alpha p}(z) + D_{\alpha p}(y)z + E_\alpha(y)D_{\alpha p}(z)\\
&= D_{\alpha p}(xz + yz).
\end{aligned}$$

Finally, we must show that if $e_1 \approx e_2$, then $D_{\alpha p}(e_1 + e_3) \approx D_{\alpha p}(e_2 + e_3)$, $D_{\alpha p}(e_1 e_3) \approx D_{\alpha p}(e_2 e_3)$, $D_{\alpha p}(e_3 e_1) \approx D_{\alpha p}(e_3 e_2)$, and $D_{\alpha p}(e_1^*) \approx D_{\alpha p}(e_2^*)$. These arguments are all quite easy. For example,

$$\begin{aligned} D_{\alpha p}(e_1 e_3) &= D_{\alpha p}(e_1) e_3 + E_\alpha(e_1) D_{\alpha p}(e_3) \\ &\approx D_{\alpha p}(e_2) e_3 + E_\alpha(e_2) D_{\alpha p}(e_3) = D_{\alpha p}(e_2 e_3) \end{aligned}$$

and

$$D_{\alpha p}(e_1^*) = D_{\alpha p}(e_1) e_1^* \approx D_{\alpha p}(e_2) e_2^* = D_{\alpha p}(e_2^*).$$

□

24.1 Closure

To establish the finiteness of the quotient automaton $\mathsf{Exp}_e / \approx$ and explain its relationship to the Kleene construction, we derive a formal relationship between the set of accessible $\approx$-classes of derivatives $\{\widehat{D}_x(e)/\approx \mid x \in (\mathsf{At} \cdot \mathsf{P})^*\}$ and certain sets of terms derived from e.

For KAT term e, we define the *closure* of e, denoted $\mathrm{cl}(e)$, to be the smallest set of terms containing e and 1 and closed under the following rules:

$$\begin{array}{lll} \dfrac{e \in \mathrm{cl}(e_1)}{e \in \mathrm{cl}(e_1 + e_2)} & \dfrac{e \in \mathrm{cl}(e_1)}{e e_2 \in \mathrm{cl}(e_1 e_2)} & \dfrac{e \in \mathrm{cl}(e_1)}{e e_1^* \in \mathrm{cl}(e_1^*)} \\ \dfrac{e \in \mathrm{cl}(e_2)}{e \in \mathrm{cl}(e_1 + e_2)} & \dfrac{e \in \mathrm{cl}(e_2)}{e \in \mathrm{cl}(e_1 e_2)} & \dfrac{e \in \mathrm{cl}(b)}{e \in \mathrm{cl}(\overline{b})} \end{array} \tag{24.3}$$

Lemma 24.2 *The set* $\mathrm{cl}(e)$ *contains at most* $|e| + 1$ *elements, where* $|e|$ *is the number of subterms of* e*.*

Proof. We show by induction on e that $\mathrm{cl}'(e)$ contains at most $|e|$ elements, where $\mathrm{cl}'(e) = \mathrm{cl}(e) - \{1\}$. For $e \in \mathsf{P} \cup \mathsf{B}$, $\mathrm{cl}'(e) = \{e\}$. For the other operators, from the rules (24.3) we have

$$\begin{aligned} \mathrm{cl}'(\overline{b}) &= \{\overline{b}\} \cup \mathrm{cl}'(b), \\ \mathrm{cl}'(e_1 + e_2) &= \{e_1 + e_2\} \cup \mathrm{cl}'(e_1) \cup \mathrm{cl}'(e_2), \\ \mathrm{cl}'(e_1 e_2) &= \{e_1 e_2\} \cup \{e e_2 \mid e \in \mathrm{cl}'(e_1)\} \cup \mathrm{cl}'(e_2), \\ \mathrm{cl}'(e_1^*) &= \{e_1^*\} \cup \{e e_1^* \mid e \in \mathrm{cl}'(e_1)\}. \end{aligned}$$

The result follows. □

24.2 Set Representation of Derivatives

We now construct a nondeterministic transition function Δ on the set of states $\mathsf{Exp} + (\mathsf{At} \times \mathsf{Exp})$ as follows. The elements of Exp are called *test states* and the elements of $\mathsf{At} \times \mathsf{Exp}$ are called *action states*. The test transitions go only from test states to action states, and the action transitions go only from action states to test states. Thus for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$,

$$\Delta_\alpha : \mathsf{Exp} \to 2^{\mathsf{At} \times \mathsf{Exp}}$$
$$\Delta_p : \mathsf{At} \times \mathsf{Exp} \to 2^{\mathsf{Exp}}.$$

The test transitions are deterministic: $\Delta_\alpha(e) \stackrel{\text{def}}{=} \{(\alpha, e)\}$. The action transitions are defined inductively:

$$\begin{aligned}
\Delta_p(\alpha, q) &\stackrel{\text{def}}{=} \begin{cases} \{1\}, & \text{if } q \in \mathsf{P} \text{ and } q = p, \\ \varnothing, & \text{if } q \in \mathsf{P} \text{ and } q \neq p \text{ or } q \in B, \end{cases} \\
\Delta_p(\alpha, e_1 + e_2) &\stackrel{\text{def}}{=} \Delta_p(\alpha, e_1) \cup \Delta_p(\alpha, e_2), \\
\Delta_p(\alpha, e_1 e_2) &\stackrel{\text{def}}{=} \begin{cases} \{e e_2 \mid e \in \Delta_p(\alpha, e_1)\} \cup \Delta_p(\alpha, e_2), & \text{if } E_\alpha(e_1) = 1, \\ \{e e_2 \mid e \in \Delta_p(\alpha, e_1)\}, & \text{if } E_\alpha(e_1) = 0, \end{cases} \\
\Delta_p(\alpha, e_1^*) &\stackrel{\text{def}}{=} \{e e_1^* \mid e \in \Delta_p(\alpha, e_1)\}.
\end{aligned}$$

Due to the bipartite structure of the states, we have $\widehat{\Delta}_{\alpha p} = \Delta_p \bullet \Delta_\alpha$, where $\widehat{\Delta}$ is the extension of Δ defined in §24.2. Then

$$\widehat{\Delta}_{\alpha p}(e) \;=\; (\Delta_p \bullet \Delta_\alpha)(e) \;=\; \bigcup \{\Delta_p(\alpha, e)\} \;=\; \Delta_p(\alpha, e). \tag{24.4}$$

We thus have

$$\begin{aligned}
\widehat{\Delta}_{\alpha p}(q) &\stackrel{\text{def}}{=} \begin{cases} \{1\}, & \text{if } q \in \mathsf{P} \text{ and } q = p, \\ \varnothing, & \text{if } q \in \mathsf{P} \text{ and } q \neq p \text{ or } q \in B, \end{cases} \\
\widehat{\Delta}_{\alpha p}(e_1 + e_2) &\stackrel{\text{def}}{=} \widehat{\Delta}_{\alpha p}(e_1) \cup \widehat{\Delta}_{\alpha p}(e_2), \\
\widehat{\Delta}_{\alpha p}(e_1 e_2) &\stackrel{\text{def}}{=} \begin{cases} \{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \cup \widehat{\Delta}_{\alpha p}(e_2), & \text{if } E_\alpha(e_1) = 1, \\ \{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0, \end{cases} \\
\widehat{\Delta}_{\alpha p}(e_1^*) &\stackrel{\text{def}}{=} \{e e_1^* \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}.
\end{aligned}$$

Lemma 24.3 *For all* KAT *terms e and $x \in (\mathsf{At} \cdot \mathsf{P})^*$, $\widehat{\Delta}_x(e) \subseteq \mathrm{cl}(e)$.*

Proof. We first show that for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$, $\widehat{\Delta}_{\alpha p}(e) \subseteq \mathrm{cl}(e)$ by induction on the structure of e. The cases $e \in \mathsf{P}$ or $e \in B$ are easy. For the

other operators,

$$\begin{aligned}
\widehat{\Delta}_{\alpha p}(e_1 + e_2) &= \widehat{\Delta}_{\alpha p}(e_1) \cup \widehat{\Delta}_{\alpha p}(e_2) \subseteq \mathrm{cl}(e_1) \cup \mathrm{cl}(e_2) \subseteq \mathrm{cl}(e_1 + e_2) \\
\widehat{\Delta}_{\alpha p}(e_1 e_2) &= \begin{cases} \{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \cup \widehat{\Delta}_{\alpha p}(e_2), & \text{if } E_\alpha(e_1) = 1 \\ \{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0 \end{cases} \\
&\subseteq \{e e_2 \mid e \in \mathrm{cl}(e_1)\} \cup \mathrm{cl}(e_2) \subseteq \mathrm{cl}(e_1 e_2) \\
\widehat{\Delta}_{\alpha p}(e_1^*) &= \{e e_1^* \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \subseteq \{e e_1^* \mid e \in \mathrm{cl}(e_1)\} \subseteq \mathrm{cl}(e_1^*).
\end{aligned}$$

For arbitrary $x \in (\mathsf{At} \cdot \mathsf{P})^*$, we proceed by induction on the length of x. The base case $x = \varepsilon$ is easy and the case $x = \alpha p$ is given by the previous argument. For $x \neq \varepsilon$ and $y \neq \varepsilon$,

$$\begin{aligned}
\widehat{\Delta}_{xy}(e) &= (\widehat{\Delta}_y \bullet \widehat{\Delta}_x)(e) = \bigcup \{\widehat{\Delta}_y(d) \mid d \in \widehat{\Delta}_x(e)\} \\
&\subseteq \bigcup \{\mathrm{cl}(d) \mid d \in \mathrm{cl}(e)\} = \mathrm{cl}(e).
\end{aligned}$$

□

Lemma 24.4 *For all* KAT *terms e and $x \in (\mathsf{At} \cdot \mathsf{P})^*$, $\widehat{D}_x(e) \approx \sum \widehat{\Delta}_x(e)$.*

Proof. We first show that for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$, $D_{\alpha p}(e) \approx \sum \widehat{\Delta}_{\alpha p}(e)$ by induction on the structure of e. For $q \in \mathsf{P}$, we have

$$D_{\alpha p}(q) = \begin{cases} 1, & \text{if } p = q \\ 0, & \text{if } p \neq q \end{cases} = \begin{cases} \sum\{1\}, & \text{if } p = q \\ \sum \varnothing, & \text{if } p \neq q \end{cases} = \sum \widehat{\Delta}_{\alpha p}(q).$$

For $b \in B$,

$$D_{\alpha p}(b) = 0 = \sum \varnothing = \sum \widehat{\Delta}_{\alpha p}(b).$$

For the other operators,

$$\begin{aligned}
D_{\alpha p}(e_1 + e_2) &= D_{\alpha p}(e_1) + D_{\alpha p}(e_2) \approx \sum \widehat{\Delta}_{\alpha p}(e_1) + \sum \widehat{\Delta}_{\alpha p}(e_2) \\
&\approx \sum (\widehat{\Delta}_{\alpha p}(e_1) \cup \widehat{\Delta}_{\alpha p}(e_2)) = \sum \widehat{\Delta}_{\alpha p}(e_1 + e_2),
\end{aligned}$$

$$\begin{aligned}
D_{\alpha p}(e_1 e_2) &= D_{\alpha p}(e_1) e_2 + E_\alpha(e_1) D_{\alpha p}(e_2) \\
&\approx (\sum \widehat{\Delta}_{\alpha p}(e_1)) e_2 + E_\alpha(e_1) \sum \widehat{\Delta}_{\alpha p}(e_2) \\
&\approx \begin{cases} \sum\{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} + \sum \widehat{\Delta}_{\alpha p}(e_2), & \text{if } E_\alpha(e_1) = 1 \\ \sum\{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0 \end{cases} \\
&\approx \begin{cases} \sum(\{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \cup \widehat{\Delta}_{\alpha p}(e_2)), & \text{if } E_\alpha(e_1) = 1 \\ \sum\{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0 \end{cases} \\
&= \sum \widehat{\Delta}_{\alpha p}(e_1 e_2),
\end{aligned}$$

$$\begin{aligned} D_{\alpha p}(e_1^*) &= D_{\alpha p}(e_1)e_1^* \approx (\sum \widehat{\Delta}_{\alpha p}(e_1))e_1^* \\ &\approx \sum\{ee_1^* \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} = \sum \widehat{\Delta}_{\alpha p}(e_1^*). \end{aligned}$$

Now we show the result for arbitrary $x \in (\mathsf{At} \cdot \mathsf{P})^*$ by induction on the length of x. The case $x = \varepsilon$ is trivial, and the case $x = \alpha p$ is given by the previous argument. Finally, for $x \neq \varepsilon$ and $y \neq \varepsilon$,

$$\begin{aligned} \widehat{D}_{xy}(e) &= \widehat{D}_y(\widehat{D}_x(e)) \\ &\approx \widehat{D}_y(\sum \widehat{\Delta}_x(e)) \qquad \text{by Lemma 24.1} \\ &= \sum\{\widehat{D}_y(d) \mid d \in \widehat{\Delta}_x(e)\} \\ &\approx \sum\{\sum \widehat{\Delta}_y(d) \mid d \in \widehat{\Delta}_x(e)\} \approx \sum\bigcup\{\widehat{\Delta}_y(d) \mid d \in \widehat{\Delta}_x(e)\} \\ &= \sum(\widehat{\Delta}_y \bullet \widehat{\Delta}_x)(e) = \sum \widehat{\Delta}_{xy}(e). \end{aligned}$$

□

Theorem 24.5 *The automaton* $\mathsf{Exp}_e/{\approx}$ *has at most* $2^{|e|+1}$ *accessible states.*

Proof. The accessible states of $\mathsf{Exp}_e/{\approx}$ are $\{\widehat{D}_x(e)/{\approx} \mid x \in (\mathsf{At} \cdot \mathsf{P})^*\}$, where $d/{\approx}$ is the congruence class of d modulo $\approx$. The stated bound follows from Lemmas 24.2, 24.3, and 24.4. □

24.3 Brzozowski Meets Kleene

It is possible to obtain $\mathsf{Exp}_e/{\approx}$ by a Kleene construction to obtain a nondeterministic AGS N_e with finitely many states, then apply the construction of §24.4 to obtain a deterministic automaton M_e with at most $2^{|e|+1}$ states. The automaton $\mathsf{Exp}_e/{\approx}$ is a homomorphic image of M_e. A version of Kleene's theorem for KAT terms and automata on guarded strings has been described previously in [72], but the current treatment parallels more closely Brzozowski's original treatment for ordinary regular expressions [19] and aligns with the general coalgebraic structure of [17, 18].

Define the nondeterministic automaton

$$N_e \stackrel{\text{def}}{=} (Q, \Delta, \text{START}, \text{ACCEPT}),$$

where the set of states Q is the disjoint union $\mathrm{cl}(e) + (\mathsf{At} \times \mathrm{cl}(e))$, the transition function Δ is that defined in §24.2, and the start and accept states are

$$\text{START} \stackrel{\text{def}}{=} \{e\} \qquad \text{ACCEPT} \stackrel{\text{def}}{=} \{(\alpha, d) \mid E_\alpha(d) = 1\}.$$

That Δ_α maps $\mathrm{cl}(e)$ to $2^{\mathsf{At} \times \mathrm{cl}(e)}$ is immediate from the definition of Δ_α, and that Δ_p maps $\mathsf{At} \times \mathrm{cl}(e)$ to $2^{\mathrm{cl}(e)}$ is guaranteed by (24.4) and Lemma 24.3.

Now let

$$M_e \stackrel{\text{def}}{=} (2^{\mathrm{cl}(e)}, \delta, \varepsilon, \text{START})$$

be the deterministic automaton obtained from N_e by the subset construction as described in §24.4. The start state of M_e is $\{e\}$, and δ and ε are given by

$$\delta_{\alpha p}(A) = \bigcup_{d \in A} \widehat{\Delta}_{\alpha p}(d) \qquad \varepsilon_\alpha(A) = \begin{cases} 1, & \text{if } \exists d \in A\ E_\alpha(d) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the accessible states are all of the form $A \subseteq \mathrm{cl}(e)$, thus by Lemma 24.2, M_e has at most $2^{|e|+1}$ accessible states.

Theorem 24.6 *For $A \subseteq \mathsf{Exp}$, the map $A \mapsto (\sum A)/\!\approx$ is a* KCT*-morphism. Ignoring inaccessible states, the quotient automaton $\mathsf{Exp}_e/\!\approx$ is the image of M_e under this map.*

Proof. We must show that the function $A \mapsto \sum A$ maps the start state of M_e to the start state of Exp_e, and that this function is a bisimulation modulo $\approx$. For δ,

$$\begin{aligned} \textstyle\sum \delta_{\alpha p}(A) &= \textstyle\sum \bigcup \{\widehat{\Delta}_{\alpha p}(d) \mid d \in A\} \\ &\approx \textstyle\sum \{\sum \widehat{\Delta}_{\alpha p}(d) \mid d \in A\} \\ &\approx \textstyle\sum \{D_{\alpha p}(d) \mid d \in A\} \qquad \text{by Lemma 24.4} \\ &= D_{\alpha p}(\textstyle\sum A), \end{aligned}$$

therefore

$$(\textstyle\sum \delta_{\alpha p}(A))/\!\approx \;=\; (D_{\alpha p}(\sum A))/\!\approx \;=\; D_{\alpha p}((\sum A)/\!\approx).$$

For ε,

$$\varepsilon_\alpha(A) = \begin{cases} 1, & \text{if } \exists d \in A\ E_\alpha(d) = 1 \\ 0, & \text{otherwise} \end{cases} = E_\alpha(\textstyle\sum A) = E_\alpha((\sum A)/\!\approx).$$

The map also preserves start states:

$$\{e\} \mapsto (\textstyle\sum \{e\})/\!\approx \;=\; e/\!\approx.$$

Thus the map $A \mapsto (\sum A)/\!\approx$ is a KCT-morphism mapping M_e to $\mathsf{Exp}_e/\!\approx$. □

24.4 Automatic Proof Generation in PSPACE

The results of Sections 24.2 and 24.3 give rise to a nondeterministic linear-space algorithm for deciding the equivalence of two given KAT terms. By Savitch's theorem [109], there is a deterministic quadratic-space algorithm. The deterministic algorithm can be used to create bisimulation proofs of equivalence or inequivalence automatically.

To obtain the linear space bound, we first show that each element of $\mathrm{cl}(e)$ corresponds to an occurrence of a subterm of e. This lets us use the occurrences of subterms of e as representatives for the elements of $\mathrm{cl}(e)$. To define the correspondence, we view terms as labeled trees; that is, as partial functions

$$e : \omega^* \to \mathsf{P} \cup \mathsf{B} \cup \{+, \cdot, {}^*, {}^-, 0, 1\}$$

with domain of definition $\mathrm{dom}\, e \subseteq \omega^*$ such that

- $\mathrm{dom}\, e$ is finite, nonempty, and prefix-closed;
- if $\sigma \in \mathrm{dom}\, e$ and $e(\sigma)$ is of arity n, then $\sigma i \in \mathrm{dom}\, e$ iff $i < n$. The arities of elements of P and B are 0 and those of $+, \cdot, {}^*, {}^-, 0, 1$ are 2, 2, 1, 1, 0, 0, respectively.

An occurrence of a subterm of e is identified by its position $\sigma \in \mathrm{dom}\, e$. The subterm at position σ is $\lambda\tau.e(\sigma\tau)$, and its domain is $\{\tau \mid \sigma\tau \in \mathrm{dom}\, e\}$.

Define a partial function $R : \omega^* \times \mathsf{Exp} \to \mathsf{Exp}$ inductively by

$$\begin{aligned}
R(0\sigma, e_1 + e_2) &\stackrel{\text{def}}{=} R(\sigma, e_1) & \qquad R(0\sigma, e_1 e_2) &\stackrel{\text{def}}{=} R(\sigma, e_1) \cdot e_2 \\
R(1\sigma, e_1 + e_2) &\stackrel{\text{def}}{=} R(\sigma, e_2) & \qquad R(1\sigma, e_1 e_2) &\stackrel{\text{def}}{=} R(\sigma, e_2) \\
R(0\sigma, e^*) &\stackrel{\text{def}}{=} R(\sigma, e) \cdot e^* & \qquad R(\varepsilon, e) &\stackrel{\text{def}}{=} e.
\end{aligned}$$

One can show by induction that $R(\sigma, e)$ is defined iff $\sigma \in \mathrm{dom}\, e$, and that a term is in $\mathrm{cl}(e)$ iff it is either 1 or $R(\sigma, e)$ for some $\sigma \in \mathrm{dom}\, e$.

Now we show how to construct coinductive equivalence and inequivalence proofs for two given terms e_1 and e_2. Construct the two nondeterministic AGS N_{e_1} and N_{e_2} as described in §24.3, representing the states by $\mathrm{dom}\, e_1$ and $\mathrm{dom}\, e_2$, respectively (assume without loss of generality that $1 = R(\sigma, e_1) = R(\tau, e_2)$ for some σ and τ). If we like, we can also reduce terms modulo $\approx$, so that if $R(\sigma, e_1) \approx R(\tau, e_1)$, we only need one of σ, τ.

Place pebbles on the start states of the two automata. Nondeterministically guess a string $y \in (\mathsf{At} \cdot \mathsf{P})^*$ and move the pebbles to all accessible states according to the transition functions of the two machines. Halt and declare e_1 and e_2 inequivalent if there exists $\alpha \in \mathsf{At}$ such that

$$E_\alpha\Big(\sum_{\tau \in A} R(\tau, e_1)\Big) \neq E_\alpha\Big(\sum_{\rho \in B} R(\rho, e_2)\Big),$$

where A and B are the sets of states of N_{e_1} and N_{e_2}, respectively, currently occupied by pebbles; we have found a guarded string $x = y\alpha$ accepted by one but not by the other, since

$$\begin{aligned} Ł(e_1)(x) &= E_\alpha(\widehat{D}_y(e_1)) = E_\alpha(\sum_{\tau\in A} R(\tau, e_1)) \\ Ł(e_2)(x) &= E_\alpha(\widehat{D}_y(e_2)) = E_\alpha(\sum_{\rho\in B} R(\rho, e_2)), \end{aligned}$$

therefore $Ł(e_1)(x) \neq Ł(e_2)(x)$.

Once we can decide equivalence in quadratic space, we can produce a bisimulation proof of equivalence in the same amount of space. We first produce the deterministic automata M_{e_1} and M_{e_2} equivalent to N_{e_1} and N_{e_2}. The states of M_{e_1} and M_{e_2} are represented by the powersets of $\operatorname{dom} e_1$ and $\operatorname{dom} e_2$, respectively. These sets are of exponential size, but they can be generated sequentially in linear space. The transition function is the action on subsets as defined in §24.4, and this can also be generated in linear space.

Now we attempt to construct the maximal bisimulation between the two deterministic automata. We iterate through all pairs of states, testing equivalence of each pair as described above. If the states are equivalent, we output the pair as bisimilar. The set of pairs that are ever output is the maximal bisimulation.

In case e_1 and e_2 are not equivalent, a witness for inequivalence can also be produced in PSPACE. A witness for inequivalence is a guarded string x accepted by one automaton but not the other. The shortest such string can be exponentially long in the worst case, but can be produced in the same way that one would produce an exponential-length accepting computation of a nondeterministic linear-space Turing machine, by a straightforward modification of the proof of Savitch's theorem [109].

Supplementary Lecture A

Propositional Dynamic Logic

Propositional Dynamic Logic (PDL) was first defined by Fischer and Ladner [35, 36]. It plays the same role in Dynamic Logic that classical propositional logic plays in classical predicate logic. It describes the properties of the interaction between programs and propositions that are independent of the domain of computation.

A.5 Basic Definitions

Syntax

Syntactically, PDL is a blend of propositional logic, modal logic, and the algebra of regular events. It has expressions of two sorts: *programs* $p, q, r, \ldots$ and *propositions* or *formulas* $\varphi, \psi, \ldots$. There are countably many atomic symbols of each sort, as well as a variety of operators for forming compound expressions from simpler ones:

- propositional operators $\vee$, $\neg$
- program operators $\cup$, ;, *
- mixed operators ?, < >

Compound programs and propositions are defined by mutual induction, as follows. If φ, ψ are propositions and p, q are programs, then

$\varphi \vee \psi$ (propositional disjunction)
$\neg\varphi$ (propositional negation)
$<p>\varphi$ (modal possibility)

are propositions and

$p;q$ (sequential composition)
$p \cup q$ (nondeterministic choice)
p^* (iteration)
$\varphi?$ (test)

are programs. The intuitive meanings of the less familiar of these constructs are as follows:

$$\begin{aligned} <p>\varphi &= \text{“It is possible to execute } p \text{ and terminate in a state satisfying } \varphi\text{.”} \\ p;q &= \text{“Execute } p\text{, then execute } q\text{.”} \\ p \cup q &= \text{“Choose either } p \text{ or } q \text{ nondeterministically and execute it.”} \\ p^* &= \text{“Execute } p \text{ repeatedly a nondeterministically chosen finite number of times.”} \\ \varphi? &= \text{“Test } \varphi\text{; proceed if true, fail if false.”} \end{aligned}$$

The set of propositions is denoted Φ and the set of programs is denoted Ψ. We avoid parentheses by assigning precedence to the operators: unary operators, including $<p>$, bind tighter than binary ones, and ; binds tighter than $\cup$. Also, under the semantics to be given in the next section, the operators ; and $\cup$ will turn out to be associative, so we may write $p;q;r$ and $p \cup q \cup r$ without ambiguity.

The primitive operators are chosen for their mathematical simplicity. A number of more conventional programming constructs can be defined from them. The propositional operators $\wedge$, $\Rightarrow$, $\Leftrightarrow$, `0`, and `1` are defined from $\vee$ and $\neg$ in the usual way. In addition:

$$\begin{aligned} \texttt{skip} &= \texttt{1?} \\ \texttt{fail} &= \texttt{0?} \\ [p]\varphi &= \neg<p>\neg\varphi \\ \textbf{if } \varphi_1 \Rightarrow p_1 \mid \cdots \mid \varphi_n \Rightarrow p_n \textbf{ fi} &= \varphi_1?;p_1 \cup \cdots \cup \varphi_n?;p_n \\ \textbf{do } \varphi_1 \Rightarrow p_1 \mid \cdots \mid \varphi_n \Rightarrow p_n \textbf{ od} &= (\varphi_1?;p_1 \cup \cdots \cup \varphi_n?;p_n)^*;(\neg\varphi_1 \wedge \cdots \wedge \neg\varphi_n)? \\ \texttt{if } \varphi \texttt{ then } p \texttt{ else } q &= \textbf{if } \varphi \Rightarrow p \mid \neg\varphi \Rightarrow q \textbf{ fi} = \varphi?;p \cup \neg\varphi?;q \\ \texttt{while } \varphi \texttt{ do } p &= \textbf{do } \varphi \Rightarrow p \textbf{ od} = (\varphi?;p)^*;\neg\varphi? \\ \textbf{repeat } p \textbf{ until } \varphi &= p;\texttt{while } \neg\varphi \texttt{ do } p = p;(\neg\varphi?;p)^*;\varphi? \\ \{\varphi\}p\{\psi\} &= \varphi \Rightarrow [p]\psi \end{aligned}$$

The propositions $<p>\varphi$ and $[p]\varphi$ are read "diamond p φ" and "box p φ", respectively. The latter has the intuitive meaning, "Whenever p terminates, it must do so in a state satisfying φ." Unlike $<p>\varphi$, $[p]\varphi$ does not imply that p terminates. Indeed, the formula $[p]0$ asserts that no computation of p terminates. For fixed program p, the operator $<p>$ behaves like a possibility operator of modal logic (see [52, 20]). The operator $[p]$ is the modal dual of $<p>$ and behaves like a modal necessity operator.

The ternary **if-then-else** operator and the binary **while-do** operator are the usual *conditional test* and *while loop* constructs found in conventional programming languages. The constructs **if-**|**-fi** and **do-**|**-od** are the *alternative guarded command* and *iterative guarded command* constructs, respectively (see [42]). The construct $\{\varphi\}p\{\psi\}$ is the classical Hoare partial correctness assertion [49].

Semantics

The formal semantics of PDL comes from modal logic. A *Kripke model* is a pair $\mathcal{M} = (S^{\mathcal{M}}, I^{\mathcal{M}})$ where $S^{\mathcal{M}} = \{u, v, \ldots\}$ is an abstract set of *states* and $I^{\mathcal{M}}$ is an *interpretation function*. Each proposition φ is interpreted as a subset $\varphi^{\mathcal{M}} \subseteq S^{\mathcal{M}}$, and each program p is interpreted as binary relation $p^{\mathcal{M}}$ on $S^{\mathcal{M}}$. We may think of $\varphi^{\mathcal{M}}$ as the set of states satisfying the proposition φ, and $p^{\mathcal{M}}$ as the input/output relation of the program p.

The function $I^{\mathcal{M}}$ assigns an arbitrary subset of $S^{\mathcal{M}}$ to each atomic proposition symbol and an arbitrary binary relation on $S^{\mathcal{M}}$ to each atomic program symbol. Compound programs and propositions receive their meanings inductively:

$$\begin{aligned}
(\varphi \vee \psi)^{\mathcal{M}} &= \varphi^{\mathcal{M}} \cup \psi^{\mathcal{M}} \\
(\neg\varphi)^{\mathcal{M}} &= S^{\mathcal{M}} - \varphi^{\mathcal{M}} \\
(<p>\varphi)^{\mathcal{M}} &= p^{\mathcal{M}} \circ \varphi^{\mathcal{M}} = \{u \mid \exists v\ (u,v) \in p^{\mathcal{M}} \text{ and } v \in \varphi^{\mathcal{M}}\} \\
(p;q)^{\mathcal{M}} &= p^{\mathcal{M}} \circ q^{\mathcal{M}} = \{(u,v) \mid \exists w\ (u,w) \in p^{\mathcal{M}} \text{ and } (w,v) \in q^{\mathcal{M}}\} \\
(p \cup q)^{\mathcal{M}} &= p^{\mathcal{M}} \cup q^{\mathcal{M}} \\
(p^*)^{\mathcal{M}} &= \text{the reflexive transitive closure of } p^{\mathcal{M}} \\
(\varphi?)^{\mathcal{M}} &= \{(u,u) \mid u \in \varphi^{\mathcal{M}}\}
\end{aligned}$$

where the reflexive-transitive closure of a binary relation ρ is

$$\bigcup_{n<\omega} \rho^n\ ,$$

where

$$\begin{aligned}
\rho^0 &= \{(u,u) \mid u \in S^{\mathcal{M}}\} \\
\rho^{n+1} &= \rho \circ \rho^n\ .
\end{aligned}$$

The symbol $\circ$ denotes relational composition.

The defined operators inherit their meanings from these definitions:

$$\begin{aligned}
(\varphi \wedge \psi)^{\mathcal{M}} &= \varphi^{\mathcal{M}} \cap \psi^{\mathcal{M}} \\
([p]\varphi)^{\mathcal{M}} &= \{u \mid \forall v\ (u,v) \in p^{\mathcal{M}} \Rightarrow v \in \varphi^{\mathcal{M}}\} \\
\mathtt{1}^{\mathcal{M}} &= S^{\mathcal{M}} \\
\mathtt{0}^{\mathcal{M}} &= \varnothing \\
\mathtt{skip}^{\mathcal{M}} &= \{(u,u) \mid u \in S^{\mathcal{M}}\} \\
\mathtt{fail}^{\mathcal{M}} &= \varnothing
\end{aligned}$$

The **if-then-else**, **while-do**, and guarded commands also receive meanings from the above definitions.

It can be argued that the input/output relations given by these definitions capture the intuitive operational meanings of these constructs. For example, the relation associated with the program `while` φ `do` p is the set of pairs (u,v) for which there exist states $u_0, u_1, \ldots, u_n$, $n \geq 0$, such that $u = u_0$, $v = u_n$, $u_i \in \varphi^{\mathcal{M}}$ and $(u_i, u_{i+1}) \in p^{\mathcal{M}}$ for $0 \leq i < n$, and $u_n \in \neg\varphi^{\mathcal{M}}$.

We often write $\mathcal{M}, u \models \varphi$ for $u \in \varphi^{\mathcal{M}}$ and say that u *satisfies* φ *in* $\mathcal{M}$. We may write $u \models \varphi$ when $\mathcal{M}$ is understood. We write $\mathcal{M} \models \varphi$ if $\mathcal{M}, u \models \varphi$ for all $u \in \mathcal{M}$, and we write $\models \varphi$ and say that φ is *valid* if $\mathcal{M} \models \varphi$ for all $\mathcal{M}$. We say that φ is *satisfiable* if $\mathcal{M}, u \models \varphi$ for some $\mathcal{M}, u$. If Σ is a set of propositions, we write $\mathcal{M} \models \Sigma$ if $\mathcal{M} \models \varphi$ for all $\varphi \in \Sigma$. A proposition ψ is said to be a *logical consequence* of Σ if for all $\mathcal{M}$, $\mathcal{M} \models \psi$ whenever $\mathcal{M} \models \Sigma$, in which case we write $\Sigma \models \psi$. (Note that this is *not* the same as saying that $\mathcal{M}, u \models \psi$ whenever $\mathcal{M}, u \models \Sigma$.) We say that an inference rule

$$\frac{\Sigma}{\psi}$$

is *sound* if ψ is a logical consequence of Σ.

This version of PDL is usually called *regular* PDL because of the primitive operators $\cup, ;, *$, which are familiar from the algebra of regular events (see [50]). Programs can be viewed as regular expressions over the atomic programs and tests. In fact, it can be shown that if φ is an atomic proposition symbol, then any two test-free programs p, q are equivalent as regular expressions if and only if the formula $\langle p\rangle\varphi \iff \langle q\rangle\varphi$ is valid.

Example A.1 *Let φ be an atomic proposition, let p be an atomic program, and let $\mathcal{M} = (S^{\mathcal{M}}, I^{\mathcal{M}})$ be a Kripke model with*

$$\begin{aligned}
S^{\mathcal{M}} &= \{u, v, w\} \\
\varphi^{\mathcal{M}} &= \{u, v\} \\
p^{\mathcal{M}} &= \{(u,v), (u,w), (v,w), (w,v)\}
\end{aligned}$$

Then $u \models \langle p\rangle\neg\varphi \wedge \langle p\rangle\varphi$, *but* $v \models [p]\neg\varphi$ *and* $w \models [p]\varphi$. *Moreover,*

$$\mathcal{M} \models \langle p^*\rangle[(p;p)^*]\varphi \wedge \langle p^*\rangle[(p;p)^*]\neg\varphi\ .$$

Other semantics besides Kripke semantics have been studied [9, 93, 59, 116, 60, 99].

Computation Sequences

Let p be a program. A *finite computation sequence* of p is a finite-length string of atomic programs and tests representing a possible sequence of atomic steps that may occur in a halting execution of p. The set of all such sequences is denoted $\mathsf{CS}(p)$. We use the word "possible" loosely—$\mathsf{CS}(p)$ is determined by the syntax of p alone, and may contain strings that are never executed in any interpretation.

Formally, the set $\mathsf{CS}(p)$ is defined by induction on syntax:

$$\begin{aligned}
\mathsf{CS}(p) &= \{p\},\ p \text{ an atomic program or test}\\
\mathsf{CS}(p;q) &= \{\alpha;\beta \mid \alpha \in \mathsf{CS}(p),\ \beta \in \mathsf{CS}(q)\}\\
\mathsf{CS}(p \cup q) &= \mathsf{CS}(p) \cup \mathsf{CS}(q)\\
\mathsf{CS}(p^*) &= \bigcup_{n\geq 0} \mathsf{CS}(p^n)
\end{aligned}$$

where $p^0 = \mathtt{skip}$ and $p^{n+1} = p;p^n$. For example, if p is an atomic program and φ is an atomic formula, then the program

$$\mathtt{while}\ \varphi\ \mathtt{do}\ p \quad = \quad (\varphi?;p)^*;\neg\varphi?$$

has as computation sequences all strings of the form

$$\varphi?;p;\varphi?;p;\cdots;\varphi?;p;\mathtt{skip};\neg\varphi?\ .$$

Note that each finite computation sequence q of a program p is itself a program, and

$$\mathsf{CS}(q) = \{q\}\ .$$

Moreover, the following proposition is not difficult to prove by induction on syntax:

Proposition A.2 $p^{\mathcal{M}} = \bigcup_{q\in\mathsf{CS}(p)} q^{\mathcal{M}}$.

A.6 A Deductive System for PDL

The following Hilbert-style axiom system for PDL was formulated by Segerberg [110].

Axioms of PDL

1. *axioms for propositional logic*
2. $\langle p\rangle\varphi \wedge [p]\psi \Rightarrow \langle p\rangle(\varphi \wedge \psi)$
3. $\langle p\rangle(\varphi \vee \psi) \iff \langle p\rangle\varphi \vee \langle p\rangle\psi$
4. $\langle p \cup q\rangle\varphi \iff \langle p\rangle\varphi \vee \langle q\rangle\varphi$
5. $\langle p; q\rangle\varphi \iff \langle p\rangle\langle q\rangle\varphi$
6. $\langle\psi?\rangle\varphi \iff \psi \wedge \varphi$
7. $(\varphi \vee \langle p\rangle\langle p^*\rangle\varphi) \Rightarrow \langle p^*\rangle\varphi$
8. $\langle p^*\rangle\varphi \Rightarrow (\varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi))$

Axioms 2 and 3 are not particular to PDL, but hold in all normal modal systems (see [52, 20]). Axiom 8 is called the PDL *induction axiom*, and is better known in its dual form (Theorem A.3(8) below).

Rules of Inference

1. *modus ponens:*
$$\frac{\varphi,\ \varphi \Rightarrow \psi}{\psi}$$
2. *modal generalization:*
$$\frac{\varphi}{[p]\varphi}$$

We write $\vdash \varphi$ if the proposition φ is a theorem of this system, and say that φ is *consistent* if not $\vdash \neg\varphi$. A set Σ of propositions is *consistent* if all finite conjunctions of elements of Σ are consistent.

The soundness of these axioms and rules over the Kripke semantics can be established by elementary arguments in relational algebra.

A.7 Basic Properties

We list some basic theorems and derived rules of PDL that are provable in the deductive system of the previous section.

Theorem A.3 *The following are theorems of* PDL*:*

1. *all propositional tautologies*

2. $[p](\varphi \Rightarrow \psi) \Rightarrow ([p]\varphi \Rightarrow [p]\psi)$
3. $[p](\varphi \wedge \psi) \iff [p]\varphi \wedge [p]\psi$
4. $[p \cup q]\varphi \iff [p]\varphi \wedge [q]\varphi$
5. $[p;q]\varphi \iff [p][q]\varphi$
6. $[\varphi?]\psi \iff (\varphi \Rightarrow \psi)$
7. $[p^*]\varphi \Rightarrow \varphi \wedge [p][p^*]\varphi$
8. $(\varphi \wedge [p^*](\varphi \Rightarrow [p]\varphi)) \Rightarrow [p^*]\varphi$
9. $\langle p\rangle(\varphi \wedge \psi) \Rightarrow \langle p\rangle\varphi \wedge \langle p\rangle\psi$
10. $[p](\varphi \vee \psi) \leftarrow [p]\varphi \vee [p]\psi$
11. $\langle p^*; p^*\rangle\varphi \iff \langle p^*\rangle\varphi$
12. $\langle p^{**}\rangle\varphi \iff \langle p^*\rangle\varphi$
13. $(\varphi \vee \langle p\rangle\langle p^*\rangle\varphi) \iff \langle p^*\rangle\varphi$
14. $\langle p^*\rangle\varphi \iff (\varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi))$
15. $(\varphi \wedge [p^*](\varphi \Rightarrow [p]\varphi)) \iff [p^*]\varphi$

Theorem A.4 *The following are sound rules of inference of* PDL*:*

1. *monotonicity of* $\langle p\rangle$*:*

$$\frac{\varphi \Rightarrow \psi}{\langle p\rangle\varphi \Rightarrow \langle p\rangle\psi}$$

2. *monotonicity of* $[p]$*:*

$$\frac{\varphi \Rightarrow \psi}{[p]\varphi \Rightarrow [p]\psi}$$

3. *reflexive-transitive closure:*

$$\frac{(\varphi \vee \langle p\rangle\psi) \Rightarrow \psi}{\langle p^*\rangle\varphi \Rightarrow \psi}$$

4. *loop invariance rule:*

$$\frac{\psi \Rightarrow [p]\psi}{\psi \Rightarrow [p^*]\psi}$$

5. *Hoare composition rule:*

$$\frac{\{\varphi\}p\{\sigma\},\ \{\sigma\}q\{\psi\}}{\{\varphi\}p;q\{\psi\}}$$

6. *Hoare conditional rule:*

$$\frac{\{\varphi \wedge \sigma\}p\{\psi\},\ \{\neg\varphi \wedge \sigma\}q\{\psi\}}{\{\sigma\}\textbf{if } \varphi \textbf{ then } p \textbf{ else } q \textbf{ fi}\{\psi\}}$$

7. *Hoare* **while** *rule:*

$$\frac{\{\varphi \wedge \psi\}p\{\psi\}}{\{\psi\}\textbf{while } \varphi \textbf{ do } p \textbf{ od}\{\neg\varphi \wedge \psi\}}$$

The properties of Theorem A.3(2-8) are the modal duals of Axioms 2-8. The converses of Theorem A.3(9,10) are *not* valid. They are violated in state u of the model of Example A.1. The rules of Theorem A.4(1,2) say that the constructs $\langle p\rangle\varphi$ and $[p]\varphi$ are *monotone* in φ with respect to the ordering of logical implication. These constructs are also monotone and antitone in p, respectively, as asserted by the following metatheorem:

Proposition A.5 *If $p^{\mathcal{M}} \subseteq q^{\mathcal{M}}$, then for all φ,*

1. $\mathcal{M} \models \langle p\rangle\varphi \Rightarrow \langle q\rangle\varphi$
2. $\mathcal{M} \models [q]\varphi \Rightarrow [p]\varphi$.

These follow from Axiom 4 and Theorem A.3(4).

The * Operator, Induction, and Reflexive-Transitive Closure

The iteration operator * is interpreted as the reflexive-transitive closure operator on binary relations. It is the means by which looping is coded in PDL. Looping introduces a level of complexity to PDL beyond the other operators. Because of it, PDL is not compact: the set

$$\{\langle p^*\rangle\varphi\} \cup \{\neg\varphi, \neg\langle p\rangle\varphi, \neg\langle p^2\rangle\varphi, \ldots\} \tag{A.1}$$

is finitely satisfiable but not satisfiable. This seems to say that looping is inherently infinitary; it is thus rather surprising that there should be a finitary complete axiomatization.

The dual propositions Axiom 8 and Theorem A.3(8) are jointly called the PDL *induction axiom*. Together with Axiom 7, they completely axiomatize the behavior of *. Intuitively, the induction axiom in the form of Theorem A.3(8) says: if φ is true initially, and if the truth of φ is preserved by the program p, then φ will be true after any number of iterations of p. It is very similar to the induction axiom of Peano arithmetic:

$$\varphi(0) \wedge \forall n\ (\varphi(n) \Rightarrow \varphi(n+1)) \quad \Rightarrow \quad \forall n\ \varphi(n)\ .$$

Here $\varphi(0)$ is the basis of the induction and $\forall n\ (\varphi(n) \Rightarrow \varphi(n+1))$ is the induction step, from which the conclusion $\forall n\ \varphi(n)$ may be drawn. In the PDL induction axiom, the basis is φ and the induction step is $[p^*](\varphi \Rightarrow [p]\varphi)$, from which the conclusion $[p^*]\varphi$ may be drawn.

The induction axiom is closely related to the reflexive-transitive closure rule (Theorem A.4(3)). The significance of this rule is best described in terms of its relationship to Axiom 7. This axiom is obtained by substituting $<p^*>\varphi$ for ψ in the premise of the reflexive-transitive closure rule. Axiom 7 thus says that $<p^*>\varphi$ is a solution of

$$(\varphi \vee <p>X) \Rightarrow X\ ; \tag{A.2}$$

the reflexive-transitive closure rule says that it is the *least* solution to (A.2) (with respect to logical implication) among all PDL propositions.

The relationship between the induction axiom (Axiom 8), the reflexive-transitive closure rule (Theorem A.4(3)), and the rule of loop invariance (Theorem A.4(4)) is summed up in the following proposition. We emphasize that this result is purely proof-theoretic and is independent of the semantics of §A.5.

Proposition A.6 *In* PDL *without the induction axiom, the following axioms and rules are interderivable:*

(i) *the induction axiom (Axiom 8);*

(ii) *the loop invariance rule (Theorem A.4(4));*

(iii) *the reflexive-transitive closure rule (Theorem A.4(3)).*

Proof. First we show that the monotonicity rule (Theorem A.4(2)) is derivable in PDL without induction. Assuming the premise $\varphi \Rightarrow \psi$ and applying modal generalization, we obtain $[p](\varphi \Rightarrow \psi)$; the conclusion $[p]\varphi \Rightarrow [p]\psi$ then follows from Theorem A.3(2) and modus ponens. The dual monotonicity rule (Theorem A.4(1)) can be derived from this rule by pure propositional reasoning.

(i) $\Rightarrow$ (ii): Assume the premise of (ii):

$$\varphi \;\Rightarrow\; [p]\varphi\,.$$

By modal generalization,

$$[p^*](\varphi \Rightarrow [p]\varphi)\,,$$

thus

$$\begin{aligned}\varphi &\;\Rightarrow\; \varphi \wedge [p^*](\varphi \Rightarrow [p]\varphi)\\ &\;\Rightarrow\; [p^*]\varphi\,.\end{aligned}$$

The first implication is by propositional reasoning, and the second is the box form of the induction axiom (Theorem A.3(8)). By transitivity of implication, we obtain

$$\varphi \;\Rightarrow\; [p^*]\varphi\,,$$

which is the conclusion of (ii).

(ii) $\Rightarrow$ (iii): Dualizing the rule (iii) by purely propositional reasoning, we obtain a rule

$$\frac{\psi \;\Rightarrow\; \varphi \wedge [p]\psi}{\psi \;\Rightarrow\; [p^*]\varphi} \tag{A.3}$$

equipollent with (iii). It thus suffices to derive (A.3) from (ii). From the premise of (A.3), we obtain by propositional reasoning the two formulas

$$\psi \;\Rightarrow\; \varphi \tag{A.4}$$

$$\psi \;\Rightarrow\; [p]\psi\,. \tag{A.5}$$

Applying (ii) to (A.5), we obtain

$$\psi \;\Rightarrow\; [p^*]\psi\,,$$

which by (A.4) and monotonicity gives

$$\psi \;\Rightarrow\; [p^*]\varphi\,.$$

This is the conclusion of (A.3).

(iii) $\Rightarrow$ (i): By Axiom 3, propositional reasoning, and Axiom 7, we have

$$\begin{aligned}&\varphi \vee \langle p\rangle(\varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi))\\ &\quad\Rightarrow\; \varphi \vee \langle p\rangle\varphi \vee \langle p\rangle\langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi)\\ &\quad\Rightarrow\; \varphi \vee (\neg\varphi \wedge \langle p\rangle\varphi) \vee \langle p\rangle\langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi)\\ &\quad\Rightarrow\; \varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi)\,.\end{aligned}$$

By transitivity of implication,

$$\varphi \vee \langle p \rangle(\varphi \vee \langle p^* \rangle(\neg\varphi \wedge \langle p \rangle\varphi)) \quad \Rightarrow \quad \varphi \vee \langle p^* \rangle(\neg\varphi \wedge \langle p \rangle\varphi) \,.$$

Applying (iii), we obtain the induction axiom:

$$\langle p^* \rangle\varphi \quad \Rightarrow \quad \varphi \vee \langle p^* \rangle(\neg\varphi \wedge \langle p \rangle\varphi) \,.$$

□

Encoding of Hoare Logic

Dynamic Logic subsumes Hoare Logic. As an illustration, we show how to derive the Hoare `while` rule (Theorem A.4(7)) in PDL. The other Hoare rules are also derivable.

Assume that the premise

$$\{\varphi \wedge \psi\}p\{\psi\} \quad = \quad (\varphi \wedge \psi) \Rightarrow [p]\psi \tag{A.6}$$

holds. We wish to derive the conclusion

$$\{\psi\}\texttt{while}\ \varphi\ \texttt{do}\ p\{\neg\varphi \wedge \psi\} \quad = \quad \psi \Rightarrow [(\varphi?; p)^*; \neg\varphi?](\neg\varphi \wedge \psi) \,. \tag{A.7}$$

Using Theorem A.3(5,6) and propositional reasoning, the right-hand-side of (A.6) is equivalent to

$$\psi \quad \Rightarrow \quad [\varphi?; p]\psi \,.$$

Applying the loop invariance rule (Theorem A.4(4)), we obtain

$$\psi \quad \Rightarrow \quad [(\varphi?; p)^*]\psi \,.$$

By the monotonicity of $[(\varphi?; p)^*]$ (Theorem A.4(2)) and propositional reasoning,

$$\psi \quad \Rightarrow \quad [(\varphi?; p)^*](\neg\varphi \Rightarrow \neg\varphi \wedge \psi) \,.$$

Again by Theorem A.3(5,6) and propositional reasoning, we obtain the right-hand-side of (A.7).

Supplementary Lecture B

?

Conway gives a construction that embeds every *-continuous Kleene algebra in an S-algebra [29, Theorem 1, p. 102]. This construction can be described as a completion by *-ideals. We show that this construction extends to a functor from the category KA^* of *-continuous Kleene algebras and Kleene algebra morphisms to the category SA of S-algebras and continuous semiring morphisms, and that this functor is a left adjoint for the forgetful functor that assigns to each S-algebra its Kleene algebra structure. Moreover, this adjunction factors into two adjunctions, one relating KA^* and the category CS of closed semirings and ω-continuous semiring morphisms, and one relating CS and SA. The functor from KA^* to CS can be described as a completion by countably generated *-ideals.

B Definitions

Examples of Kleene Algebras

An important example of a Kleene algebra is $\mathsf{Reg}\,\Sigma$, the family of regular sets over a finite alphabet Σ. The equational theory of this structure is called the *algebra of regular events*. This theory was axiomatized by Salomaa [107], but his axiomatization involved rules that were not sound in general when interpreted over other Kleene algebras. It was shown in

[57] that the Kleene algebra axioms completely axiomatize the algebra of regular events. In other words, $\mathsf{Reg}\,\Sigma$ is the free Kleene algebra on free generators Σ. Conway [29, Theorem 5, p. 108] claimed without proof that the axioms for right-handed Kleene algebra completely axiomatize the algebra of regular events, but to his knowledge [30] and the author's, no proof has ever appeared in the literature. A weaker completeness proof involving (**??**) instead of (**??**–**??**) was previously given in [58].

Examples of Closed Semirings

B.1 Conway's Algebras

According to the definition in [29], an S-algebra

$$(SS, \textstyle\sum, \cdot, 0, 1)$$

is similar to a closed semiring, except that $\sum$ is defined not on sequences but on multisets of elements of SS; moreover, there is no cardinality restriction on the multiset (thus $\sum$ is too big to be represented in ZF set theory!). However, as with closed semirings, the value that $\sum$ takes on a given multiset is independent of the multiplicity of the elements, so $\sum$ might as well be defined on subsets of SS instead of multisets. So defined, $\sum A$ gives the supremum of A with respect to the order $\leq$. (We assume the axiom $\sum\{a\} = a$, which is omitted in [29].)

Thus, the only essential difference between S-algebras and closed semirings is that closed semirings are only required to contain suprema of countable sets, whereas S-algebras must contain suprema of all sets. Thus every S-algebra is automatically a closed semiring and every continuous semiring morphism (semiring morphism preserving all suprema) is automatically ω-continuous, and these notions coincide on countable algebras. This gives a forgetful functor $\mathsf{G} : \mathsf{SA} \to \mathsf{CS}$ from the category SA of S-algebras and continuous semiring morphisms to CS.

B A Kleene algebra that is not *-continuous

Recall the definitions of the categories

category	*objects*	*morphisms*
SA	S-algebras	continuous semiring morphisms
CS	closed semirings	ω-continuous semiring morphisms
KA*	*-continuous Kleene algebras	Kleene algebra morphisms
KA	Kleene algebras	Kleene algebra morphisms
RA	R-algebras	maps preserving $+, \cdot, ^*, 0, 1$

B The universal Horn theory of regular events

The *universal Horn theory* of a class of structures is the set of universally quantified equational implications of the form

$$\alpha_1 = \beta_1 \wedge \cdots \wedge \alpha_n = \beta_n \quad \Rightarrow \quad \alpha = \beta$$

true in all structures in the class. In this section, we give a brief argument showing that the axioms for Kleene algebra given in §B, which characterize the universal Horn theory of Kleene algebras and the equational theory of $\mathsf{Reg}\,\Sigma$, do not characterize the universal Horn theory of $\mathsf{Reg}\,\Sigma$. This refutes a conjecture of Conway [29, p. 103].

Consider the equational implication

$$a^2 = 1 \quad \Rightarrow \quad a = 1\,.$$

This formula is true in $\mathsf{Reg}\,\Sigma$ for all values of a; in other words, for any regular event A, if $A^2 = \{\epsilon\}$ then $A = \{\epsilon\}$. However, this formula is not a theorem of Kleene algebra, since there exists a Kleene algebra in which it fails: interpret a as the matrix

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

in the Kleene algebra of 2×2 matrices over $\mathsf{Reg}\,\Sigma$. This algebra was shown to be a Kleene algebra in [57].

It is an open problem to give a complete axiomatization of the universal Horn theory of the regular events.

B Right-handed Kleene algebras are not necessarily left-handed

Recall the right- and left-handed Kleene algebra rules

$$ax \le x \quad \Rightarrow \quad a^* x \le x \tag{B.1}$$
$$xa \le x \quad \Rightarrow \quad xa^* \le x\,. \tag{B.2}$$

In [101], Pratt asks whether these two rules are equivalent, and conjectures that they are not. His conjecture actually refers to the more specific case of Boolean monoids, in which $\le$ comes from a Boolean algebra. However, the question makes sense in the more general context of Kleene algebras. In this section we construct a right-handed Kleene algebra that is not left handed; this answers Pratt's question in the more general case of Kleene algebras and provides evidence in favor of his conjecture. The conjecture as stated in [101] is still open.

Consider the set $\mathcal{F}$ of strict, finitely additive set functions $f : 2^X \to 2^X$ for any infinite set X; that is, functions f for which

$$\begin{aligned} f(A \cup B) &= f(A) \cup f(B) \\ f(\varnothing) &= \varnothing\,. \end{aligned}$$

Define

$$\begin{aligned} (f \cup g)(A) &= f(A) \cup g(A) \\ (f \circ g)(A) &= f(g(A)) \\ f^0(A) &= A \\ f^{\alpha+1}(A) &= f(f^\alpha(A)) \\ f^\lambda(A) &= \bigcup_{\alpha<\lambda} f^\alpha(A)\,, \quad \lambda \text{ a limit ordinal} \\ f^*(A) &= \bigcup_\alpha f^\alpha(A) \\ 0(A) &= \varnothing \\ 1(A) &= A\,. \end{aligned}$$

Note that finite additivity implies monotonicity.

Proposition B.1 *The structure*

$$(\mathcal{F},\ \cup,\ \cdot,\ ^*,\ 0,\ 1)$$

is a right-handed Kleene algebra but not a left-handed Kleene algebra.

Proof. To show that $\mathcal{F}$ is a right-handed Kleene algebra, we need to verify all the Kleene algebra axioms except (**??**). The additive associativity, commutativity, idempotence, and identity laws hold since they hold for set union. Function composition is surely associative with two-sided identity 1. The function 0 is a left annihilator by definition and a right annihilator because of strictness of functions in $\mathcal{F}$. The left distributivity law holds by definition of $\cup$ and $\circ$, and the right distributivity law holds because of finite additivity of functions in $\mathcal{F}$. Thus $\mathcal{F}$ is an idempotent semiring.

The axioms (**??**–**??**) for * can be verified by transfinite induction. Let us verify the right-handed rule (**??**). Suppose $f \circ g \leq g$. Then for any $A \subseteq X$, $f(g(A)) \subseteq g(A)$. We have

$$\begin{aligned} f^0(g(A)) &= g(A) \\ \\ f^{\alpha+1}(g(A)) &= f(f^\alpha(g(A))) \\ &\subseteq f(g(A)) \text{ by induction hypothesis and monotonicity of } f \\ &\subseteq g(A) \\ \\ f^\lambda(g(A)) &= \bigcup_{\alpha<\lambda} f^\alpha(g(A)) \\ &\subseteq \bigcup_{\alpha<\lambda} g(A) \\ &= g(A)\,, \ \lambda \text{ a limit ordinal} \\ \\ f^*(g(A)) &= \bigcup_{\alpha} f^\alpha(g(A)) \\ &\subseteq g(A)\,. \end{aligned}$$

All we need to show that (**??**) is not satisfied is the existence of a strict, finitely additive function g that is not ω continuous, that is, such that

$$g^{\omega+1} \not\leq g^\omega\,. \tag{B.3}$$

Then taking $f = g^\omega$, we have

$$f \circ g = g^\omega \circ g \leq g^\omega = f\,,$$

but

$$\begin{aligned} f \circ g^* \leq f &\Rightarrow g^\omega \circ g^* \leq g^\omega \\ &\Rightarrow g^* \leq g^\omega \\ &\Rightarrow g^{\omega+1} \leq g^\omega\,, \end{aligned}$$

contradicting (B.3). There are many g that satisfy (B.3); for example, pick out a countable proper subset Y of X and call them the natural numbers; then define

$$g(A) \quad = \quad \begin{cases} A \cup \{x+1 \mid x \in A\}, & \text{if } A \text{ is a finite subset of } Y, \\ X, & \text{otherwise.} \end{cases}$$

Then $g^\omega(\{0\}) = Y$ but $g^{\omega+1}(\{0\}) = g(Y) = X$. □

Acknowledgments

I am grateful to John Horton Conway, Vaughan Pratt, and Dana Scott for their valuable comments.

Supplementary Lecture C

Models of KAT

In this lecture we explore some other classes of models of KA and KAT and their relationships. We have seen a number of different kinds of models of KA and KAT, including language models (sets of strings and sets of guarded strings), relational models (sets of pairs), and trace models (sets of traces). These are all examples of a general class of models which we might call *generalized language models*.

Typed Monoids, aka Small Categories

A *typed monoid* (or *small category*) is a two-sorted structure $(M, \Delta, \cdot, \mathsf{id}, \mathsf{src}, \mathsf{tgt})$, where

- M and Δ are sets (in the typed monoid view, the *elements* and *types*, respectively; and in the category-theoretic view, the *morphisms* and *objects*, respectively);
- $\mathsf{src} : M \to \Delta$ and $\mathsf{tgt} : M \to \Delta$ specify the *source* and *target* type of an element, respectively;
- $\cdot : M \times M \to M$ is a typed associative multiplication, often elided in expressions; and
- $\mathsf{id} : \Delta \to M$ associates an *identity element* with each type.

For $s, t \in \Delta$ and $m \in M$, we write $m : s \to t$ if $\mathsf{src}\, m = s$ and $\mathsf{tgt}\, m = t$. The typing of multiplication and the identities is governed by the following rules:

$$\mathsf{id}\, s : s \to s \qquad \frac{m : s \to t \qquad n : t \to u}{mn : s \to u}$$

The typed multiplication operation must be associative, and the typed identities are two-sided ientities for multiplication:

$$\frac{\mathsf{id}\, s : s \to s \qquad m : s \to t}{(\mathsf{id}\, s)m = m : s \to t} \qquad \frac{\mathsf{id}\, t : t \to t \qquad m : s \to t}{m\, \mathsf{id}\, t = m : s \to t} \qquad \frac{m : s \to t \qquad n : t \to u \qquad p : u \to v}{(mn)p = m(np) : s \to v}$$

Example C.1

1. *A monoid $(M, \cdot, 1)$ is a typed monoid with a single type. In particular, Σ^*, the free monoid on generators Σ, is a typed monoid with a single type whose multiplication operation is string concatenation and whose identity is the null string ε.*

2. *The guarded strings* GS *over an alphabet P of action letters and an alphabet B of test letters form a typed monoid with types* At*, the atoms of the free Boolean algebra generated by B. The type of a guarded string is $x : \mathsf{first}\, x \to \mathsf{last}\, x$. The typed multiplication operation is fusion product $x\alpha \diamond \alpha y = x\alpha y$. The identity at type $\alpha \in \mathsf{At}$ is the guarded string α of length 0.*

3. *The set $X \times X$ forms a typed monoid with elements $X \times X$ and types X. There is a unique element of each type: $(s, t) : s \to t$. Multiplication is $(s, t) \cdot (t, u) = (s, u)$. The identity elements are $\mathsf{id}\, s = (s, s)$.*

4. *The traces in a Kripke model (X, π, ρ) with $\pi : P \to 2^{X \times X}$ interpreting the action letters as binary relations on X and $\rho : B \to 2^X$ interpreting the primitive tests as subsets of X form a typed monoid. The types of traces are $\sigma : \mathsf{first}\, \sigma \to \mathsf{last}\, \sigma$. The identity elements are $\mathsf{id}\, s = s$, the traces of length 0, and multiplication is fusion product $\sigma \diamond \tau$. Note that guarded strings* (2) *are a special case. This is the free category generated by the labeled graph (X, π, ρ).*

A *morphism* of typed monoids is a structure-preserving map $f : M_1 \to M_2$. In category-theoretic terms, it is just a functor. Formally,

- $f(\mathsf{src}\, m) = \mathsf{src}\, f(m)$ and $f(\mathsf{tgt}\, m) = \mathsf{tgt}\, f(m)$; equivalently, if $m : s \to t$, then $f(m) : f(s) \to f(t)$;
- $f(mn) = f(m)f(n)$; and
- $f(\mathsf{id}\, s) = \mathsf{id}\, f(s)$.

Example C.2

1. *The map* $\sigma \mapsto (\mathsf{first}\,\sigma, \mathsf{last}\,\sigma)$ *is a morphism from the traces of a Kripke model* (X, π, ρ) *(Example C.1*(4) *above) to binary relations on* X *(Example C.1*(3) *above).*

Language Models

Given a typed monoid $(M, \Delta, \cdot, \mathsf{id}, \mathsf{src}, \mathsf{tgt})$, we can form a star-continuous KAT with subsets of M as actions and subsets of $\{\mathsf{id}\,s \mid s \in \Delta\}$ as tests. The KAT operations are defined as follows: for $A, B \subseteq M$ and $R \subseteq \{\mathsf{id}\,s \mid s \in \Delta\}$,

$$\begin{array}{lll} A + B = A \cup B & AB = \{xy \mid x \in A,\ y \in B,\ \mathsf{tgt}\,x = \mathsf{src}\,y\} & \\ 1 = \{\mathsf{id}\,s \mid s \in \Delta\} & 0 = \varnothing & \overline{R} = 1 - R \\ A^* = \bigcup_{n \geq 0} A^n, \quad \text{where } A^0 = 1 \text{ and } A^{n+1} = AA^n. & & \end{array} \tag{C.1}$$

Any subalgebra of such a KAT is called a *language model*. If the underlying typed monoid is a set of pairs as in Example C.1(3) above, the model is called a *relational model*.

Theorem C.3 *Every language model is isomorphic to a relational model.*

Proof. Given a subalgebra K of the language model on a typed monoid $(M, \Delta, \cdot, \mathsf{id}, \mathsf{src}, \mathsf{tgt})$ as defined in (C.1), for $A \in K$ define

$$h(A) = \{(x, xy) \mid x \in M,\ y \in A,\ \mathsf{tgt}\,x = \mathsf{src}\,y\}.$$

Note that on tests $R \subseteq \{\mathsf{id}\,s \mid s \in \Delta\}$, h returns a subset of the identity relation on M:

$$h(R) = \{(x, x\,\mathsf{id}\,s) \mid x \in M,\ \mathsf{id}\,s \in R,\ \mathsf{tgt}\,x = s\} = \{(x, x) \mid x \in M,\ \mathsf{id}\,\mathsf{tgt}\,x \in R\}.$$

We need to verify that h is a KAT homomorphism. We do this for multiplication and negation and leave the verification for the other operators as exercises.

For multiplication,

$$\begin{aligned}
&h(AB)\\
&= \{(x, xy) \mid x \in M,\ y \in AB,\ \mathsf{tgt}\, x = \mathsf{src}\, y\}\\
&= \{(x, xy) \mid x \in M,\ y \in \{uv \mid u \in A,\ v \in B,\ \mathsf{tgt}\, u = \mathsf{src}\, v\},\ \mathsf{tgt}\, x = \mathsf{src}\, y\}\\
&= \{(x, xuv) \mid x \in M,\ u \in A,\ v \in B,\ \mathsf{tgt}\, u = \mathsf{src}\, v,\ \mathsf{tgt}\, x = \mathsf{src}\, uv\}\\
&= \{(x, xu) \cdot (xu, xuv) \mid x \in M,\ u \in A,\ v \in B,\ \mathsf{tgt}\, u = \mathsf{src}\, v,\ \mathsf{tgt}\, x = \mathsf{src}\, u\}\\
&= \{(x, xu) \cdot (xu, xuv) \mid x \in M,\ u \in A,\ v \in B,\ \mathsf{tgt}\, xu = \mathsf{src}\, v,\ \mathsf{tgt}\, x = \mathsf{src}\, u\}\\
&= \{(x, xu) \cdot (y, yv) \mid x, y \in M,\ u \in A,\ v \in B,\ \mathsf{tgt}\, y = \mathsf{src}\, v,\ \mathsf{tgt}\, x = \mathsf{src}\, u,\ xu = y\}\\
&= \{(x, xu) \cdot (y, yv) \mid x, y \in M,\ u \in A,\ v \in B,\ \mathsf{tgt}\, y = \mathsf{src}\, v,\ \mathsf{tgt}\, x = \mathsf{src}\, u,\ \mathsf{tgt}(x, xu) = \mathsf{src}(y, yv)\}\\
&= \{(x, xu) \mid x \in M,\ u \in A,\ \mathsf{tgt}\, x = \mathsf{src}\, u\} \cdot \{(y, yv) \mid y \in M,\ v \in B,\ \mathsf{tgt}\, y = \mathsf{src}\, v\}\\
&= h(A)h(B).
\end{aligned}$$

For Boolean complement,

$$\begin{aligned}
h(\overline{R}) &= \{(x, xy) \mid x \in M,\ y \in \overline{R},\ \mathsf{tgt}\, x = \mathsf{src}\, y\}\\
&= \{(x, xy) \mid x \in M,\ y \in 1 - R,\ \mathsf{tgt}\, x = \mathsf{src}\, y\}\\
&= \{(x, x\, \mathsf{id}\, s) \mid x \in M,\ \mathsf{id}\, s \notin R,\ \mathsf{tgt}\, x = s\}\\
&= \{(x, x) \mid x \in M,\ \mathsf{id}\, \mathsf{tgt}\, x \notin R\}\\
&= 1 - h(R) = \overline{h(R)}.
\end{aligned}$$

The map h is an isomorphism, as A can be uniquely recovered from $h(A)$:

$$\begin{aligned}
A &= \{y \mid (\mathsf{id}\, \mathsf{src}\, y, y) \in \{(x, xy) \mid x \in M,\ y \in A,\ \mathsf{tgt}\, x = \mathsf{src}\, y\}\}\\
&= \{y \mid (\mathsf{id}\, \mathsf{src}\, y, y) \in h(A)\}.
\end{aligned}$$

□

Exercises

Homework 1

1. Prove the following theorems of KA. Reason equationally, using only the axioms of KA.

   (a) $x^* = x^*x^*$

   (b) $x^* = x^{**}$

   (c) $(x+y)^* = x^*(yx^*)^*$

   (d) $(xy)^*x = x(yx)^*$

   (e) for all $n \in \mathbb{N}$, $x^* = (1+x)^{n-1}(x^n)^*$

   (f) $xy = yz \to x^*y = yz^*$

2. A KA K is *commutative* if $xy = yx$ for all $x, y \in K$. Prove that the following hold in all commutative KAs.

   (a) $x^*y^* = (xy)^*(x^* + y^*)$

   (b) (Pilling normal form) Every regular expression over a finite alphabet Σ is equivalent to a sum of expressions of the form $xy_1^* \cdots y_n^*$, where $x, y_1, \ldots, y_n \in \Sigma^*$.

3. Let $\mathsf{Reg}\,\Sigma$ denote the Kleene algebra of regular sets over alphabet Σ. Prove that $\mathsf{Reg}\,\Sigma$ is isomorphic to a relation algebra.

4. Let Σ be a finite alphabet and K a Kleene algebra. A *power series in noncommuting variables* Σ *with coefficients in* K is a map $\sigma : \Sigma^* \to K$. The power series σ is often written as a formal sum

   $$\sum_{x \in \Sigma^*} \sigma(x) \cdot x.$$

   The set of all such power series is denoted $K\langle\langle\Sigma\rangle\rangle$. Addition on $K\langle\langle\Sigma\rangle\rangle$ is defined pointwise, and multiplication is defined as follows:

   $$(\sigma \cdot \tau)(x) \stackrel{\text{def}}{=} \sum_{x=yz} \sigma(y) \cdot \tau(z)$$

   where the sum is over all ways of expressing x as a product of two strings y and z.

   (a) Give definitions of 0 and 1. Convince yourself that $K\langle\langle\Sigma\rangle\rangle$ forms an idempotent semiring under these definitions.

(b) Define an operation * on $K\langle\langle\Sigma\rangle\rangle$ as follows:

$$\sigma^*(x) \stackrel{\text{def}}{=} \sum_{x=y_1\cdots y_n} \sigma(\varepsilon)^*\sigma(y_1)\sigma(\varepsilon)^*\sigma(y_2)\sigma(\varepsilon)^*\cdots\sigma(\varepsilon)^*\sigma(y_n)\sigma(\varepsilon)^*$$

where ε is the null string and the sum is over all ways of expressing x as a product of zero or more strings $y_1, \ldots, y_n$. Show that $K\langle\langle\Sigma\rangle\rangle$ forms a Kleene algebra under this definition.

Homework 2

1. Conway [29, p. 27] defines an S-*algebra* to be a structure $(S, \sum, 0, \cdot, 1, ^*)$ such that for arbitrary index sets I, J, J_i ($i \in I$), $e_i \in S$, and $e^0 = 1$, $e^{n+1} = e \cdot e^n$:

$$\sum_{i \in \varnothing} e_i = 0 \qquad\qquad e \cdot 1 = 1 \cdot e = e$$

$$\sum_{i \in I} \sum_{j \in J_i} e_j = \sum_{j \in \bigcup_{i \in I} J_i} e_j \qquad\qquad e_1 \cdot (e_2 \cdot e_3) = (e_1 \cdot e_2) \cdot e_3$$

$$(\sum_{i \in I} e_i) \cdot (\sum_{j \in J} e_j) = \sum_{(i,j) \in I \times J} e_i \cdot e_j \qquad\qquad e^* = \sum_{i=0}^{\infty} e^i$$

Binary addition is defined as $e_1 + e_2 = \sum_{i \in \{1,2\}} e_i$. Conway's axiomatization contains a subtle omission: the given axioms do not entail idempotence. Conway clearly intends idempotence to hold, since he uses it immediately afterward.

   (a) Construct an algebra satisfying Conway's axioms in which $\sum$ is not idempotent.

   (b) Give an axiom to correct this omission. Your axiom should be as succinct as possible.

2. Let M be an arbitrary monoid. The powerset 2^M becomes a star-continuous Kleene algebra under the usual language-theoretic definitions of the Kleene algebra operators:

$$A + B \stackrel{\text{def}}{=} A \cup B \qquad\qquad 0 \stackrel{\text{def}}{=} \varnothing$$

$$AB \stackrel{\text{def}}{=} \{xy \mid x \in A,\ y \in B\} \qquad\qquad 1 \stackrel{\text{def}}{=} \{1_M\}$$

$$A^* \stackrel{\text{def}}{=} \bigcup_n A^n$$

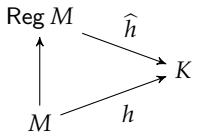
Define $\mathsf{Reg}\, M$ to be the smallest Kleene subalgebra of 2^M containing all singletons $\{x\}$, $x \in M$. Elements of $\mathsf{Reg}\, M$ are called *regular subsets* of M.

   (a) If $h : M \to M'$ is a monoid homomorphism, define $\mathsf{Reg}\, h : \mathsf{Reg}\, M \to \mathsf{Reg}\, M'$ by

$$\mathsf{Reg}\, h\, (A) \stackrel{\text{def}}{=} \{h(x) \mid x \in A\}.$$

   Show that $\mathsf{Reg}\, h$ is a Kleene algebra homomorphism. (Thus Reg constitutes a functor from the category Mon of monoids and monoid homomorphisms to the category KA^* of star-continuous Kleene algebras and Kleene algebra homomorphisms.)

(b) Show that any monoid homomorphism $h : M \to K$ from M to the multiplicative monoid of a star-continuous Kleene algebra K extends uniquely to a Kleene algebra homomorphism $\widehat{h} : \mathsf{Reg}\, M \to K$.



Homework 3

1. Prove that the $n \times n$ matrices over a star-continuous Kleene algebra again form a star-continuous Kleene algebra.

2. Strassen's matrix multiplication algorithm can be used to multiply two $n \times n$ matrices over a ring using approximately $n^{\log_2 7} = n^{2.807\cdots}$ multiplications in the underlying ring. The best known result of this form is by Coppersmith and Winograd, who achieve $n^{2.376\cdots}$. Show that over arbitrary semirings, n^3 multiplications are necessary in general. (*Hint*. Interpret over $\mathsf{Reg}\,\Sigma$, where $\Sigma = \{a_{ij}, b_{ij} \mid 1 \leq i, j \leq n\}$. What semiring expressions could possibly be equivalent to $\sum_{j=1}^{n} a_{ij} b_{jk}$?)

3. A *residuation algebra* is an idempotent semiring with extra operators $\rightarrow$ and $\leftarrow$, called *right* and *left residuation*, respectively, satisfying the following axioms:

$$ax \leq b \Leftrightarrow x \leq a \rightarrow b \tag{1}$$
$$xa \leq b \Leftrightarrow x \leq b \leftarrow a. \tag{2}$$

   The residuation operators give a kind of left and right inverse of $\cdot$.

   (a) Show that the regular sets of strings over a finite alphabet Σ and the binary relations on a set X form residuation algebras under appropriate definitions of $\rightarrow$ and $\leftarrow$.

   (b) Show that in the presence of the axioms of idempotent semirings, (1) and (2) are equivalent to the inequalities

$$a(a \rightarrow b) \leq b \tag{3}$$
$$(b \leftarrow a)a \leq b \tag{4}$$
$$a \rightarrow b \leq a \rightarrow (b + c) \tag{5}$$
$$b \leftarrow a \leq (b + c) \leftarrow a \tag{6}$$
$$x \leq a \rightarrow ax \tag{7}$$
$$x \leq xa \leftarrow a. \tag{8}$$

   Thus residuation algebras can be axiomatized purely equationally.

   (c) An *action algebra* [102] is a structure $(K, +, \cdot, ^*, 0, 1, \rightarrow, \leftarrow)$ that is both a residuation algebra and a Kleene algebra. Show that in the presence of the axioms of residuation algebras, the Kleene algebra

axioms for * are equivalent to

$$1 + a + a^*a^* \leq a^* \tag{9}$$
$$a^* \leq (a+b)^* \tag{10}$$
$$(x \to x)^* = x \to x \tag{11}$$
$$(x \leftarrow x)^* = x \leftarrow x. \tag{12}$$

Thus in the presence of $\to$ and $\leftarrow$, * can be axiomatized purely equationally.

Homework 4

1. Prove the following theorems of KAT:

   (a) In any KAT, if $bq = qb$, then

   $$bq^* \quad = \quad (bq)^*b \quad = \quad q^*b \quad = \quad b(qb)^*.$$

   (b) In any KA, if p is generated by a set of elements all of which commute with q, then p commutes with q.

2. (a) Prove the following relationship between the equational theory and the Hoare theory of KAT:

   $$\models p \leq q \text{ if and only if } \mathsf{REL} \models uqv = 0 \Rightarrow upv = 0,$$

   where u and v are action letters that do not occur in p or q.

   (b) Give a counterexample showing that the relationship does not hold without the u and v.

3. In reasoning with KAT, the pair of premises $p = pc$ and $cp = c$ for atomic action p and test c often arise (for example, see (15) and (16) in Exercise 4 below). The premise $p = pc$ is equivalent to $p\bar{c} = 0$, therefore can be eliminated using the generalization of Cohen's theorem to KAT. However, it can be shown that the premise $cp = c$ is not equivalent to any formula of the form $q = 0$.

   (a) Show that the following equations are equivalent:

   (i) $cp = c$
   (ii) $cp + \bar{c} = 1$
   (iii) $p = c + \bar{c}p$.

   (b) Show that premises of the form $cp = c$ can be eliminated when p is a primitive letter. *Hint*. Consider the relationships among the following metastatements:

   (A) $\models p = c + \bar{c}p \;\Rightarrow\; \varphi(p)$
   (B) $\models p = c + \bar{c}q \;\Rightarrow\; \varphi(p)$, where q is a new atomic action symbol
   (C) $\models \varphi(c + \bar{c}p)$.

4. **(The Dead Variable Paradox)** A variable is a *dead* at some point in a program if its value will never be used. For example, the variable x is dead immediately before the execution of the assignment $x := 2$. Let d

be the assertion "x is dead" and let p be the program $x := 2$. We have the following two properties:

$$p = dp \quad x \text{ is dead immediately before assigning to it} \tag{13}$$
$$d = pd \quad \text{it is pointless to assign a value to a dead variable.} \tag{14}$$

Let c be the assertion $x = 2$. We also have the following two properties:

$$p = pc \quad \text{assigning } x \text{ a value makes it have that value} \tag{15}$$
$$c = cp \quad \text{it is pointless to assign to a variable a value it already has} \tag{16}$$

Now it is clear that $pp = p$, since assigning the same value twice to the same variable is surely redundant. Here are two proofs of this fact using the properties above as assumptions.

$$\begin{array}{lcll} pp & = & pdp & \text{by (13)} \\ & = & dp & \text{by (14)} \\ & = & p & \text{by (13),} \end{array}$$

$$\begin{array}{lcll} pp & = & pcp & \text{by (15)} \\ & = & pc & \text{by (16)} \\ & = & p & \text{by (15).} \end{array}$$

Both of these proofs seem reasonable. But consider the following argument:

$$\begin{array}{lcll} pp & = & pcdp & \text{by (13) and (15)} \\ & = & pdcp & \text{by commutativity of Boolean multiplication} \\ & = & dc & \text{by (14) and (16).} \end{array}$$

We have just shown that two assignments $x := 2 \; ; \; x := 2$ are equivalent to asserting that x has value 2 and that it is dead. This is clearly absurd. So what is wrong with the argument?

Miscellaneous Exercises

The annotation $^{\mathrm{H}}$ indicates that there is a hint for this exercise in the Hints section beginning on p. 215, and $^{\mathrm{S}}$ indicates that there is a solution in the Solutions section beginning on p. 217. The number of stars indicates the approximate level of difficulty.

$^{\mathrm{HS}}$1. Prove that every propositional partial correctness assertion $\{b\}\,p\,\{c\}$ that is relationally valid without premises is derivable in extended Hoare logic.

?? We wish to show that the following are equivalent:

(a) $bp\overline{c} = 0$
(b) $bp = bpc$
(c) $bp \leq bpc$
(d) $bp \leq pc$.

This can be argued easily in KAT. Assuming (i),

$$\begin{aligned} bp &= bp(c + \overline{c}) && \text{by the axiom } a1 = a \text{ and Boolean algebra} \\ &= bpc + bp\overline{c} && \text{by distributivity} \\ &= bpc && \text{by Lemma 18.1(i) and the axiom } a + 0 = a. \end{aligned}$$

Conversely, assuming (ii),

$$\begin{aligned} bp\bar{c} &= bpc\bar{c} && \text{by Lemma 18.1(ii)} \\ &= bp0 && \text{by associativity and Boolean algebra} \\ &= 0 && \text{by the axiom } a0 = 0. \end{aligned}$$

The equation (ii) is equivalent to the inequality $bp \leq bpc$, since the reverse inequality is a theorem of KAT; it follows immediately from the axiom $c \leq 1$ of Boolean algebra and monotonicity of multiplication.

The equivalence of (i) and (ii) was observed by Manes and Arbib [81] in a different context.

Hints and Solutions

Homework 1 Solutions

1. Prove the following theorems of KA. Reason equationally, using only the axioms of KA.

   (a) $x^* = x^*x^*$

   We show the inequality in both directions. For $\leq$,

   $$\begin{aligned} x^* &\leq x^* + xx^*x^* && \text{since } a \leq a + b \\ &= (1 + xx^*)x^* && \text{distributivity} \\ &\leq x^*x^* && \text{axiom } 1 + aa^* \leq a^* \text{ and monotonicity.} \end{aligned}$$

   For $\geq$, we apply the star rule $b + ac \leq c \Rightarrow a^*b \leq c$ with $a = x$ and $b = c = x^*$.

   $$\begin{aligned} x^* + xx^* &\leq x^* + 1 + xx^* && \text{since } a \leq a + b \\ &= x^* + x^* && \text{axiom } 1 + aa^* \leq a^* \text{ and monotonicity} \\ &= x^* && \text{idempotence.} \end{aligned}$$

   By the star rule, $x^*x^* \leq x^*$.

   (b) $x^* = x^{**}$

   We show the inequality in both directions. For $\leq$,

   $$x^* \leq 1 + x^* + x^*x^*x^{**} = 1 + x^*(1 + x^*x^{**}) \leq 1 + x^*x^{**} \leq x^{**}.$$

   For $\geq$, we apply the star rule $b + ac \leq c \Rightarrow a^*b \leq c$ with $a = c = x^*$ and $b = 1$. Using (i),

   $$1 + x^*x^* \leq 1 + xx^* + x^*x^* \leq x^* + x^* \leq x^*.$$

   By the star rule, $x^{**} = x^{**}1 \leq x^*$.

   (c) $(x + y)^* = x^*(yx^*)^*$

   For the direction $\geq$, we replace x and y by $x + y$ everywhere and use monotonicity along with (i) and (ii).

   $$\begin{aligned} x^*(yx^*)^* &\leq (x + y)^*((x + y)(x + y)^*)^* \\ &\leq (x + y)^*(1 + (x + y)(x + y)^*)^* \\ &\leq (x + y)^*(x + y)^{**} \\ &= (x + y)^*(x + y)^* \\ &= (x + y)^*. \end{aligned}$$

For $\leq$, we apply the star rule $b + ac \leq c \Rightarrow a^*b \leq c$ with $a = x + y$, $b = 1$, and $c = x^*(yx^*)^*$.

$$\begin{aligned} 1 + (x+y)x^*(yx^*)^* &= 1 + xx^*(yx^*)^* + yx^*(yx^*)^* \\ &= xx^*(yx^*)^* + 1 + yx^*(yx^*)^* \\ &= xx^*(yx^*)^* + (yx^*)^* \\ &= (xx^* + 1)(yx^*)^* \\ &= x^*(yx^*)^*. \end{aligned}$$

By the star rule, $(x+y)^* = (x+y)^*1 \leq x^*(yx^*)^*$.

(d) $(xy)^*x = x(yx)^*$

By symmetry, we only need to prove $\leq$. We use the star rule $b + ac \leq c \Rightarrow a^*b \leq c$ with $a = xy$, $b = x$, and $c = x(yx)^*$.

$$x + xyx(yx)^* = x(1 + yx(yx)^*) = x(yx)^*.$$

By the star rule, $(xy)^*x \leq x(yx)^*$.

(e) For all $n \in \mathbb{N}$, $x^* = (1+x)^{n-1}(x^n)^*$

It follows from the axiom $1 + aa^* \leq a$ and monotonicity that $1 + x \leq x^*$:

$$1 + x \leq 1 + x + x^2x^* = 1 + x(1 + xx^*) \leq 1 + xx^* \leq x^*.$$

The right-to-left inequality follows from this, (a), (b), and monotonicity:

$$(1+x)^{n-1}(x^n)^* \leq (x^*)^{n-1}((x^*)^n)^* \leq x^*x^{**} \leq x^*x^* = x^*.$$

For the other direction, we first show by induction that $(1+x)^n = \sum_{i=0}^{n} x^i$:

$$(1+x)^{n+1} = (1+x)(1+x)^n = (1+x)\sum_{i=0}^{n} x^i = \sum_{i=0}^{n} x^i + \sum_{i=1}^{n+1} x^i = \sum_{i=0}^{n+1} x^i.$$

Here idempotence is used to combine terms of like powers. Now we apply the star rule $b + ac \leq c \Rightarrow a^*b \leq c$ with $a = x$, $b = 1$,

and $c = (1+x)^{n-1}(x^n)^*$.

$$\begin{aligned}
1 + x(1+x)^{n-1}(x^n)^* &= 1 + x(\sum_{i=0}^{n-1} x^i)(x^n)^* \\
&\leq (x^n)^* + (\sum_{i=1}^{n} x^i)(x^n)^* \\
&\leq (1 + \sum_{i=1}^{n} x^i)(x^n)^* \\
&= (\sum_{i=0}^{n-1} x^i + x^n)(x^n)^* \\
&= (\sum_{i=0}^{n-1} x^i)(x^n)^* + (x^n)(x^n)^* \\
&\leq (1+x)^{n-1}(x^n)^* + 1 + (x^n)(x^n)^* \\
&\leq (1+x)^{n-1}(x^n)^* + (x^n)^* \\
&= (1+x)^{n-1}(x^n)^*.
\end{aligned}$$

By the star rule, $x^* = x^*1 \leq (1+x)^{n-1}(x^n)^*$.

(f) $xy = yz \Rightarrow x^*y = yz^*$
By symmetry, we only need to prove $xy \leq yz \Rightarrow x^*y \leq yz^*$. We assume $xy \leq yz$ and use the star rule $b + ac \leq c \Rightarrow a^*b \leq c$ with $a = x$, $b = y$, and $c = yz^*$ under that assumption.

$$y + xyz^* \leq y + yzz^* = y(1 + zz^*) \leq yz^*.$$

The first inequality is from the assumption $xy \leq yz$ and monotonicity. By the star rule, $x^*y \leq yz^*$.

2. A KA K is *commutative* if $xy = yx$ for all $x, y \in K$. Prove that the following hold in all commutative KAs.

(a) $x^*y^* = (xy)^*(x^* + y^*)$
Both of these expressions are equivalent to $(x+y)^*$ in a commutative KA. Let's prove the three inequalities (i) $x^*y^* \leq (x+y)^*$, (ii) $(x+y)^* \leq (xy)^*(x^* + y^*)$, and (iii) $(xy)^*(x^* + y^*) \leq x^*y^*$.
(i) By monotonicity and 1(a), $x^*y^* \leq (x+y)^*(x+y)^* \leq (x+y)^*$.
(ii) By the star rule, it suffices to show $1 + (x+y)(xy)^*(x^* + y^*) \leq (xy)^*(x^* + y^*)$. Surely $1 \leq (xy)^*(x^* + y^*)$, and by symmetry, it suffices to show $x(xy)^*(x^* + y^*) \leq (xy)^*(x^* + y^*)$. By monotonicity, we need only show $x(xy)^*x^* \leq (xy)^*x^*$ and $x(xy)^*y^* \leq$

$(xy)^*(x^* + y^*)$. For the former,

$$x(xy)^*x^* = (xy)^*xx^* \leq (xy)^*(1 + xx^*) = (xy)^*x^*.$$

For the latter,

$$\begin{aligned} x(xy)^*y^* &= x(xy)^*(1 + yy^*) = x(xy)^* + x(xy)^*yy^* \\ &= (xy)^*x + xy(xy)^*y^* \leq (xy)^*x^* + (xy)^*y^* = (xy)^*(x^* + y^*). \end{aligned}$$

(iii) It suffices to show $(xy)^* \leq x^*y^*$, because then $(xy)^*(x^* + y^*) \leq (xy)^*(x^*y^* + x^*y^*) \leq x^*y^*(x^*y^* + x^*y^*) = x^*y^*$. To show $(xy)^* \leq x^*y^*$, we use the star rule. We need to show $1 + xyx^*y^* \leq x^*y^*$. Using commutativity,

$$1 + xyx^*y^* = 1 + xx^*yy^* \leq 1 + xx^* + yy^* + xx^*yy^* = (1 + xx^*)(1 + yy^*) = x^*y^*.$$

(b) (Pilling normal form) Every regular expression over a finite alphabet Σ is equivalent to a sum of expressions of the form $xy_1^* \cdots y_n^*$, where $x, y_1, \ldots, y_n \in \Sigma^*$.

The proof is by induction on the structure of the expression. The base cases $a \in \Sigma$, 0, and 1 are easy. For the induction step, if both e_1 and e_2 can be written as a sum of terms of the form $xy_1^* \cdots y_n^*$, then clearly so can $e_1 + e_2$. Their product can also be written in this form, since if $e_1 = \sum_i s_i$ and $e_2 = \sum_j t_j$, then by distributivity $e_1e_2 = \sum_{i,j} s_it_j$, and each term s_it_j can be rearranged to the form $xy_1^* \cdots y_n^*$ by commutativity.

It remains to treat the case e^*. We inductively write $e = t_1 + \cdots + t_n$ in normal form, so $e^* = (t_1 + \cdots + t_n)^*$ with each t_i of the form $xy_1^* \cdots y_n^*$. Using the equation $(x + y)^* = x^*y^*$ from part (a) inductively, this is equivalent to $t_1^* \cdots t_n^*$, so it remains to show how to write $(xy_1^* \cdots y_n^*)^*$ in normal form. We claim that

$$(xy_1^* \cdots y_n^*)^* = 1 + xx^*y_1^* \cdots y_n^*$$

which is of the desired form. We prove this by inequalities in both directions. For the inequality $\geq$, by monotonicity and commutativity,

$$\begin{aligned} 1 + xx^*y_1^* \cdots y_n^* &\leq 1 + x(xy_1^* \cdots y_n^*)^*y_1^* \cdots y_n^* \\ &= 1 + xy_1^* \cdots y_n^*(xy_1^* \cdots y_n^*)^* \\ &= (xy_1^* \cdots y_n^*)^*. \end{aligned}$$

For the inequality $\leq$, we use the star rule. It suffices to show

$$1 + (xy_1^* \cdots y_n^*)(1 + xx^*y_1^* \cdots y_n^*) \leq 1 + xx^*y_1^* \cdots y_n^*.$$

But

$$\begin{aligned}1 + (xy_1^* \cdots y_n^*)(1 + xx^*y_1^* \cdots y_n^*) &= 1 + xy_1^* \cdots y_n^* + xy_1^* \cdots y_n^* xx^* y_1^* \cdots y_n^* \\ &= 1 + xy_1^* \cdots y_n^* + x^2x^*y_1^*y_1^* \cdots y_n^*y_n^* \\ &= 1 + xy_1^* \cdots y_n^* + x^2x^*y_1^* \cdots y_n^* \\ &= 1 + x(1 + xx^*)y_1^* \cdots y_n^* \\ &= 1 + xx^*y_1^* \cdots y_n^*.\end{aligned}$$

3. For $A \subseteq \Sigma^*$, define $h(A) = \{(x, xy) \mid x \in \Sigma^*,\ y \in A\}$. This is a homomorphism:

$$\begin{aligned}h(\{\varepsilon\}) &= \{(x,x) \mid x \in \Sigma^*\} \\ h(\varnothing) &= \varnothing \\ h(\bigcup_i A_i) &= \{(x,xy) \mid x \in \Sigma^*,\ y \in \bigcup_i A_i\} \\ &= \bigcup_i \{(x,xy) \mid x \in \Sigma^*,\ y \in A_i\} \\ &= \bigcup_i h(A_i) \\ h(A \cdot B) &= \{(x,xy) \mid x \in \Sigma^*,\ y \in A \cdot B\} \\ &= \{(x,xyz) \mid x \in \Sigma^*,\ y \in A,\ z \in B\} \\ &= \{(x,xy) \mid x \in \Sigma^*,\ y \in A\} \circ \{(w,wz) \mid x \in \Sigma^*,\ z \in B\} \\ &= h(A) \circ h(B) \\ h(A^*) &= h(\bigcup_{n \geq 0} A^n) \\ &= \bigcup_{n \geq 0} h(A)^n.\end{aligned}$$

The map h is injective, since $A = \{y \mid (\varepsilon, y) \in h(A)\}$, therefore A is uniquely determined by $h(A)$. Thus $\mathsf{Reg}\,\Sigma$ (or for that matter any language model over Σ) is isomorphic to its image in $2^{\Sigma^* \times \Sigma^*}$ under h.

4. (b) Note that in the definition

$$\sigma^*(x) \stackrel{\text{def}}{=} \sum_{x = y_1 \cdots y_n} \sigma(\varepsilon)^* \sigma(y_1) \sigma(\varepsilon)^* \sigma(y_2) \sigma(\varepsilon)^* \cdots \sigma(\varepsilon)^* \sigma(y_n) \sigma(\varepsilon)^*,$$

the sum is formally infinite; however, the supremum exists, since if $x = y_1 \cdots y_n$ and $y_i = \varepsilon$, then $x = y_1 \cdots y_{i-1} y_{i+1} \cdots y_n$ and

$\sigma(\varepsilon)^*\sigma(y_i)\sigma(\varepsilon)^* \leq \sigma(\varepsilon)^*$. We also allow $n = 0$ in the case $x = \varepsilon$, which gives $\sigma^*(\varepsilon) \stackrel{\text{def}}{=} \sigma(\varepsilon)^*$.

We first show that $1 + \sigma\sigma^* \leq \sigma^*$. Since addition is defined pointwise, we can verify the inequality pointwise as well. For ε,

$$(1 + \sigma\sigma^*)(\varepsilon) \;=\; 1(\varepsilon) + \sigma(\varepsilon)\sigma^*(\varepsilon) \;=\; 1 + \sigma(\varepsilon)\sigma(\varepsilon)^* \;=\; \sigma(\varepsilon)^*.$$

For $x \neq \varepsilon$,

$$\begin{aligned}
&(1 + \sigma\sigma^*)(x) \\
&\quad= \quad 1(x) + \sum_{x=yz} \sigma(y)\sigma^*(z) \\
&\quad= \quad 0 + \sum_{x=yz} \sigma(y) \sum_{z=z_1\cdots z_n} \sigma(\varepsilon)^*\sigma(z_1)\sigma(\varepsilon)^* \cdots \sigma(\varepsilon)^*\sigma(z_n)\sigma(\varepsilon)^* \\
&\quad= \quad \sum_{x=yz_1\cdots z_n} \sigma(y)\sigma(\varepsilon)^*\sigma(z_1)\sigma(\varepsilon)^* \cdots \sigma(\varepsilon)^*\sigma(z_n)\sigma(\varepsilon)^* \\
&\quad\leq \quad \sum_{x=yz_1\cdots z_n} \sigma(\varepsilon)^*\sigma(y)\sigma(\varepsilon)^*\sigma(z_1)\sigma(\varepsilon)^* \cdots \sigma(\varepsilon)^*\sigma(z_n)\sigma(\varepsilon)^* \\
&\quad= \quad \sigma^*(x).
\end{aligned}$$

Now we show that

$$\sigma\tau \leq \tau \quad\Rightarrow\quad \sigma^*\tau \leq \tau. \tag{17}$$

(This is an alternative axiom for * as shown in class.) The left-hand side of (17) says that for all y, z,

$$\sigma(y)\tau(z) \quad\leq\quad \tau(yz) \tag{18}$$

and the right-hand side says that for all n and $y_1, \ldots, y_n, z$,

$$\sigma(\varepsilon)^*\sigma(y_1)\sigma(\varepsilon)^* \cdots \sigma(\varepsilon)^*\sigma(y_n)\sigma(\varepsilon)^*\tau(z) \quad\leq\quad \tau(y_1 \cdots y_n z) \tag{19}$$

We show (19) by induction on n. For $n = 0$, we must show that

$$\sigma(\varepsilon)^*\tau(z) \quad\leq\quad \tau(z). \tag{20}$$

This follows by an axiom from $\sigma(\varepsilon)\tau(z) \leq \tau(z)$, which is a instance of (18). Now suppose (19) holds for n. For $n + 1$, we have

$$\begin{aligned}
&\sigma(\varepsilon)^*\sigma(y_1)\sigma(\varepsilon)^* \cdots \sigma(\varepsilon)^*\sigma(y_{n+1})\sigma(\varepsilon)^*\tau(z) \\
&\quad\leq \quad \sigma(\varepsilon)^*\sigma(y_1)\tau(y_2 \cdots y_{n+1} z) && \text{induction hypothesis and monotonicity} \\
&\quad\leq \quad \sigma(\varepsilon)^*\tau(y_1 y_2 \cdots y_{n+1} z) && \text{by (18)} \\
&\quad\leq \quad \tau(y_1 \cdots y_{n+1} z) && \text{by (20).}
\end{aligned}$$

The arguments for the other axioms of * are symmetric.

Homework 2 Solutions

1. (a) Starting with any nontrivial monoid, define $0 \stackrel{\text{def}}{=} 1$ and define all sums to be 0. The monoid axioms hold since it is a monoid, the star axiom holds just by definition of e^*, and the rest hold since in each case both sides of the equation are 0.

   (b) $\sum\{1\} = 1$.

2. (b) Let K be a star-continuous Kleene algebra. It follows from the definition of $\mathsf{Reg}\, K$ that $A \in \mathsf{Reg}\, K$ iff there exists a finite set Σ, an interpretation $I : \mathsf{Exp}\,\Sigma \to K$, and a regular expression $t \in \mathsf{Exp}\,\Sigma$ such that $A = \{I(x) \mid x \in R_\Sigma(t)\}$. By a theorem proved in class, A has a supremum in K, since

$$\sup A = \sup\{I(x) \mid x \in R_\Sigma(t)\} \;=\; I(t).$$

Moreover, the supremum operator is a Kleene algebra homomorphism $\mathsf{Reg}\, K \to K$: $\sup \varnothing = 0$, $\sup\{1\} = 1$, and

$$\begin{aligned}
&\sup(\{I(x) \mid x \in R_\Sigma(s)\} \cup \{I(x) \mid x \in R_\Sigma(t)\}) \\
&\qquad = \sup\{I(x) \mid x \in R_\Sigma(s) \cup R_\Sigma(t)\} \\
&\qquad = \sup\{I(x) \mid x \in R_\Sigma(s+t)\} \\
&\qquad = I(s+t) \\
&\qquad = I(s) + I(t) \\
&\qquad = \sup\{I(x) \mid x \in R_\Sigma(s)\} + \sup\{I(x) \mid x \in R_\Sigma(t)\}
\end{aligned}$$

$$\begin{aligned}
&\sup(\{I(x) \mid x \in R_\Sigma(s)\} \cdot \{I(y) \mid y \in R_\Sigma(t)\}) \\
&\qquad = \sup\{I(x)I(y) \mid x \in R_\Sigma(s),\ y \in R_\Sigma(t)\} \\
&\qquad = \sup\{I(z) \mid z \in R_\Sigma(st)\} \\
&\qquad = I(st) \\
&\qquad = I(s)I(t) \\
&\qquad = \sup\{I(x) \mid x \in R_\Sigma(s)\} \cdot \sup\{I(x) \mid x \in R_\Sigma(t)\}.
\end{aligned}$$

The argument for * follows from $+$ and $\cdot$.
Now define $\widehat{h} : \mathsf{Reg}\, M \to K$ by

$$\widehat{h}(A) \stackrel{\text{def}}{=} \sup\{h(x) \mid x \in A\} = \sup(\mathsf{Reg}\, h(A)).$$

This is a homomorphism since it is a composition of homomorphisms $\mathsf{Reg}\, h : \mathsf{Reg}\, M \to \mathsf{Reg}\, K$ and $\sup : \mathsf{Reg}\, K \to K$. It is unique since it is determined by its action on the generators $\{\{x\} \mid x \in M\}$ of $\mathsf{Reg}\, M$: $\widehat{h}(\{x\}) = \{h(x)\}$.

Homework 3 Solutions

1. As argued in class, the $n \times n$ matrices form an idempotent semiring.

   To show that $AB^mC \leq AB^*C$ for all m, it suffices to show that $B^m \leq B^*$ for all m. This follows by induction from $I + B^*B = B^*$ using monotonicity, so it suffices to show that.

   For $n = 2$, we have

$$\begin{aligned}
&\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} a & b \\ c & d \end{bmatrix}^* \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} (a+bd^*c)^* & (a+bd^*c)^*bd^* \\ (d+ca^*b)^*ca^* & (d+ca^*b)^* \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} (a+bd^*c)^*(a+bd^*c) & (a+bd^*c)^*(b+bd^*d) \\ (d+ca^*b)^*(ca^*a+c) & (d+ca^*b)^*(ca^*b+d) \end{bmatrix} \\
&= \begin{bmatrix} 1+(a+bd^*c)^*(a+bd^*c) & (a+bd^*c)^*b(1+d^*d) \\ (d+ca^*b)^*c(a^*a+1) & 1+(d+ca^*b)^*(d+ca^*b) \end{bmatrix} \\
&= \begin{bmatrix} (a+bd^*c)^* & (a+bd^*c)^*bd^* \\ (d+ca^*b)^*ca^* & (d+ca^*b)^* \end{bmatrix} \\
&= \begin{bmatrix} a & b \\ c & d \end{bmatrix}^*.
\end{aligned}$$

   For $n \geq 3$, we use induction. Break up the matrix into four submatrices such that the upper left and lower right submatrices are square, and carry out the argument above, interpreting a, b, c, d as the four submatrices.

   Now we show that if $AB^mC \leq X$ for all m, then $AB^*C \leq X$. As above, we start with the case $n = 2$. First, it follows easily by induction and monotonicity that for $m \geq 0$,

$$\begin{bmatrix} a^m & 0 \\ 0 & d^m \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}^m \leq \begin{bmatrix} a & b \\ c & d \end{bmatrix}^m.$$

   It follows that

$$\begin{aligned}
\begin{bmatrix} bd^mc & 0 \\ 0 & ca^mb \end{bmatrix} &= \begin{bmatrix} 0 & b \\ c & 0 \end{bmatrix} \cdot \begin{bmatrix} a^m & 0 \\ 0 & d^m \end{bmatrix} \cdot \begin{bmatrix} 0 & b \\ c & 0 \end{bmatrix} \\
&\leq \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}^m \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \leq \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{m+2}.
\end{aligned}$$

Abbreviating $\sum_{i=0}^{m} A^i$ by $A^{\leq m}$, we thus have

$$\begin{aligned}\begin{bmatrix} a+bd^mc & 0 \\ 0 & d+ca^mb \end{bmatrix} &= \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} + \begin{bmatrix} bd^mc & 0 \\ 0 & ca^mb \end{bmatrix} \\ &\leq \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{m+2} \leq \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\leq m+2},\end{aligned}$$

therefore for any k and any $m_1, \ldots, m_k$, letting $\ell = \sum_{i=1}^{k}(m_i+2)$,

$$\begin{bmatrix} \prod_{i=1}^{k}(a+bd^{m_i}c) & 0 \\ 0 & \prod_{i=1}^{k}(d+ca^{m_i}b) \end{bmatrix} \leq \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\leq \ell}$$

and

$$\begin{aligned}&\begin{bmatrix} \prod_{i=1}^{k}(a+bd^{m_i}c) & \prod_{i=1}^{k}(a+bd^{m_i}c)bd^j \\ \prod_{i=1}^{k}(d+ca^{m_i}b)ca^j & \prod_{i=1}^{k}(d+ca^{m_i}b) \end{bmatrix} \\ &= \begin{bmatrix} \prod_{i=1}^{k}(a+bd^{m_i}c) & 0 \\ 0 & \prod_{i=1}^{k}(d+ca^{m_i}b) \end{bmatrix} \cdot \begin{bmatrix} 1 & bd^j \\ ca^j & 1 \end{bmatrix} \\ &= \begin{bmatrix} \prod_{i=1}^{k}(a+bd^{m_i}c) & 0 \\ 0 & \prod_{i=1}^{k}(d+ca^{m_i}b) \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & b \\ c & 0 \end{bmatrix} \cdot \begin{bmatrix} a^j & 0 \\ 0 & d^j \end{bmatrix} \right) \\ &\leq \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\leq \ell} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\leq j+1} \\ &= \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\leq \ell+j+1}.\end{aligned}$$

Thus for any A and C,

$$A \cdot \begin{bmatrix} \prod_{i=1}^{k}(a+bd^{m_i}c) & \prod_{i=1}^{k}(a+bd^{m_i}c)bd^j \\ \prod_{i=1}^{k}(d+ca^{m_i}b)ca^j & \prod_{i=1}^{k}(d+ca^{m_i}b) \end{bmatrix} \cdot C \leq A \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\leq \ell+j+1} \cdot C.$$

Now if

$$A \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{m} \cdot C \leq \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

for all m, then for all k, j, and $m_1, \ldots, m_k$,

$$A \cdot \begin{bmatrix} \prod_{i=1}^{k}(a+bd^{m_i}c) & \prod_{i=1}^{k}(a+bd^{m_i}c)bd^j \\ \prod_{i=1}^{k}(d+ca^{m_i}b)ca^j & \prod_{i=1}^{k}(d+ca^{m_i}b) \end{bmatrix} \cdot C \leq A \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\leq \ell+j+1} \cdot C \leq \begin{bmatrix} x & y \\ z & w \end{bmatrix}.$$

It follows from the axiom of star-continuity that for all k,

$$A \cdot \begin{bmatrix} (a+bd^*c)^k & (a+bd^*c)^k bd^* \\ (d+ca^*b)^k ca^* & (d+ca^*b)^k \end{bmatrix} \cdot C \leq \begin{bmatrix} x & y \\ z & w \end{bmatrix},$$

therefore

$$A \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}^* \cdot C = A \cdot \begin{bmatrix} (a+bd^*c)^* & (a+bd^*c)^*bd^* \\ (d+ca^*b)^*ca^* & (d+ca^*b)^* \end{bmatrix} \cdot C \leq \begin{bmatrix} x & y \\ z & w \end{bmatrix}.$$

As above, the result for $n \geq 3$ is obtained by induction, interpreting a, b, c, d as submatrices of an $n \times n$ matrix with a and d square.

2. Let $\Sigma = \{a_{ij}, b_{ij} \mid 1 \leq i, j \leq n\}$. Let R be the standard interpretation $R : \mathsf{Exp}\,\Sigma \to \mathsf{Reg}\,\Sigma$. Let A and B be two matrices whose ij^{th} entries are a_{ij} and b_{ij}, respectively. Under the interpretation R, the ij^{th} entry of AB is the regular set $\{a_{ik}b_{kj} \mid 1 \leq k \leq n\}$. To compute AB, we must have calculated n^2 subexpressions e_{ij}, $1 \leq i, j \leq n$, where $R(e_{ij}) = \{a_{ik}b_{kj} \mid 1 \leq k \leq n\}$.

   Let s be an expression over Σ such that $R(s) \subseteq \{a_{ik}b_{kj} \mid 1 \leq k \leq n\}$. We show by induction on the structure of s that for any $a_{ik}b_{kj} \in R(s)$, s must contain a subexpression pq such that $R(p) = \{a_{ik}\}$ and $R(q) = \{b_{kj}\}$.

   If s is 0, then $R(s) = \varnothing$ and there is nothing to prove.

   The expression s cannot be of the form 1 or t^*, because then $\varepsilon \in R(s)$, but $\varepsilon \notin \{a_{ik}b_{kj} \mid 1 \leq k \leq n\}$.

   If s is of the form $u + v$, then $R(s) = R(u) \cup R(v)$ and the result follows from the induction hypothesis on the smaller terms u and v.

   If s is of the form uv, and one of $R(u)$ or $R(v)$ is $\varnothing$, then $R(s) = \varnothing$, and there is nothing to prove. Otherwise both $R(u)$ and $R(v)$ are nonempty. Neither $R(u)$ nor $R(v)$ contains strings of two different lengths, otherwise $R(s)$ would contain strings of two different lengths, which is impossible by assumption. If $R(u) = \{\varepsilon\}$, then $R(v) = R(s)$, and the result follows by the induction hypothesis on v, and similarly if $R(v) = \{\varepsilon\}$. The only other possibility is if both $R(u)$ and $R(v)$ contain strings of length 1. But each may contain no more than one string of length 1, otherwise $R(uv)$ would contain a string of length 2 not in $\{a_{ik}b_{kj} \mid 1 \leq k \leq n\}$. Thus $R(u)$ and $R(v)$ each contain a single letter, and it must be that $R(u) = \{a_{ik}\}$ and $R(v) = \{b_{kj}\}$ for some k in order for $R(uv) \subseteq \{a_{ik}b_{kj} \mid 1 \leq k \leq n\}$, so u and v are the desired p and q, respectively.

   Applying this result to e_{ij}, we see that e_{ij} must contain n subexpressions $p_{ijk}q_{ijk}$, $1 \leq k \leq n$, such that $R(p_{ijk}) = \{a_{ik}\}$ and $R(q_{ijk}) = \{b_{kj}\}$. All these subexpressions $p_{ijk}q_{ijk}$ must be distinct, since their interpretations $R(p_{ijk}q_{ijk})$ are all distinct. Thus these n^3 subexpressions account for n^3 distinct multiplications.

3. (a) For $A, B \subseteq \Sigma^*$, define

$$B \leftarrow A \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \forall y \in A\ xy \in B\}.$$

Then $\leftarrow$ is a left residuation operator, since (2) is satisfied:

$$\begin{aligned} C \subseteq B \leftarrow A &\Leftrightarrow C \subseteq \{x \in \Sigma^* \mid \forall y \in A\ xy \in B\} \\ &\Leftrightarrow x \in C \Rightarrow y \in A \Rightarrow xy \in B \\ &\Leftrightarrow x \in C \wedge y \in A \Rightarrow xy \in B \\ &\Leftrightarrow CA \subseteq B. \end{aligned}$$

We must show that $B \leftarrow A$ is regular if A and B are. Actually $B \leftarrow A$ is regular whenever B is, regardless of whether A is. Let M be a finite automaton for B with states Q, start state s, accept states F, and transition function δ. Define a new automaton M' that is identical to M except for its accept states, which are

$$F' \stackrel{\text{def}}{=} \{q \in Q \mid \forall y \in A\ \delta(q, y) \in F\}.$$

Then M' accepts $B \leftarrow A$, since

$$\begin{aligned} M' \text{ accepts } x &\Leftrightarrow \delta(s, x) \in F' \\ &\Leftrightarrow \delta(s, x) \in \{q \in Q \mid \forall y \in A\ \delta(q, y) \in F\} \\ &\Leftrightarrow \forall y \in A\ \delta(\delta(s, x), y) \in F \\ &\Leftrightarrow \forall y \in A\ \delta(s, xy) \in F \\ &\Leftrightarrow \forall y \in A\ xy \in B \\ &\Leftrightarrow x \in B \leftarrow A. \end{aligned}$$

The right residuation operator $A \rightarrow B$ is defined symmetrically and its properties established by symmetric analogs of the arguments above.

For binary relations R, S on a set X, define

$$B \leftarrow A \stackrel{\text{def}}{=} \{(x, y) \mid \forall z\ (y, z) \in A \Rightarrow (x, z) \in B\}.$$

Then $\leftarrow$ is a left residuation operator, since (2) is satisfied:

$$\begin{aligned} C \subseteq B \leftarrow A &\Leftrightarrow (x, y) \in C \Rightarrow (x, y) \in B \leftarrow A \\ &\Leftrightarrow (x, y) \in C \Rightarrow (y, z) \in A \Rightarrow (x, z) \in B \\ &\Leftrightarrow (x, y) \in C \wedge (y, z) \in A \Rightarrow (x, z) \in B \\ &\Leftrightarrow (x, z) \in C\ ;\ A \Rightarrow (x, z) \in B \\ &\Leftrightarrow C\ ;\ A \subseteq B. \end{aligned}$$

The right residuation operator $\rightarrow$ is defined symmetrically.

(b) We show that (1) is equivalent to the conjunction of (3), (5), and (7). That (2) is equivalent to the conjunction of (4), (6) and (8) follows by symmetric arguments.

Assume (1). Then (3) follows by applying (1) in the right-to-left direction with $x = a \to b$. For (5), we have $a(a \to b) \leq b + c$ by (3); then (5) follows by applying (1) in the left-to-right direction. Finally, (7) follows by applying (1) in the left-to-right direction with $b = ax$.

Now assume (3), (5), and (7). Assuming the left-hand side of (1), we have

$$\begin{aligned} x &\leq a \to ax && \text{by (7)} \\ &\leq a \to (ax + b) && \text{by (5)} \\ &= a \to b && \text{by the left-hand side of (1).} \end{aligned}$$

This gives the right-hand side of (1). Conversely, assuming the right-hand side of (1), we have

$$\begin{aligned} x \leq a \to b &\Rightarrow ax \leq a(a \to b) && \text{by monotonicity} \\ &\Rightarrow ax \leq b && \text{by (3),} \end{aligned}$$

which is the left-hand side of (1).

(c) First we show that the axioms of Kleene algebra and residuation algebra imply (9)–(12). The axioms of KA alone imply (9) and (10), as shown in class. For (11), we have $x \to x \leq (x \to x)^*$ by (9). To show $(x \to x)^* \leq x \to x$, it suffices to show that $1 + (x \to x)(x \to x) \leq x \to x$. But $1 \leq x \to x$ is immediate from (1). To show $(x \to x)(x \to x) \leq x \to x$, by (1) it suffices to show $x(x \to x)(x \to x) \leq x$. This follows from two applications of the fact $x(x \to x) \leq x$, which is an instance of (3), and monotonicity. The property (12) follows from a symmetric argument.

Assuming (9)–(12), the Kleene algebra axiom $1 + xx^* \leq x^*$ follows quite easily from (9) and monotonicity. For the KA axiom $ax \leq x \Rightarrow a^*x \leq x$,

$$\begin{aligned} ax \leq x &\Rightarrow a \leq x \leftarrow x && \text{by (2)} \\ &\Rightarrow a^* \leq (x \leftarrow x)^* && \text{by (10)} \\ &\Rightarrow a^* \leq x \leftarrow x && \text{by (12)} \\ &\Rightarrow a^*x \leq x && \text{by (2).} \end{aligned}$$

Homework 4 Solutions

1. (a) That $bq^* = q^*b$ and $(bq)^*b = b(qb)^*$ follow from part (b). Also, $b(qb)^* \leq bq^*$ follows directly from monotonicity, so it remains to show $bq^* \leq b(qb)^*$. By an axiom of KA, it suffices to show $b + b(qb)^*q \leq b(qb)^*$. But

$$\begin{aligned} b + b(qb)^*q &\leq b + bb(qb)^*q && \text{by Boolean algebra} \\ &= b + b(qb)^*qb && \text{by part (b), as argued above} \\ &= b(1 + (qb)^*qb) && \text{by distributivity} \\ &= b(qb)^* && \text{by an axiom of KA.} \end{aligned}$$

   (b) This is a straightforward induction on the construction of p from the generators. The only nontrivial case is the case of *, for which we must show that $rq = qr \Rightarrow r^*q = qr^*$. By symmetry, it suffices to show that $rq \leq qr \Rightarrow r^*q \leq qr^*$. By an axiom of Kleene algebra, it suffices to show that $rq \leq qr \Rightarrow q + rqr^* \leq qr^*$. But

$$\begin{aligned} q + rqr^* &\leq q + qrr^* && \text{by the assumption } rq \leq qr \\ &= q(1 + rr^*) && \text{by distributivity} \\ &= qr^* && \text{by an axiom of KA.} \end{aligned}$$

2. (a) Suppose REL $\not\models p \leq q$. Let K be a relational interpretation with $(s,t) \in K(p) - K(q)$. Reinterpret u and v so that $K(u) = (s,s)$ and $K(v) = (t,t)$. This can be done without changing the interpretation of p or q by the assumption that neither u nor v occurs in p or q. Then $K(upv) = \{(s,t)\}$ but $K(uqv) = \varnothing$, so $K \not\models uqv = 0 \Rightarrow upv = 0$.

      The converse follows immediately from the transitivity of $\leq$.

   (b) The direction ($\Rightarrow$) holds in any Kleene algebra: if $p \leq q$ and $q = 0$, then $p = 0$ follows from transitivity of $\leq$. The other direction does not hold. Let q be an atomic program and let $p = qq$. Certainly $q = 0 \Rightarrow qq = 0$ in any relational model, but this does not imply that $qq \leq q$. For example, take the states to be $\{0,1\}$ and interpret q as the relation $\{(0,1),(1,0)\}$.

3. (a) (i) $\Rightarrow$ (ii): $cp = c \Rightarrow cp + \overline{c} = c + \overline{c} = 1$.
   (ii) $\Rightarrow$ (i): $1 = cp + \overline{c} \Rightarrow c = c(cp + \overline{c}) = ccp + c\overline{c} = cp$.
   (i) $\Rightarrow$ (iii): $cp = c \Rightarrow p = (c + \overline{c})p = cp + \overline{c}p = c + \overline{c}p$.
   (iii) $\Rightarrow$ (i): $p = c + \overline{c}p \Rightarrow cp = c(c + \overline{c}p) = cc + c\overline{c}p = c$.

(b) (A) $\Rightarrow$ (B): Assume $p = c + \bar{c}q$. Multiplying on the left by c, we get $cp = c$, which by (a) is equivalent to $p = c + \bar{c}p$. By (A), $\varphi(p)$ holds.
(B) $\Rightarrow$ (C): Since p is atomic, we can substitute $c + \bar{c}q$ for p in $\varphi(p)$ to get $\varphi(c + \bar{c}q)$, and p does not occur in this formula. Thus the premise $p = c + \bar{c}q$ is essentially a definition of p in terms of q, but is irrelevant to the truth of $\varphi(c + \bar{c}q)$ and can be eliminated. But $\varphi(c + \bar{c}q)$ is equivalent to $\varphi(c + \bar{c}p)$, since both p and q are atomic and do not occur elsewhere in the expression.
(C) $\Rightarrow$ (A): This is immediate from the law of substitution of equals for equals.

4. The problem is that the proposition d = "x is a dead variable" is not a property of the local state, but depends on nonlocal information determined by the program context. The Kleene algebra axioms can change the context, and with it the truth of d, without changing the input/output relation of the program. Thus d cannot be modeled by a binary relation on states, since its truth or falsity is not determined solely by the state.

   One solution is to use a program w whose interpretation is "make x undefined" instead of d. The program w can be regarded as an assignment of an undefined value to x. As such, it is a transformation of the local state, much like an ordinary assignment. Since w is a program and not a test, it is not required by the axioms of KAT to commute with tests.

Hints for Selected Miscellaneous Exercises

1. Induction on the structure of p.

Solutions to Selected Miscellaneous Exercises

1. Let $\{b\}\,p\,\{c\}$ be a relationally valid PCA. By Theorem **??**, $bp\bar{c} = 0$ is a theorem of KAT. Now we give a proof of $\{b\}\,p\,\{c\}$ in HL by induction on p.

   For $p = q + r$, we have

   $$\begin{array}{lcll} b(q+r)\bar{c} = 0 & \Rightarrow & bq\bar{c} = 0 \text{ and } br\bar{c} = 0 & \\ & \Rightarrow & \vdash_{\mathsf{HL}} \{b\}\,q\,\{c\} \text{ and } \vdash_{\mathsf{HL}} \{b\}\,r\,\{c\} & \text{by the induction hypothesis} \\ & \Rightarrow & \vdash_{\mathsf{HL}} \{b\}\,q+r\,\{c\} & \text{by the + rule of HL.} \end{array}$$

   For $p = qr$, we have

   $$bqr\bar{c} = 0 \quad \Rightarrow \quad \mathsf{GS}(bq)\mathsf{GS}(r\bar{c}) = \varnothing,$$

   where GS is the guarded string interpretation. This says that there do not exist guarded strings $\sigma \in \mathsf{GS}(bq)$ and $\tau \in \mathsf{GS}(r\bar{c})$ such that $\mathsf{last}\,\sigma = \mathsf{first}\,\tau$. (Recall that $\mathsf{last}\,\sigma$ is the last atom in σ and $\mathsf{first}\,\tau$ is the first atom in τ). Let $d = \bigvee\{\mathsf{last}\,\sigma \mid \sigma \in \mathsf{GS}(bq)\}$. Then $\mathsf{GS}(bq) \subseteq \mathsf{GS}(bqd)$ and $\mathsf{GS}(r\bar{c}) \subseteq \mathsf{GS}(\bar{d}r\bar{c})$, therefore by completeness of KATover the guarded string model, $bq\bar{d} = 0$ and $dr\bar{c} = 0$ are theorems of KAT. By the induction hypothesis, $\vdash_{\mathsf{HL}} \{b\}\,q\,\{d\}$ and $\vdash_{\mathsf{HL}} \{d\}\,r\,\{c\}$. By the composition rule of HL, $\vdash_{\mathsf{HL}} \{b\}\,qr\,\{c\}$.

For $p = q^*$, we have

$$bq^*\bar{c} = 0 \quad \Rightarrow \quad \mathsf{GS}(bq^*\bar{c}) = \varnothing,$$

so for all n, $\mathsf{GS}(bq^n\bar{c}) = \varnothing$. Let $d = \bigvee\{\mathsf{last}\,\sigma \mid \sigma \in \mathsf{GS}(bq^n),\ n \geq 0\}$. Then $b \leq d$, $d \leq c$, and $\mathsf{GS}(dq) \subseteq \mathsf{GS}(dqd)$, thus $dq\bar{d} = 0$ is a theorem of KAT. By the induction hypothesis, $\vdash_{\mathsf{HL}} \{d\}\,q\,\{d\}$. By the * rule of HL, $\vdash_{\mathsf{HL}} \{d\}\,q^*\,\{d\}$, and by weakening, $\vdash_{\mathsf{HL}} \{b\}\,q^*\,\{c\}$.

?? Assuming Lemma 18.1(i),

$$\begin{aligned} bp &= bp(c + \bar{c}) && \text{by the axiom } a1 = a \text{ and Boolean algebra} \\ &= bpc + bp\bar{c} && \text{by distributivity} \\ &= bpc && \text{by Lemma 18.1(i) and the axiom } a + 0 = a. \end{aligned}$$

Conversely, assuming Lemma 18.1(ii),

$$\begin{aligned} bp\bar{c} &= bpc\bar{c} && \text{by Lemma 18.1(ii)} \\ &= bp0 && \text{by associativity and Boolean algebra} \\ &= 0 && \text{by the axiom } a0 = 0. \end{aligned}$$

The equation Lemma 18.1(ii) is equivalent to the inequality $bp \leq bpc$, since the reverse inequality is a theorem of KAT; it follows immediately from the axiom $c \leq 1$ of Boolean algebra and monotonicity of multiplication.

References

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1975.

[2] S. Anderaa. On the algebra of regular expressions. Appl. Math., Harvard Univ., 1965. Cambridge, Mass., 1–18.

[3] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. NetKAT: Semantic foundations for networks. In *Proc. 41st ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages (POPL'14)*, pages 113–126, San Diego, California, USA, January 2014. ACM.

[4] Allegra Angus and Dexter Kozen. Kleene algebra with tests and program schematology. Technical Report TR2001-1844, Computer Science Department, Cornell University, July 2001.

[5] K. R. Apt. Ten years of Hoare's logic: a survey—part I. *ACM Trans. Programming Languages and Systems*, 3:431–483, 1981.

[6] K. V. Archangelsky. A new finite complete solvable quasiequational calculus for algebra of regular languages. Manuscript, Kiev State University, 1992.

[7] Roland Carl Backhouse. *Closure Algorithms and the Star-Height Problem of Regular Languages*. PhD thesis, Imperial College, London, U.K., 1975.

[8] Adam Barth and Dexter Kozen. Equational verification of cache blocking in LU decomposition using Kleene algebra with tests. Technical Report TR2002-1865, Computer Science Department, Cornell University, June 2002.

[9] F. Berman. A completeness technique for *D*-axiomatizable semantics. In *Proc. 11th Symp. Theory of Comput.*, pages 160–166. ACM, 1979.

[10] Jean Berstel. *Transductions and Context-free Languages*. Teubner, Stuttgart, Germany, 1979.

[11] Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*. Springer-Verlag, Berlin, 1984.

[12] Stephen L. Bloom and Zoltán Ésik. Floyd–Hoare logic in iteration theories. *J. Assoc. Comput. Mach.*, 38(4):887–934, October 1991.

[13] Stephen L. Bloom and Zoltán Ésik. Program correctness and matricial iteration theories. In *Proc. 7th Int. Conf. Mathematical Foundations of Programming Semantics*, volume 598 of *Lecture Notes in Computer Science*, pages 457–476. Springer-Verlag, 1992.

[14] Stephen L. Bloom and Zoltán Ésik. Equational axioms for regular sets. *Math. Struct. Comput. Sci.*, 3:1–24, 1993.

[15] Maurice Boffa. Une remarque sur les systèmes complets d'identités rationnelles. *Informatique Théoretique et Applications/Theoretical Informatics and Applications*, 24(4):419–423, 1990.

[16] C. Böhm and G. Jacopini. Flow diagrams, Turing machines, and languages with only two formation rules. *Commun. ACM*, 9(5):366–371, May 1966.

[17] Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra M. Silva. Regular expressions for polynomial coalgebras. Technical Report SEN-E0703, Centrum voor Wiskunde en Informatica, Amsterdam, December 2007.

[18] Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra M. Silva. A Kleene theorem for polynomial coalgebras. Manuscript, May 2008.

[19] Janusz A. Brzozowski. Derivatives of regular expressions. *J. Assoc. Comput. Mach.*, 11:481–494, 1964.

[20] B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[21] Hubie Chen and Riccardo Pucella. A coalgebraic approach to Kleene algebra with tests. *Electronic Notes in Theoretical Computer Science*, 82(1), 2003.

[22] E. M. Clarke, S. M. German, and J. Y. Halpern. Effective axiomatizations of Hoare logics. *J. Assoc. Comput. Mach.*, 30:612–636, 1983.

[23] Ernie Cohen. Hypotheses in Kleene algebra. Technical Report TM-ARH-023814, Bellcore, 1993. `http://citeseer.nj.nec.com/1688.html`.

[24] Ernie Cohen, February 1994. Personal communication.

[25] Ernie Cohen. Lazy caching in Kleene algebra, 1994. `http://citeseer.nj.nec.com/22581.html`.

[26] Ernie Cohen. Using Kleene algebra to reason about concurrency control. Technical report, Telcordia, Morristown, N.J., 1994.

[27] Ernie Cohen and Dexter Kozen. A note on the complexity of propositional Hoare logic. *Trans. Computational Logic*, 1(1):171–174, July 2000.

[28] Ernie Cohen, Dexter Kozen, and Frederick Smith. The complexity of Kleene algebra with tests. Technical Report TR96-1598, Computer Science Department, Cornell University, July 1996.

[29] John Horton Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, London, 1971.

[30] S. A. Cook. The complexity of theorem proving procedures. In *Proc. 3rd Symp. Theory of Computing*, pages 151–158. ACM, 1971.

[31] S. A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM J. Comput.*, 7(1):70–90, February 1978.

[32] Patrick Cousot. Methods and logics for proving programs. In J. van Leeuwen, editor, *Handbood of Theoretical Computer Science*, volume B, pages 841–993. Elsevier, Amsterdam, 1990.

[33] Martin Davis. *Computability and Unsolvability*. McGraw-Hill, New York, 1958.

[34] S. Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, New York, 1974.

[35] Michael J. Fischer and Richard E. Ladner. Propositional modal logic of programs. In *Proc. 9th Symp. Theory of Comput.*, pages 286–294. ACM, 1977.

[36] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.

[37] Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. A coalgebraic decision procedure for NetKAT. In *Proc. 42nd ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages (POPL'15)*, pages 343–355, Mumbai, India, January 2015. ACM.

[38] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[39] F. Gécseg and I. Peák. *Algebraic Theory of Automata*. Akadémiai Kiadó, Budapest, 1972.

[40] Alan Gibbons and Wojciech Rytter. On the decidability of some problems about rational subsets of free partially commutative monoids. *Theoretical Computer Science*, 48:329–337, 1986.

[41] P. V. Gorshkov. Rational data structures and their applications. *Cybernetics*, 25(6):760–765, Nov.–Dec. 1989.

[42] D. Gries. *The Science of Programming*. Springer-Verlag, 1981.

[43] J. Y. Halpern and J. H. Reif. The propositional dynamic logic of deterministic, well-structured programs. *Theor. Comput. Sci.*, 27:127–165, 1983.

[44] Chris Hardin and Dexter Kozen. On the complexity of the Horn theory of REL. Technical Report TR2003-1896, Computer Science Department, Cornell University, May 2003.

[45] David Harel. On folk theorems. *Comm. Assoc. Comput. Mach.*, 23(7):379–389, July 1980.

[46] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

[47] Peter G. Hinman. *Recursion Theoretic Hierarchies*. Springer-Verlag, New York, 1977.

[48] K. Hirose and M. Oya. General theory of flowcharts. *Comment. Math. Univ. St. Pauli*, 21(2):55–71, 1972.

[49] C. A. R. Hoare. An axiomatic basis for computer programming. *Comm. Assoc. Comput. Mach.*, 12:576–80, 1969.

[50] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[51] Mark Hopkins and Dexter Kozen. Parikh's theorem in commutative Kleene algebra. In *Proc. 14th Conf. Logic in Computer Science (LICS'99)*, pages 394–401. IEEE, July 1999.

[52] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen, 1968.

[53] Kazuo Iwano and Kenneth Steiglitz. A semiring on convex polygons and zero-sum cycle problems. *SIAM J. Comput.*, 19(5):883–901, 1990.

[54] B. Jonsson and A. Tarski. Representation problems for relation algebras, abstract 89t. *Bull. Amer. Math. Soc.*, 54:80, 1948.

[55] Donald M. Kaplan. Regular expressions and the equivalence of programs. *J. Comput. Syst. Sci.*, 3:361–386, 1969.

[56] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, N.J., 1956.

[57] D. C. Kozen. On Kleene algebras and closed semirings. In Rovan, editor, *Proc. Math. Found. Comput. Sci. 1990*, volume 452 of *Lect. Notes in Comput. Sci.*, pages 26–47. Springer-Verlag, 1990.

[58] D. C. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Proc. 6th Symp. Logic in Comput. Sci.*, pages 214–225. IEEE, 1991.

[59] Dexter Kozen. On the duality of dynamic algebras and Kripke models. In E. Engeler, editor, *Proc. Workshop on Logic of Programs*, volume 125 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1979.

[60] Dexter Kozen. A representation theorem for models of *-free *PDL*. In *Proc. 7th Colloq. Automata, Languages, and Programming*, pages 351–362. EATCS, July 1980.

[61] Dexter Kozen. On induction vs. *-continuity. In Kozen, editor, *Proc. Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 167–176, New York, 1981. Springer-Verlag.

[62] Dexter Kozen. On Kleene algebras and closed semirings. In Rovan, editor, *Proc. Math. Found. Comput. Sci.*, volume 452 of *Lecture Notes in Computer Science*, pages 26–47, Banska-Bystrica, Slovakia, 1990. Springer-Verlag.

[63] Dexter Kozen. *The Design and Analysis of Algorithms*. Springer-Verlag, New York, 1991.

[64] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Infor. and Comput.*, 110(2):366–390, May 1994.

[65] Dexter Kozen. On action algebras. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, pages 78–88. MIT Press, 1994.

[66] Dexter Kozen. Kleene algebra with tests and commutativity conditions. In T. Margaria and B. Steffen, editors, *Proc. Second Int. Workshop Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *Lecture Notes in Computer Science*, pages 14–33, Passau, Germany, March 1996. Springer-Verlag.

[67] Dexter Kozen. *Automata and Computability*. Springer-Verlag, New York, 1997.

[68] Dexter Kozen. Kleene algebra with tests. *Transactions on Programming Languages and Systems*, 19(3):427–443, May 1997.

[69] Dexter Kozen. On the complexity of reasoning in Kleene algebra. In *Proc. 12th Symp. Logic in Comput. Sci.*, pages 195–202, Los Alamitos, Ca., June 1997. IEEE.

[70] Dexter Kozen. On Hoare logic and Kleene algebra with tests. *Trans. Computational Logic*, 1(1):60–76, July 2000.

[71] Dexter Kozen. On the complexity of reasoning in Kleene algebra. *Information and Computation*, 179:152–162, 2002.

[72] Dexter Kozen. Automata on guarded strings and applications. *Matématica Contemporânea*, 24:117–139, 2003.

[73] Dexter Kozen. Kleene algebras with tests and the static analysis of programs. Technical Report TR2003-1915, Computer Science Department, Cornell University, November 2003.

[74] 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.

[75] 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.

[76] Dexter Kozen and Jerzy Tiuryn. On the completeness of propositional Hoare logic. In J. Desharnais, editor, *Proc. 5th Int. Seminar Relational Methods in Computer Science (RelMiCS 2000)*, pages 195–202, January 2000.

[77] Daniel Krob. A complete system of *B*-rational identities. *Theoretical Computer Science*, 89(2):207–343, October 1991.

[78] Werner Kuich. The Kleene and Parikh theorem in complete semirings. In T. Ottmann, editor, *Proc. 14th Colloq. Automata, Languages, and Programming*, volume 267 of *Lecture Notes in Computer Science*, pages 212–225, New York, 1987. EATCS, Springer-Verlag.

[79] Werner Kuich and Arto Salomaa. *Semirings, Automata, and Languages*. Springer-Verlag, Berlin, 1986.

[80] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 1981.

[81] E. G. Manes and M. A. Arbib. *Algebraic Approaches to Program Semantics*. Springer-Verlag, New York, 1986.

[82] Nikolay Mateev, Vijay Menon, and Keshav Pingali. Fractal symbolic analysis. In *Proc. 15th Int. Conf. on Supercomputing*, pages 38–49. ACM, ACM Press, 2001.

[83] Ernst W. Mayr and Albert Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Adv. Math.*, 46(3):305–329, December 1982.

[84] Kurt Mehlhorn. *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1984.

[85] J. Meseguer and J. A. Goguen. Initiality, induction and computability. In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 460–541. Cambridge University Press, 1985.

[86] G. Mirkowska. *Algorithmic Logic and its Applications*. PhD thesis, University of Warsaw, 1972. in Polish.

[87] B. Möller. Calculating with pointer structures. In R. Bird and L. Meertens, editors, *Algorithmic Languages and Calculi. Proc. IFIP TC2/WG2.1 Working Conference*, pages 24–48. Chapman and Hall, February 1997.

[88] B. Möller. Safer ways to pointer manipulation. Technical Report 2000-4, Institut für Informatik, Universität Augsburg, 2000.

[89] J. Myhill. Finite automata and the representation of events. Technical Note WADD 57-624, Wright Patterson AFB, 1957.

[90] A. Nerode. Linear automaton transformations. *Proc. Amer. Math. Soc.*, 9:541–544, 1958.

[91] K. C. Ng. *Relation Algebras with Transitive Closure*. PhD thesis, University of California, Berkeley, 1984.

[92] K. C. Ng and A. Tarski. Relation algebras with transitive closure, abstract 742-02-09. *Notices Amer. Math. Soc.*, 24:A29–A30, 1977.

[93] H. Nishimura. Sequential method in propositional dynamic logic. *Acta Informatica*, 12:377–400, 1979.

[94] R. J. Parikh. On context-free languages. *J. Assoc. Comput. Mach.*, 13(4):570–581, 1966.

[95] D. L. Pilling. Commutative regular equations and Parikh's theorem. *J. London Math. Soc.*, 6(4):663–666, June 1973.

[96] Damien Pous. Relational algebra and KAT in Coq, February 2013. Available at `http://perso.ens-lyon.fr/damien.pous/ra`.

[97] Damien Pous. Symbolic algorithms for language equivalence and Kleene algebra with tests. In *POPL*, January 2015. To appear.

[98] V. R. Pratt. A practical decision method for propositional dynamic logic. In *Proc. 10th Symp. Theory of Comput.*, pages 326–337. ACM, 1978.

[99] V. R. Pratt. Models of program logics. In *Proc. 20th Symp. Found. Comput. Sci.*, pages 115–122. IEEE, 1979.

[100] V. R. Pratt. Dynamic algebras and the nature of induction. In *Proc. 12th Symp. Theory of Comput.*, pages 22–28. ACM, 1980.

[101] Vaughan Pratt. Dynamic algebras as a well-behaved fragment of relation algebras. In D. Pigozzi, editor, *Proc. Conf. on Algebra and Computer Science*, volume 425 of *Lecture Notes in Computer Science*, pages 77–110, Ames, Iowa, June 1988. Springer-Verlag.

[102] Vaughan Pratt. Action logic and pure induction. In J. van Eijck, editor, *Proc. Logics in AI: European Workshop JELIA '90*, volume 478 of *Lecture Notes in Computer Science*, pages 97–120, New York, September 1990. Springer-Verlag.

[103] M. O. Rabin and D. S. Scott. Finite automata and their decision problems. *IBM J. Res. Develop.*, 3(2):115–125, 1959.

[104] V. N. Redko. On defining relations for the algebra of regular events. *Ukrain. Mat. Z.*, 16:120–126, 1964. In Russian.

[105] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[106] Jacques Sakarovitch. Kleene's theorem revisited: A formal path from Kleene to Chomsky. In A. Kelemenova and J. Keleman, editors, *Trends, Techniques, and Problems in Theoretical Computer Science*, volume 281 of *Lecture Notes in Computer Science*, pages 39–50, New York, 1987. Springer-Verlag.

[107] Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. Assoc. Comput. Mach.*, 13(1):158–169, January 1966.

[108] Arto Salomaa and Matti Soittola. *Automata Theoretic Aspects of Formal Power Series*. Springer-Verlag, New York, 1978.

[109] W. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[110] K. Segerberg. A completeness theorem in the modal logic of programs (preliminary report). *Not. Amer. Math. Soc.*, 24(6):A–552, 1977.

[111] A. Selman. Completeness of calculi for axiomatically defined classes of algebras. *Algebra Universalis*, 2:20–32, 1972.

[112] Ian Stewart. *Galois Theory*. Chapman and Hall, London, 1973.

[113] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proc. 5th Symp. Theory of Computing*, pages 1–9, New York, 1973. ACM.

[114] Robert E. Tarjan. A unified approach to path problems. *J. Assoc. Comput. Mach.*, pages 577–593, 1981.

[115] Walter Taylor. Equational logic. *Houston J. Math.*, pages i–83, 1979. Survey.

[116] V. Trnkova and J. Reiterman. Dynamic algebras which are not Kripke structures. In *Proc. 9th Symp. on Math. Found. Comput. Sci.*, pages 528–538, 1980.

[117] B. L. van der Waerden. *Algebra*, volume 1. Frederick Ungar, 1970.

[118] James Worthington. Automatic proof generation in Kleene algebra. In R. Berghammer, B. Möller, and G. Struth, editors, *10th Int. Conf. Relational Methods in Computer Science (RelMiCS10) and 5th Int. Conf. Applications of Kleene Algebra (AKA5)*, volume 4988 of *Lect. Notes in Computer Science*, pages 382–396. Springer-Verlag, 2008.

Notation and Abbreviations

KA	Kleene algebra	3

Index