Kleene Algebra — Lecture 5

Tobias Kappé

January 2022

1 Today's lecture

In the last lecture, we saw that we can convert any rational expression e into a finite automaton that accepts the language denoted by e. On an intuitive level, this result tells us that we can convert a *program* — i.e., a piece of text with instructions — into a *machine* — i.e., something that can can act out a series of actions in a desired order on our behalf. It is not a stretch of the imagination to think of this procedure as a kind of "compilation": your program is converted into something that could be executed by an electronic circuit.

The idea that rational expressions can be converted into finite automata is known as *Kleene's theorem*, and it is a celebrated and central result in theoretical computer science. It tells us that every language (e.g., sets of sequences of events) that we write down can also be described by a finite automaton. Indeed, Kleene's theorem says something stronger: a (state in a) finite automaton can also be converted back into an equivalent rational expression, thereby showing that rational expressions and finite automata are exactly equal in terms of their expressiveness. Today, we will study this other half of Kleene's theorem.

Returning to the perspective of compilation, you could think of today's inverse construction as "decompilation": we take an abstract machine and obtain a representation of its behavior in terms of code. As usual with techniques like this, the outcome of this conversion may be rather unwieldy if you are not careful. However, the method obtained to achieve this procedure is rather interesting from a theoretical point of view, and will be very useful when we start discussing logical completeness, in the next lecture.

2 Solving equations

Consider the automaton in Figure 1, and suppose you want to find an expression $e \in \mathbb{E}$ such that $[\![e]\!] = L(q_1)$. You could such an e, and try to prove that it meets your expectation. But this is a risky endeavour; your guess could be wrong, so you might be waste your time on a proof that does not go through.

Instead, let's use our knowledge about equivalence between expressions to synthesize the expressions we are looking for, based on the structure of the automaton. That's right, it's time to start *solving equations* using Kleene Algebra. It is useful to first widen our perspective to finding an expression for each state in the automaton of interest. Returning to our example, this means that we are looking for expressions $e_0, e_1 \in \mathbb{E}$ such that $[\![e_i]\!] = L(q_i)$.

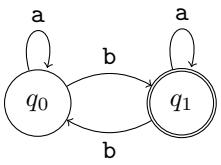


Figure 1: An automaton to be converted into rational expressions.

What are the constraints that we need to put on these expressions? Well, because any word starting with an $\mathtt{a}$ and continuing with a word from $L(q_1)$ is a word in $L(q_1)$, we know that $\mathtt{a} \cdot e_1 \leqq e_1$. Similarly, $\mathtt{b} \cdot e_0 \leqq e_1$. What's more, since q_1 is accepting, it must be the case that $1 \leqq e_1$ — the behavior that accepts immediately is covered by the behavior encoded in e_1. If we carry out this process for q_0 too, we find the following constraints:

$$\mathtt{a} \cdot e_0 \leqq e_0 \qquad \mathtt{b} \cdot e_0 \leqq e_1 \qquad \mathtt{a} \cdot e_1 \leqq e_1 \qquad \mathtt{b} \cdot e_0 \leqq e_1 \qquad 1 \leqq e_1$$

Perhaps more concisely, if we use the fact that $e \leqq f$ and $g \leqq f$ holds if and only if $e + g \leqq f$, we can condense the above down to two constraints:

$$\begin{array}{rcccccc} & & \mathtt{a} \cdot e_0 & + & \mathtt{b} \cdot e_1 & \leqq & e_0 \\ 1 & + & \mathtt{a} \cdot e_1 & + & \mathtt{b} \cdot e_0 & \leqq & e_1 \end{array}$$

Let's see what we can derive about e_0 and e_1 based on the above conditions. For one thing, we can rearrange the second constraint into $(1+\mathtt{b}\cdot e_0)+\mathtt{a}\cdot e_1 \leqq e_1$; by the fixpoint axioms, it then follows that

$$\mathtt{a}^* \cdot (1 + \mathtt{b} \cdot e_0) \leqq e_1$$

Substituting the above into the first constraint will give us the following

$$\mathtt{a} \cdot e_0 + \mathtt{b} \cdot (\mathtt{a}^* \cdot (1 + \mathtt{b} \cdot e_0)) \leqq e_0$$

Using distributivity, associativity and commutativity, we can rearrange this into

$$(\mathtt{b} \cdot \mathtt{a}^*) + (\mathtt{a} + \mathtt{b} \cdot \mathtt{a}^* \cdot \mathtt{b}) \cdot e_0 \leqq e_0$$

By the fixpoint axiom, we get the following *closed* lower bound on e_0:

$$(\mathtt{a} + \mathtt{b} \cdot \mathtt{a}^* \cdot \mathtt{b})^* \cdot \mathtt{b} \cdot \mathtt{a}^* \leqq e_0$$

Peering at this expression on the left, you can probably tell that it is pretty close to the language of q_0: it describes all words that start with a series of words that are either $\mathtt{a}$ or $\mathtt{b}w\mathtt{b}$ with $w \in \{\mathtt{a}\}^*$, followed by a $\mathtt{b}$, followed by a series of $\mathtt{a}$'s. Clearly, each of these words takes you from q_0 to q_1.

What's more, you could make a formal argument that a word that goes from q_0 to q_1 should match this expression, because any path from q_0 to q_1 can be broken up into cycles that go from q_0 to q_0 without visiting q_0 in between, and those are labeled by either $\mathtt{a}$ or $w \in [\![\mathtt{b} \cdot \mathtt{a}^* \cdot \mathtt{b}]\!]$, followed by a phase that goes to q_1 (reading $\mathtt{b}$) and then stays there (reading some number of $\mathtt{a}$'s).

3 Solving automata

Let's generalize the approach that we saw just now to work for all automata. To this end, we need a generic method to condense constraints on the expressions we are looking for from an automaton. The following definition fits that bill.

Definition 5.1 (Solution)**.** Let $A = \langle Q, F, \delta \rangle$ be an automaton. A *solution* to A is a function $s : Q \to \mathbb{E}$, such that for all $q \in Q$ it holds that

$$[q \in F] + \sum_{\mathtt{a} \in \Sigma} \mathtt{a} \cdot s(\delta(q, \mathtt{a})) \leqq s(q)$$

Note that if you unroll the definition above to the automaton in Figure 1, you will find that a solution is exactly a function $s : Q \to \mathbb{E}$ such that

$$\begin{array}{rcrcrcl} 0 & + & \mathtt{a} \cdot s(q_0) & + & \mathtt{b} \cdot s(q_1) & \leqq & s(q_0) \\ 1 & + & \mathtt{a} \cdot s(q_1) & + & \mathtt{b} \cdot s(q_0) & \leqq & s(q_1) \end{array}$$

which matches exactly the constraints set forward in the previous section.

However, having a solution to an automaton is not enough for our purposes. For instance, one solution to the system above would be to set $s(q) = (\mathtt{a} + \mathtt{b})^*$ for all $q \in Q$. Clearly, this is an overestimation, as it includes behavior outside of $L(q_0)$, like ϵ. The following tries to restrict our solutions to be conservative.

Definition 5.2 (Least solution)**.** Let A be an automaton, and let s be a solution to A. We say that s is a *least* solution to A when s is (pointwise) least w.r.t. $\leqq$; i.e., for all solutions s' to A and for all $q \in Q$ it holds that $s(q) \leqq s'(q)$.

With this idea in hand, we can now go forward and show that if we did have a least solution, it would fit our requirement of denotationally describing the languages of states in the automaton. This goes as follows.

Lemma 5.3. *Let $A = \langle Q, F, \delta \rangle$ be an automaton, and let $s : Q \to \mathbb{E}$ be a least solution to A. Then $[\![s(q)]\!] = L(q)$ for all $q \in Q$.*

Proof. To show that for all $q \in Q$, $L(q) \subseteq [\![s(q)]\!]$, we show that for all $w \in \Sigma^*$ and $q \in Q$, if $w \in L(q)$ then $w \in [\![s(q)]\!]$, by induction on w. In the base, where $w = \epsilon$, we have $q \in Q$. In that case, since s is a solution to A, we know that $1 \leqq s(q)$, and hence $\epsilon \in [\![s(q)]\!]$ as well. For the inductive step, let $w = \mathtt{a}w'$, and assume the claim holds for w'. Then, since $\mathtt{a}w' \in L(q)$, it follows that $w' \in L(\delta(q, \mathtt{a}))$. By induction, $w' \in [\![s(\delta(q, \mathtt{a}))]\!]$. Since s is a solution to A, $\mathtt{a} \cdot s(\delta(q, \mathtt{a})) \leqq s(q)$, and thus $[\![\mathtt{a} \cdot s(\delta(q, \mathtt{a}))]\!] \subseteq [\![s(q)]\!]$, meaning that $w = \mathtt{a}w' \in [\![s(q)]\!]$.

For the converse inclusion we need a detour. First, define $s' : Q \to \mathbb{E}$ as:

$$s'(q) = [q \in F] + \sum_{\mathtt{a} \in \Sigma} \mathtt{a} \cdot s(\delta(q, \mathtt{a}))$$

In particular, this means that $s'(q) \leqq s(q)$ for all $q \in Q$, because s is a solution to A. We now claim that s' is a solution to A, as well. To see this, note that

$$[q \in F] + \sum_{\mathtt{a} \in \Sigma} \mathtt{a} \cdot s'(\delta(q, \mathtt{a})) \leqq [q \in F] + \sum_{\mathtt{a} \in \Sigma} \mathtt{a} \cdot s(\delta(q, \mathtt{a})) = s'(q)$$

where we used that $s'(\delta(q, \mathtt{a})) \leqq s(\delta(q, \mathtt{a}))$, as well as monotonicity of all operators w.r.t. $\leqq$. But now, since s is the *least* solution to A, we have that $s(q) \leqq s'(q)$ for all $q \in Q$. In particular, we find for all $q \in Q$ that

$$s(q) \leqq [q \in F] + \sum_{\mathtt{a} \in \Sigma} \mathtt{a} \cdot s(\delta(q, \mathtt{a})) \tag{1}$$

To prove the inclusion, we claim that for all $w \in \Sigma^*$ and $q \in Q$, it holds that $w \in [\![s(q)]\!]$ implies $w \in L(q)$. In the base, where $w = \epsilon$, note that by (1) and soundness, it follows that $q \in F$, and hence $\epsilon \in L(q)$ by definition. For the inductive step, let $w = \mathtt{a}w'$ and assume the claim holds for w'. Again by (1) and soundness, we find that $w' \in [\![s(\delta(q, \mathtt{a}))]\!]$. By induction, we then know that $w' \in L(\delta(q, \mathtt{a}))$, and thus $w = \mathtt{a}w' \in L(q)$. This completes the proof. □

4 Enter the matrix

At this point, we can condense an automaton to conditions on expressions that describe the languages of its states. At least for the example we considered, finding a solution to these conditions is possible. But does this hold in general? It turns out the answer is yes, but we need to take another detour first.

Let's take another look at the solution conditions solution from the example:

$$\begin{array}{lllllll} 0 & + & \mathtt{a} \cdot s(q_0) & + & \mathtt{b} \cdot s(q_1) & \leqq & s(q_0) \\ 1 & + & \mathtt{a} \cdot s(q_1) & + & \mathtt{b} \cdot s(q_0) & \leqq & s(q_1) \end{array}$$

If you have taken a linear algebra course, you might recognize this format as a linear system, and realize that such a system can be written more succinctly using matrices. If you haven't, don't worry — we are about to go through the definitions you need to see what is meant here.

Definition 5.4 (Vectors and matrices)**.** Let S be a set. An *S-vector (over $\mathbb{E}$)* is a function $v : S \to \mathbb{E}$, and an *S-matrix (over $\mathbb{E}$)* is a function $M : S \times S \to \mathbb{E}$.

Let $A = \langle Q, F, \delta \rangle$ be an automaton. We have already seen an example of a vector: a solution to A is a Q-vector. As an example of a matrix, consider the Q-matrix M where each cell is populated by transition labels from q to q':

$$M_A(q, q') = \sum_{\delta(q, \mathtt{a}) = q'} \mathtt{a}$$

Traditionally, we write a vector as a column, and a matrix as a square, with the contents laid out in some fixed order that is clear from our choice of S. For instance, if $Q = \{q_0, q_1\}$ as in the example, we can represent the Q-vector s where $s(q_0) = e_0$ and $s(q_1) = e_1$, as well as the matrix M_A defined above, by

$$s = \begin{bmatrix} e_0 \\ e_1 \end{bmatrix} \qquad\qquad M_A = \begin{bmatrix} \mathtt{a} & \mathtt{b} \\ \mathtt{b} & \mathtt{a} \end{bmatrix}$$

We can define addition of S-vectors, and multiplication of an S-matrix by an S-vector, in a manner analogous to the same definitions in linear algebra.

Definition 5.5 (Operations and equivalence on vectors and matrices)**.** Let S be a set, let s, t be S-vectors, and let M, N be S-matrices. We write $s + t$ and $M + N$ for the pointwise addition of vectors, respectively matrices, as follows:

$$(s+t)(x) = s(x) + t(x) \qquad (M+N)(x,y) = M(x,y) + N(x,y)$$

Furthermore, we write $M \cdot s$ and $M \cdot N$ for the vector, respectively matrix, where

$$(M \cdot s)(x) = \sum_{y \in S} M(x,y) \cdot s(y) \qquad (M \cdot N)(x,y) = \sum_{z \in S} M(x,z) \cdot N(z,y)$$

Lastly, we extend equivalence in a pointwise manner, writing $s \equiv t$ when $s(x) \equiv t(x)$ for all $x \in S$, and $M \equiv N$ when $M(x,y) \equiv N(x,y)$ for all $x, y \in S$. Just like before, we write $s \leqq t$ when $s + t \equiv t$, and $M \leqq N$ when $M + N \equiv N$.

If we represent vectors as columns and matrices as tables, then matrix-vector multiplication works by taking each row of the matrix, and multiplying the i-th element of that row with the i-th element of the vector, summing them all up to get the i-th element of the resulting vector. For instance,

$$\begin{bmatrix} \mathtt{a} & \mathtt{b} \\ \mathtt{b} & \mathtt{a} \end{bmatrix} \cdot \begin{bmatrix} e_0 \\ e_1 \end{bmatrix} = \begin{bmatrix} \mathtt{a} \cdot e_0 + \mathtt{b} \cdot e_1 \\ \mathtt{b} \cdot e_0 + \mathtt{a} \cdot e_1 \end{bmatrix}$$

Matrix-matrix multiplication works similarly, operating on each row of the matrix on the left, and each column of the matrix on the right.

We can now encode the two constraints that we derived from the example automaton into one equivalence of vectors, as follows:

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} \mathtt{a} & \mathtt{b} \\ \mathtt{b} & \mathtt{a} \end{bmatrix} \cdot \begin{bmatrix} e_0 \\ e_1 \end{bmatrix} = \begin{bmatrix} 0 + \mathtt{a} \cdot e_0 + \mathtt{b} \cdot e_1 \\ 1 + \mathtt{b} \cdot e_1 + \mathtt{b} \cdot e_0 \end{bmatrix} \leqq \begin{bmatrix} e_0 \\ e_1 \end{bmatrix}$$

More generally, we can encode the constraints derived from an automaton into one equivalence of vectors, as witnessed by the following lemma.

Lemma 5.6 (Solutions to automata, matrix-style)**.** *Let $A = \langle Q, F, \delta \rangle$ be an automaton, and let M_A respectively b_A be a Q-matrix and Q-vector, given by*

$$M_A(q,q') = \sum_{\delta(q,\mathtt{a}) = q'} \mathtt{a} \qquad b_A(q) = [q \in F]$$

Let s be a solution to A, and suppose t be a least Q-vector (w.r.t. $\leqq$) such that $b_A + M_A \cdot t \leqq t$. Then s and t are the same, up to $\equiv$ — in other words, $s \equiv t$.

Proof. Suppose s is a solution to A (not necessarily the least one). We can directly compute $(b_A + M_A \cdot s)(q)$, to find that

$$\begin{aligned}
(b_A + M_A \cdot s)(q) &= b_A(q) + \sum_{q' \in Q} M_A(q,q') \cdot s(q') && \text{(def. matrix operations)} \\
&= [q \in F] + \sum_{q' \in Q} \Big(\sum_{\delta(q,\mathtt{a}) = q'} \mathtt{a} \Big) \cdot s(q') && \text{(def. } b_A \text{ and } M_A\text{)} \\
&\equiv [q \in F] + \sum_{\mathtt{a} \in \Sigma} \mathtt{a} \cdot s(\delta(q,\mathtt{a})) && \text{(distributivity)} \\
&\leqq s(q) && (s \text{ is a solution to } A)
\end{aligned}$$

hence $b_A + M_A \cdot s \leqq s$. Since t is the least such vector, we have that $t \leqq s$.

For the other direction, suppose t is such that $b_A + M_A \cdot t \leqq t$. We claim that t is a solution to A, as witnessed by the following derivation:

$$\begin{aligned}
[q \in F] + \sum_{\mathtt{a} \in \Sigma} \mathtt{a} \cdot s(\delta(q, \mathtt{a})) &\equiv [q \in F] + \sum_{q' \in Q} \Big(\sum_{\delta(q,\mathtt{a}) = q'} \mathtt{a} \Big) \cdot s(q') && \text{(distributivity)} \\
&= b_A(q) + \sum_{q' \in Q} M_A(q, q') \cdot s(q') && \text{(def. } b_A \text{ and } M_A\text{)} \\
&= (b_A + M_A \cdot s)(q) && \text{(def. matrix operations)} \\
&\leqq s(q) && \text{(premise)}
\end{aligned}$$

Since s is the least solution, it follows that $s \leqq t$.

These arguments apply to the least solution, in particular and the least Q-vector t such that $b_A + M_A \cdot t \leqq t$. Hence $s \leqq t$ and $t \leqq s$, meaning $s \equiv t$. □

So, what do we gain from this characterization? Well, it turns out that we can comptue exactly such a Q-vector. for the sake of later discussion, it is convenient to prove something even more general. First, some notation.

Definition 5.7 (Scalar multiplication)**.** Let S be a set and let s be an S-vector. Furthermore, let $e \in \mathbb{E}$. We write $s ⨟ e$ for the S-vector given by $(s ⨟ e)(x) = s(x) \cdot e$.

Lemma 5.8. *Let Q be a finite set, with M a Q-matrix and b a Q-vector. We can construct a Q-vector s, such that $b + M \cdot s \leqq s$, and furthermore if t is a Q-vector and $z \in \mathbb{E}$ with $b ⨟ z + M \cdot t \leqq t$, then $s ⨟ z \leqq t$.*

Proof. To get an idea, let's look at the special case where Q is a singleton set. Here, M and b contain just one expression, and vector addition and multiplication comes down to adding and multiplying that expression. Thus, we are really looking $s \in \mathbb{E}$ where $b + M \cdot s \leqq s$, and if $t \in \mathbb{E}$ such that $b \cdot z \leqq M \cdot t$, then $s \cdot z$. But we already know exactly such an expression: it's $M^* \cdot b$. After all,

$$b + M \cdot M^* \cdot b \equiv (1 + M \cdot M^*) \cdot b \equiv M^* \cdot b \qquad b \cdot z + M \cdot t \leqq t \implies M^* \cdot b \cdot z \leqq t$$

Our job is to generalize this approach to sets Q that contain *more* elements. To this end, we proceed by induction on (the size of) Q.

In the base, where $Q = \emptyset$, we can simply choose s to be the unique Q-vector $s : \emptyset \to \mathbb{E}$, which satisfies both conditions vacuously. For the inductive step, let $p \in Q$ be our *pivot*, and choose $Q' = Q \setminus \{p\}$. We are going to create a Q'-vector and Q'-matrix, and then use the induction hypothesis to obtain a Q'-vector; next, we will extend this Q'-vector into a Q-vector satisfying our objectives.

Let M' and b' respectively be the Q'-matrix and Q'-vector given by

$$M'(q, q') = M(q, q') + M(q, p) \cdot M(p, p)^* \cdot M(p, q')$$

$$b'(q) = b(q) + M(q, p) \cdot M(p, p)^* \cdot b(p)$$

Now, by induction we obtain an Q'-vector s' such that $b' + M' \cdot s' \leqq s'$, and moreover if t' is a Q'-vector such that $b' + M' \cdot t' \leqq t'$, then $s' \leqq t'$.

We extend s to a Q-vector as follows:

$$s(q) = \begin{cases} s'(q) & q \in Q' \\ M(p,p)^* \cdot \Big(b(p) + \sum_{q' \in Q'} M(p, q') \cdot s'(q') \Big) & q = p \end{cases}$$

We now claim that s satisfies the two constraints in our goal.

- To see that $b + M \cdot s \leqq s$, let $q \in Q$. There are two cases to consider. First, let $q \in Q$; we can derive as follows:

$$\begin{aligned}(b + M \cdot s)(q) &= b(q) + \sum_{q' \in Q} M(q, q') \cdot s(q') \\ &\equiv b(q) + M(q, p) \cdot s(p) + \sum_{q' \in Q'} M(q, q') \cdot s(q') \\ &\equiv b(q) + M(q, p) \cdot M(p, p)^* \cdot \Big(b(p) + \sum_{q' \in Q'} M(p, q') \cdot s'(q')\Big) \\ &\qquad + \sum_{q' \in Q'} M(q, q') \cdot s(q') \qquad (\dagger)\end{aligned}$$

  We consider two cases, based on q.

  - If $q \in Q'$, then $(\dagger)$ can be rearranged into

$$\begin{aligned}&b(q) + M(q, p) \cdot M(p, p)^* \cdot b(p) \\ &\qquad + M(q, p) \cdot M(p, p)^* \cdot \sum_{q' \in Q'} M(p, q') \cdot s'(q') \\ &\qquad + \sum_{q' \in Q'} M(q, q') \cdot s'(q') \\ &\equiv b(q) + M(q, p) \cdot M(p, p)^* \cdot b(p) + \sum_{q' \in Q'} M'(q, q') \cdot s'(q') \\ &\equiv b'(q) + \sum_{q' \in Q'} M'(q, q') \cdot s'(q') \\ &= (b' + M' \cdot s')(q) \leqq s'(q) = s(q)\end{aligned}$$

    where we use the fact that $b' + M' \cdot s' \leqq s'$.
  - Otherwise, $q = p$, then $(\dagger)$ is the expression below, whence

$$\begin{aligned}&b(p) + M(p, p) \cdot M(p, p)^* \cdot \Big(b(p) + \sum_{q' \in Q'} M(p, q') \cdot s'(q')\Big) \\ &\qquad + \sum_{q' \in Q'} M(p, q') \cdot s(q') \\ &\equiv (1 + M(p, p) \cdot M(p, p)^*) \cdot \Big(b(p) + \sum_{q' \in Q'} M(p, q') \cdot s'(q')\Big) \\ &\equiv M(p, p)^* \cdot \Big(b(p) + \sum_{q' \in Q'} M(p, q') \cdot s'(q')\Big) \\ &= s(q)\end{aligned}$$

    In both cases, we find that $(b + M \cdot s)(q) \leqq s(q)$.

- Suppose that t is a Q-vector such that $b \fatsemi z + M \cdot t \leqq t$. In particular, note that this implies that we have

$$\begin{aligned}&b(p) \cdot z + M(p, p) \cdot t(p) + \sum_{q' \in Q'} M(p, q') \cdot t(q') \\ &\equiv b(p) \cdot z + \sum_{q' \in Q} M(p, q') \cdot s(q') \leqq t(p)\end{aligned}$$

By the fixpoint axiom, it then follows that

$$M(p,p)^* \cdot \Big(b(p) \cdot z + \sum_{q' \in Q'} M(p,q') \cdot t(q')\Big) \leqq t(p) \tag{2}$$

We are going to use the induction hypothesis yet again: choose the Q'-vector t' where $t'(q) = t(q)$. By induction, we have that if $b' \fatsemi z + M' \cdot t' \leqq t'$, then $s' \fatsemi z \leqq t'$. To discharge this premise, let $q \in Q'$ and derive as follows:

$$\begin{aligned}
&(b' \fatsemi z + M' \cdot t')(q) \\
&= b'(q) \cdot z + \sum_{q' \in Q'} M'(q,q') \cdot t'(q') \\
&\equiv b(q) \cdot z + M(q,p) \cdot M(p,p)^* \cdot b(p) \cdot z \\
&\qquad + \sum_{q' \in Q'} (M(q,q') + M(q,p) \cdot M(p,p)^* \cdot M(p,q')) \cdot t'(q') \\
&\equiv b(q) \cdot z + M(q,p) \cdot M(p,p)^* \cdot \Big(b(p) \cdot z + \sum_{q' \in Q'} M(p,q') \cdot t'(q')\Big) \\
&\qquad + \sum_{q' \in Q'} M(q,q') \cdot t(q') \\
&\leqq b(q) \cdot z + M(q,p) \cdot t(p) + \sum_{q' \in Q'} M(q,q') \cdot t(q') && \text{(by (2))} \\
&\equiv b(q) \cdot z + \sum_{q' \in Q} M(q,q') \cdot t(q') \leqq t(q)
\end{aligned}$$

Thus, we have for $q \in Q'$ that $(s \fatsemi z)(q) = (s' \fatsemi z)(q) \leqq t'(q) = t(q)$. Finally, to show that $(s \fatsemi z)(p) \leqq t(p)$, we derive that

$$\begin{aligned}
s(p) \cdot z &\equiv M(p,p)^* \cdot \Big(b(p) + \sum_{q' \in Q'} M(p,q') \cdot s'(q')\Big) \cdot z \\
&\equiv M(p,p)^* \cdot \Big(b(p) \cdot z + \sum_{q' \in Q'} M(p,q') \cdot s'(q') \cdot z\Big) \\
&\leqq M(p,p)^* \cdot \Big(b(p) \cdot z + \sum_{q' \in Q'} M(p,q') \cdot t'(q')\Big) \\
&\leqq t(p) && \text{(by (2))}
\end{aligned}$$

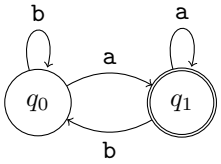
This completes the proof. □

We have achieved our goal: by Lemma 5.8, we can find a vector which, by Lemma 5.6, is a solution to our automaton, and by Lemma 5.3, this vector contains the expressions we are looking for. We can wrap this up as follows.

Theorem 5.9 (Automata to expressions)**.** *Let $\langle Q, F, \delta \rangle$ be an automaton. For each $q \in Q$, we can construct an expression e_q such that $[\![e_q]\!] = L(q)$.*

5 Homework

1. Consider the following automaton:



   (a) Suppose e_0 and e_1 are such that $[\![e_0]\!] = L(q_0)$, and $[\![e_1]\!] = L(q_1)$. Derive the two constraints on e_0 and e_1, just like we did for the expressions that described the languages of the automaton in Figure 1.

   (b) Write the constraints that you derived in the previous exercise as *one* constraint, involving two vectors and one matrix.

   (c) Find lower bounds on the expressions e_0 and e_1 satisfying either the pair of constraints from 1a, or equivalently the constraint from 1b.

   You may use the systematic method from the proof of Lemma 5.8, or the more ad hoc method from Section 2, whichever you prefer.

2. Let Q be a finite set. When M is an Q-matrix and b is an Q-vector, we write $\mathsf{solve}_M(b)$ for the Q-vector such that for all $z \in \mathbb{E}$ and Q-vectors t:

$$b + M \cdot \mathsf{solve}_M(b) \leqq \mathsf{solve}_M(b) \qquad b \fatsemi z + M \cdot t \leqq t \implies \mathsf{solve}_M(b) \fatsemi z \leqq t$$

   Note that such a vector exists, and can be computed, per Lemma 5.8.

   In this exercise, we are going to prove that solve_M is *linear*, in the sense that you may know from linear algebra. Don't worry if you do not know exactly what that means, because we will spell it out.

   (a) Let $z \in \mathbb{E}$. Show that $b \fatsemi z + M \cdot (\mathsf{solve}_M(b) \fatsemi z) \leqq \mathsf{solve}_M(b) \fatsemi z$. Conclude from this, using the properties of solve_M, that

$$\mathsf{solve}_M(b \fatsemi z) \equiv \mathsf{solve}_M(b) \fatsemi z$$

   Hint: for the second part, use that if $e \leqq f \leqq e$, then $e \equiv f$.

   (b) Let b_1 and b_2 be Q-vectors. Prove the following:

      i. $\mathsf{solve}_M(b_1 + b_2) \leqq \mathsf{solve}_M(b_1) + \mathsf{solve}_M(b_2)$
      ii. $\mathsf{solve}_M(b_1) + \mathsf{solve}_M(b_2) \leqq \mathsf{solve}_M(b_1 + b_2)$

   Conclude that $\mathsf{solve}_M(b_1 + b_2) \equiv \mathsf{solve}_M(b_1) + \mathsf{solve}_M(b_2)$.

6 Bibliographical notes

The intuition behind the construction of Lemma 5.8 go back to [Kle56]. The perspective of using matrices over something close to rational terms can be traced back to Conway's monograph [Con71] and Backhouse [Bac75]. Also related is Kozen's treatment of matrices over rational terms [Koz96]; we will derive some facts closer to his results in the next lecture.

References

[Bac75] Roland Backhouse. *Closure algorithms and the star-height problem of regular languages*. PhD thesis, University of London, 1975.

[Con71] John Horton Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, Ltd., London, 1971.

[Kle56] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In Claude E. Shannon and John McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[Koz96] Dexter Kozen. Kleene algebra with tests and commutativity conditions. In *TACAS*, pages 14–33, 1996. doi:10.1007/3-540-61042-1_35.