Kleene Algebras and Algebraic Path Problems

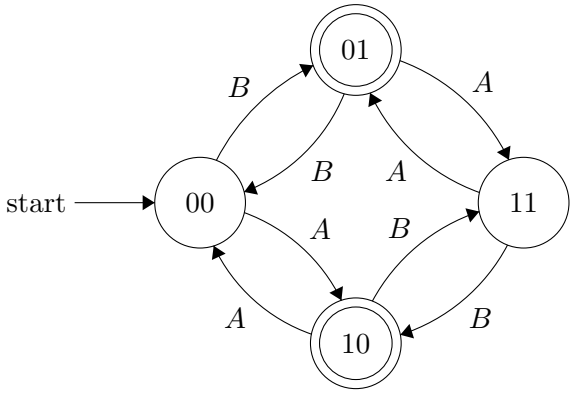
Davis Foote

May 8, 2015

1 Regular Languages

1.1 Deterministic Finite Automata

A **deterministic finite automaton (DFA)** is a model of computation that can simulate any decision algorithm which uses a constant amount of memory [4]. As a brief example, consider the problem of designing a "computer" which operates an overhead light hooked up to two switches A and B. There are four states the system can be in corresponding to whether each switch is on (1) or off (0). Both switches start out at 0 and initially the light is off. Every time a switch is flipped, we want to toggle whether the light is on. This behavior can be summarized in the following diagram:



The two bits correspond to the state of each of switch A and B, and an arrow (edge) from one state u to another state v labeled with the name of a switch s means that if switch s is flipped while the system is in state u, it will transition to state v. The light is on whenever the system is in one of the circled states.

The formal definition of a DFA D is a 5-tuple $D = (Q, \Sigma, \delta, q_0, F)$. Q is the set of states. Σ is the input alphabet, i.e. the set of possible inputs to the automaton (in the above example, $\Sigma = \{A, B\}$). $\delta : Q \times \Sigma \to Q$ is the transition function, which maps a current state and an input to a resulting state. This function provides the edges when a DFA is depicted as a graph (as above). q_0 is the starting state. F is the set of **accepting states** (the states which are circled above) [4].

A DFA then defines a function $D : \Sigma^* \to \{\texttt{accept}, \texttt{reject}\}$. Σ^* is the set of all **strings** over Σ, i.e. the set of all finite-length sequences of elements of Σ including the length-0 **empty string** ϵ. ‘

A DFA $D = (Q, \Sigma, \delta, q_0, F)$ **accepts** a string $s = s_1 s_2 \dots s_n$ if there exists a sequence of states $q_0 q_1 q_2 \dots q_n$ such that $q_i = \delta(q_{i-1}, s_i)$ for all $1 \leq i \leq n$ and $q_n \in F$ [4].

A **language** over a given alphabet is simply a subset of the set of all strings over that alphabet. The language **decided** by a DFA D is the set of all strings which are accepted by D. A **regular language** is any language which can be decided by a DFA.

The following are a few more examples of regular languages. Try to design a DFA that would decide each one (it is sufficient to draw the graph representation):

- $L_1 = \{s \in \{X, Y\}^* |$ The length of s is 2$\}$
- $L_2 = \{s \in \{a, b, c\}^* |$ s contains the substring ‘ab’$\}$
- $L_3 = \{s \in \{0, 1\}^* |$ The number which is represented in binary as s is divisible by 3$\}$

1.2 Nondeterministic Finite Automata

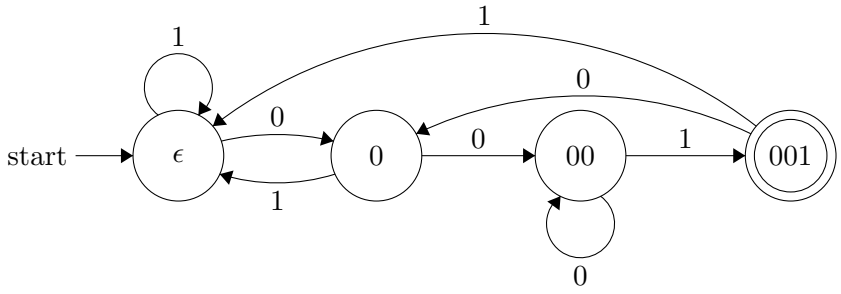
A similar model is a Nondeterministic Finite Automaton (NFA). Informally, an NFA is like a DFA except that it can have multiple transitions out of a single state corresponding to the same symbol, in which case it branches and follows both paths of computation. It can also have no transitions corresponding to a state/symbol pair, in which case that branch of computation is terminated. An NFA accepts an input if and only if at least one of its branches of computation is in an accept state after the input string has been read.

Formally, an NFA N is defined as another 5-tuple, $N = (Q, \Sigma, \delta, q_0, F)$. Q, Σ, q_0, and F are defined as above. $\delta : Q \times \Sigma \to 2^Q$ is the state transition function which maps a state/symbol pair to a *subset* of Q (which could be empty). At any point, the set of all states that some branch of computation is currently in is called the set of **active states** [4].

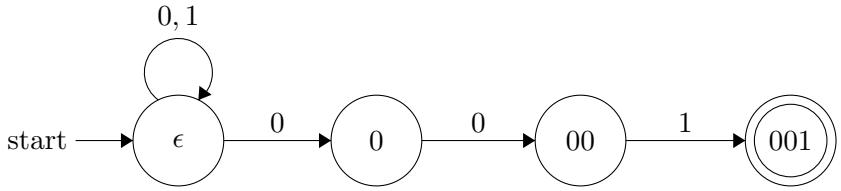
$N = (Q, \Sigma, \delta, q_0, F)$ defines the function $N : \Sigma^* \to \{\texttt{accept}, \texttt{reject}\}$. N accepts a string

$s = s_1 s_2 \dots s_n \in \Sigma^*$ if and only if there exists a sequence of states $q_0 q_1 q_2 \dots q_n$ such that $q_i \in \delta(q_{i-1}, s_i)$ for all $1 \leq i \leq n$ and $q_n \in F$.

A quick motivating example for NFAs is the language L_{001} of binary strings ending in '001'. This is a DFA that decides L_{001}:



Looking at this DFA, it is not immediately clear what it is doing. This NFA that decides the same language is much more concise and comprehensible:



Here, there is a self-loop in the first state corresponding to both symbols in the alphabet, so ϵ is always one of the active states. Every time a '0' is encountered, a new branch of computation begins and '0' becomes one of the active states. If a symbol is encountered that has no corresponding edge in the graph, that branch is terminated. This includes the case that the substring '001' appears before the end of the string; one branch will be in the state '001', at which point another character will follow which will terminate that branch since there are no edges leaving '001'. So the only strings that will be accepted are those that end in '001'.

A very important result in formal language theory is that while NFAs can often decide languages much more concisely than DFAs, they are equivalent in computing power, in the sense that any language which can be decided by a DFA can be decided by an NFA and vice-versa. In other words, **the set of languages which can be decided by NFAs is the set of regular languages** [4]. It is trivial to convert any DFA D into an equivalent NFA N; just change the transition function so that $\delta_N(q, s) = \{\delta_D(q, s)\}$. To find an equivalent DFA D to an NFA N, $Q_D = 2^{Q_N}$, i.e. there is a state in D corresponding to

each subset of the states in N, and the transition function of D corresponds to the change in active states in N given a symbol. $q_{0,D} = \{q_{0,N}\}$ and $F_D = \{q \in Q_D | q \cap F_N \neq \emptyset\}$, i.e. the set of accepting states in D are those which contain at least one accepting state in N.

1.3 Regular Expressions

The most common application of NFAs in modern computer science is matching patterns in a large body of text. **Regular expressions** are a form of notation for languages. Given an alphabet Σ, the set of regular expressions can be defined recursively as follows [4] (where L_R refers to the language denoted by a regular expression R):

1. The empty set $\emptyset$, the set containing the empty string $\{\epsilon\}$, and any set containing exactly one character of Σ are all regular expressions which denote the obvious corresponding language. The set $\{a\}$ for $a \in \Sigma$ or $a = \epsilon$ is usually simply abbreviated to a.

2. Given any two regular expressions R and S, $R \cup S$ is a regular expression denoting the language of strings which are either in L_R or L_S.

3. Given any two regular expressions R and S, $R \circ S$, usually abbreviated to RS, is a regular expression denoting the language $L_{RS} = \{r \circ s | r \in L_R, s \in L_S\}$. The $\circ$ is the **concatenation operator**. For any strings $r = r_1 r_2 \dots r_n, s = s_1 s_2 \dots s_m$, $r \circ s = r_1 r_2 \dots r_n s_1 s_2 \dots s_m$.

4. Given a regular expression R, R^* is a regular expression. This $*$ operation is the same as defined above (called the **Kleene star** [13]). L_{R^*} is the smallest set of strings containing all strings in L_R that is closed under concatenation. A somewhat more useful definition of the Kleene star will be given later in the context of semirings. Some simple examples are $\{01\}^* = \{\epsilon, 01, 0101, \dots\}$ and $\{0, 01\}^* = \{\epsilon, 0, 01, 00, 000, 001, 010, \dots\}$.

$(0 \cup 1)^*001$ is a regular expression that denotes the language from the previous expression of binary strings which end in '001'. Regular expressions for L_1, L_2, and L_3 from section 1.1 are $(X \cup Y)(X \cup Y)$, $(a \cup b \cup c)^*ab(a \cup b \cup c)^*$, and $(0 \cup 1(01^*0)^*1)^*$, respectively. The latter is not at all obvious and deriving it requires use of Kleene's algorithm to convert an NFA to a regular expression, which I will not detail here. A more real-world example is the language of email addresses, which can be built as follows (here a^+ is used as shorthand

for aa^*, the language containing at least one a):

$$\begin{aligned} R_{upper} &= A \cup B \cup \ldots \cup Z \\ R_{lower} &= a \cup b \cup \ldots \cup z \\ R_{number} &= 0 \cup 1 \cup \ldots \cup 9 \\ R_{alphanumeric} &= R_{upper} \cup R_{lower} \cup R_{number} \\ R_{TLD} &= com \cup net \cup org \cup edu \cup gov \\ R_{email} &= (R_{alphanumeric})^+ @ (R_{alphanumeric})^+ . R_{TLD} \end{aligned}$$

As mentioned above, there is an algorithm called Kleene's algorithm which will find an equivalent regular expression to any NFA [9]. Another algorithm called Thompson's construction algorithm will find an equivalent NFA to any regular expression [15]. These two algorithms together prove that **the set of languages denoted by regular expressions is the set of regular languages** [13]. So we have that DFAs, NFAs, and regular expressions are all equivalent in computing ability in the sense that all can decide the same set of languages.

2 Kleene Algebras

2.1 Monoids

A **monoid** is much like a group except that it does not require group elements to have inverses. Specifically, it is a set M endowed with an associative binary operation $\diamond : M \times M \to M$ with an identity element e [6]. A common monoid in computer science is the string monoid, which consists of the set of finite-length strings over an alphabet with the concatenation operator, where the empty string ϵ is the identity string [7].

2.2 Semirings

A **semiring** is much like a ring except that it doesn't require additive inverses. Formally it is a set K together with two operations $+$ and $\cdot$ where $\langle K, + \rangle$ is a commutative monoid with identity 0, $\langle K, \cdot \rangle$ is a monoid with identity 1, multiplication is left- and right-distributive, and for any $x \in K, 0 \cdot x = x \cdot 0 = 0$ [14]. Note that in a normal ring with additive inverses we have that $0 \cdot x = (x - x) \cdot x = x \cdot x - x \cdot x = 0$. In a semiring this does not follow from the first three axioms so it must be an axiom that multiplication by 0 annihilates all elements.

As promised, a more useful definition of the Kleene star can now be given [11]. Given an

element x of a semiring R,

$$x^* = \sum_{k\geq 0} \prod_{i=1}^{k} x \ = \sum_{k\geq 0} x^k = 1 + x + x^2 + x^3 + \dots$$

One interesting example of a semiring is the **tropical semiring** $Tr = \langle R \cup \{\infty\}, \min, +, \infty, 0\rangle$. Here 'addition' is the min operation and 'multiplication' is normal real-number addition [16]. It is easy to see that both operations are associative and commutative. ∞ is the identity w.r.t. min because for any $x \in Tr, \min(x, \infty) = x$. Clearly 0 is the identity w.r.t. normal addition. Addition by ∞ annihilates Tr because for any $x \in Tr$, $x + \infty = \infty + x = \infty$. It is left to the reader to show that addition distributes over min.

2.3 Partially Ordered Sets and Idempotent Operators

A partial order on a set P is a relation $\leq$ that satisfies the following axioms for all $a, b, c \in P$ [12]

- $a \leq a$ (reflexivity)
- $a \leq b \wedge b \leq a \Rightarrow a = b$ (antisymmetry)
- $a \leq b \wedge b \leq c \Rightarrow a \leq c$ (transitivity)

A set coupled with a partial ordering is called a **partially ordered set** or **poset**.

An element a of a set S with a binary operation $+$ is called **idempotent with respect to** $+$ if $a + a = a$. If all elements of S are idempotent with respect to $+$, $+$ is said to be **idempotent** [3].

Any set P with an associative and commutative idempotent operator $+$ implicitly gives a partial ordering [3] $\leq$ defined by

$$a \leq b \Leftrightarrow a + b = b$$

We can check that $\leq$ satisfies the partial ordering axioms:

- Reflexivity: Let $a \in P$. Since $a + a = a$, $a \leq a$.
- Antisymmetry: Let $a, b \in P$ such that $a \leq b$ and $b \leq a$. Then $a + b = b$ and $b + a = a$. $+$ is commutative, so $a + b = b + a$. Therefore $a = b$.
- Transitivity: Let $a, b, c \in P$ such that $a \leq b$ and $b \leq c$. Then $a + b = b$ and $b + c = c$. Since $+$ is associative, $(a + b) + c = b + c = c = a + (b + c) = a + c$. So $a + c = c$ and therefore $a \leq c$.

2.4 Kleene Algebras

A **Kleene algebra** is a set K with two binary operations, $+$ and $\cdot$, and a unary operation *, which satisfy the following axioms for all $a, x \in K$ [8]

(1) $\langle K, +, \cdot \rangle$ is a semiring with additive identity 0 and multiplicative identity 1.

(2) $+$ is idempotent (Note: A semiring in which $+$ is idempotent is called an idempotent semiring).

(3) $1 + aa^* \leq a^*$ (where $\leq$ is the partial ordering given by $+$)

(4) $1 + a^*a \leq a^*$

(5) $ax \leq x \Rightarrow a^*x \leq x$

(6) $xa \leq x \Rightarrow xa^* \leq x$

It should come as no surprise that the set of regular expressions $\mathcal{R}$ is a Kleene algebra with $\cup$ as addition, $\circ$ (abbreviated as juxtaposition) as multiplication, the * operator, $\emptyset$ as 0 and ϵ as 1 [8]. Note that with $\cup$ defining our partial ordering, $A \leq B \Leftrightarrow A \cup B = B \Leftrightarrow A \subseteq B$, so the partial ordering will be referred to as $\subseteq$. We can verify the axioms:

(1) The semiring axioms are verified as follows:

- Set union is associative and commutative and $\emptyset$ is the identity with respect to union.
- Language concatenation is easily seen to be associative (but not commutative). To see that ϵ (shorthand for $\{\epsilon\}$) is indeed the identity language w.r.t. $\circ$, let $R \in \mathcal{R}$. $R\epsilon = \{r \circ e : r \in R \wedge e \in \{\epsilon\}\} = R$ since a string r concatenated with ϵ is just r. Left identity is similarly proved.
- Let $R, S, T \in \mathcal{R}$.

$$\begin{aligned} R(S \cup T) &= \{r \circ a : r \in R \wedge a \in (S \cup T)\} \\ &= \{r \circ s : r \in R \wedge a \in S\} \cup \{r \circ t : r \in R \wedge t \in T\} \\ &= RS \cup RT \end{aligned}$$

  The proof that $(S \cup T)R = SR \cup TR$ is nearly identical.
- Let $R \in \mathcal{R}$. $R\emptyset = \{r \circ a : r \in R \wedge a \in \emptyset\}$. Since there is no such a, $R\emptyset$ is empty and therefore $R\emptyset = \emptyset$. Similarly, $\emptyset R = \emptyset$.

(2) For any set R, $R \cup R = R$, so $\cup$ is idempotent.

(3) Let $R \in \mathcal{R}$. By the definition of the Kleene star, $R \subseteq R^*$ and R^* is closed under concatenation, so $RR^* \subseteq R^*$. Additionally, also by the definition of the Kleene star, the empty string is in R^*, so $\epsilon \subseteq R^*$. Let $a \in \epsilon \cup RR^*$. Then either $a \in \epsilon$ or $a \in RR^*$; in both cases $a \in R^*$. Therefore, $\epsilon \cup RR^* \subseteq R^*$.

(4) The proof that $\forall R \in \mathcal{R} : \epsilon \cup R^*R \subseteq R^*$ is nearly identical to the previous proof.

(5) Let $R, X \in \mathcal{R}$. Suppose $RX \subseteq X$. Then for all $r \in R$ and $x \in X$, $rx \in X$. We want to show that this implies that $R^*X \subseteq X$. Let $r \in R^*$ and $x \in X$. Since $r \in R^*$, r is some sequence of elements of R; call these elements $r_1, r_2, \ldots r_n$ so that $r = r_1 r_2 \ldots r_n$. Then $rx = r_1 r_2 \ldots r_{n-1}(r_n x)$. Since $r_n \in R$ and $x \in X$, by assumption, $r_n x \in X$. Call $x_n = r_n x$. Then $rx = r_1 r_2 \ldots r_{n-2}(r_{n-1} x_n)$. Again by assumption, $r_{n-1} x_n \in X$. A simple induction argument shows that $rx \in X$. Therefore, $R^*X \subseteq X$.

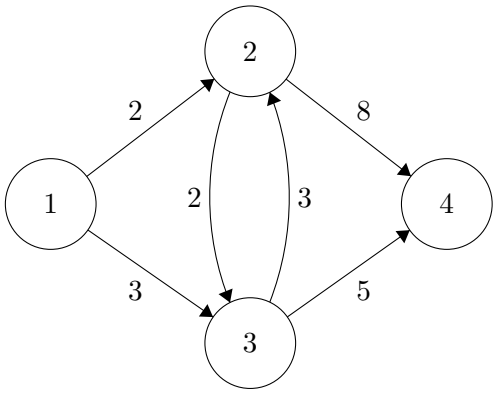
(6) Again, the proof of (6) is nearly identical to that of (5).

Many results can be derived from the axioms that give insights into the algebraic behavior of regular expressions. The main interest among mathematicians in axiomatizing this behavior is that having an algebra for regular languages gives a decision procedure for determining if two regular expressions refer to the same language [8].

3 Algebraic Path Problems

3.1 Weighted Digraphs

A **weighted digraph** (I'll use "graph" interchangeably) $G = (V, W)$ consists of a set of n vertices V and an $n \times n$ **adjacency matrix** W with entries from some idempotent semiring R. W_{ij} refers to the **weight** on the edge from V_i to V_j. If there is no edge from V_i to V_j, $W_{ij} = 0_R$ [2]. For example, we would represent the following graph, with weights from the tropical semiring (note that min is idempotent in Tr), as such:



V	1	2	3	4
1	∞	2	3	∞
2	∞	∞	2	8
3	∞	3	∞	5
4	∞	∞	∞	∞

A **path** from $i \in V$ to $j \in V$ in a graph, denoted $(i \to j)$, is a sequence of vertices from V which starts with i and ends with j. The **weight of a path** from i to j, denoted $w(i \to j)$, is the product of the weights of all edges on the path [2]. More concretely, the weight of a path $v_1 v_2 \dots v_n$ is $\prod_{i=1}^{n-1} W_{i,i+1}$.

3.2 The Algebraic Path Problem

The **algebraic path problem** is the problem of computing the sum of the weights of all paths from i to j, denoted $d(i,j)$ [2].

$$d(i,j) = \sum_{p \in \text{ all } (i \to j)} w(p)$$

Consider the set $M_n(R)$ of $n \times n$ matrices with entries from an idempotent semiring R. Under the operations of normal matrix addition and matrix multiplication with the zero matrix as the additive identity and the identity matrix as the multiplicative identity, $M_n(R)$ is itself an idempotent semiring (verifying this is fairly simple and is left to the reader). An adjacency matrix W of a graph $G = (V, W)$ on n vertices, then, is an element of $M_n(R)$. When looking at successive powers of W, an interesting pattern emerges:

$$\begin{aligned}
(W^2)_{ij} &= \sum_{v \in V} W_{iv} \cdot W_{vj} \\
(W^3)_{ij} &= \sum_{v_1, v_2 \in V} W_{i,n_1} \cdot W_{n_1,n_2} \cdot W_{n_2,j} \\
&\vdots \\
(W^m)_{ij} &= \sum_{v_1, v_2, \dots, v_{m-1} \in V} W_{i,n_1} \cdot \left(\prod_{k=2}^{m-1} W_{n_{k-1},n_k} \right) \cdot W_{n_{m-1},j} \\
&= \sum_{p \in \text{ all length-}m \text{ paths } (i \to j)} w(p)
\end{aligned}$$

In other words, $(W^m)_{ij}$ gives the sum of the weights of all length-m paths from i to j. This means that the sum of the weights of all paths from i to j can be written as the sum of the (i,j)th entry of every power of W. Therefore,

$$d(i,j) = \sum_{m \geq 0} (W^m)_{ij} = \left(\sum_{m \geq 0} W^m \right)_{ij}$$

This looks familiar; in fact, since W is a member of an idempotent semiring, we can apply the Kleene star to it and yield the above expression. This gives the following solution to the algebraic path problem [2]:

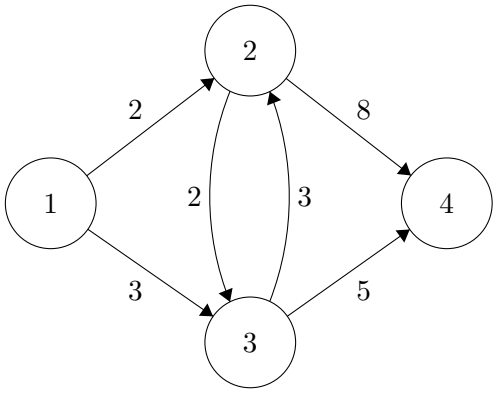
$$\boxed{d(i,j) = (W^*)_{ij}}$$

The reason it is so interesting that the solution to the algebraic path problem can be expressed so succinctly is that this problem is a generalization of several different real problems in computer science. By simply changing the underlying algebra, solutions to the algebraic path problem become solutions to these other problems.

3.3 Shortest Path

The shortest path problem is to find, given a graph $G = (V, W)$ with real-valued costs on the edges and $s, t \in V$, the path with the lowest cost.

Consider the graph from section 3.1 (repeated here for convenience).



V	1	2	3	4
1	∞	2	3	∞
2	∞	∞	2	8
3	∞	3	∞	5
4	∞	∞	∞	∞

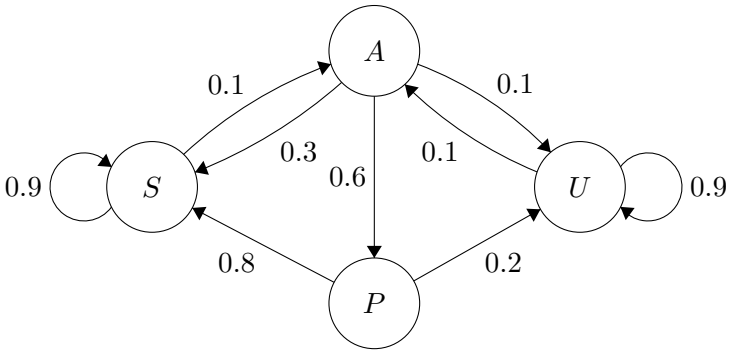
Suppose the edges are one-way roads with differing amounts of traffic so that the weights correspond to the number of minutes required to traverse each road. We are tasked with finding the fastest time we could get from intersection 1 to intersection 4. Using the tropical semiring as our algebra, we have that the solution to the algebraic path problem is

$$\begin{aligned} d(i,j) &= (W^*)_{ij} \\ &= \min_{m \geq 0} (W^m)_{ij} \\ &= \min_{\text{all } (i \to j) \text{ paths}} \sum \text{edge costs in path} \end{aligned}$$

Which is exactly an expression for the cost of the shortest path from i to j.

3.4 Maximum Likelihood in a Markov Chain

A **Markov chain** $M = (V, W)$ is a graph that models a stochastic process. Its vertices are the set of possible states for the system at any point in time, and an edge from $u \in V$ to $v \in V$ has a weight equal to the probability that the system will transition from state u to state v in one time step [1]. This is an example of a Markov chain:



This is the stochastic process of my life. State S is "sleeping". A is "awake in bed". P is "looking at my phone to see if it's worth getting out of bed". U is "up and at 'em". All I know is that I started out awake, and now I am asleep. I want to know the probability of my most likely path to falling asleep (the **maximum likelihood estimate** of my path). We can model this as an algebraic path problem. The **Viterbi semiring** is the semiring $Vit = \langle [0, 1], \max, \cdot, 0, 1 \rangle$. This is the corresponding adjacency matrix W with entries from Vit:

V	S	A	P	U
S	0.9	0.1	0	0
A	0.3	0	0.6	0.1
P	0.8	0	0	0.2
U	0	0.1	0	0.9

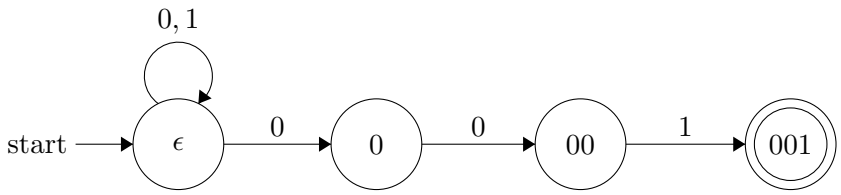
Using the Viterbi semiring as our algebra, we have the following solution to the algebraic path problem:

$$\begin{aligned} d(i, j) &= (W^*)_{ij} \\ &= \max_{\text{all } (i \to j) \text{ paths}} \prod \text{transition probabilities in path} \end{aligned}$$

Which is exactly the probability of the most likely path from i to j.

3.5 Language Decided by an NFA

To tie everything together, we come back to a problem from the first section: how can we tell from an NFA which strings it will accept? Let's look back at an earlier example:



This NFA has exactly one accept state, which makes it easy to model this problem as an algebraic path problem; in general, though, any NFA can be converted to one with only one accept state so that the following setup will work [4]. If we treat the symbols on the transition arrows as elements of a Kleene algebra (which actually gives a more general model of NFAs called GNFAs–"generalized NFAs"–that can have arbitrary regular expressions on their edges) then we have the following adjacency matrix W:

N	ϵ	0	00	001
ϵ	$0 \cup 1$	0	$\emptyset$	$\emptyset$
0	$\emptyset$	$\emptyset$	0	$\emptyset$
00	$\emptyset$	$\emptyset$	$\emptyset$	1
001	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Using a Kleene algebra as our underlying algebra, the solution to the algebraic path problem is [5]

$$\begin{aligned} d(i,j) &= (W^*)_{ij} \\ &= \bigcup_{\text{all } (i \to j) \text{ paths}} \bigcirc \text{ the languages on the transitions of the path, in order} \end{aligned}$$

This is exactly a description of the set of strings accepted by the NFA.

References

[1] D.P. Bertsekas and J.N. Tsitsiklis. *Introduction to Probability*. Athena Scientific books. Athena Scientific, 2002.

[2] Eugene Fink. A survey of sequential and systolic algorithms for the algebraic path problem. *Department of Computer Science, University of Waterloo, 1992. Technical Report CS-92-37*, 1992.

[3] "Encyclopedia of Mathematics". Idempotent. [Online; accessed 8-May-2015].

[4] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 3rd edition, 2012.

[5] Robert Endre Tarjan. A unified approach to path problems. *J. ACM*, 28(3):577–593, July 1981.

[6] Eric W. Weisstein. Monoid. From MathWorld—A Wolfram Web Resource. Last visited on 8/5/2015.

[7] Wikibooks. Haskell/monoids — wikibooks, the free textbook project, 2014. [Online; accessed 8-May-2015].

[8] Wikipedia. Kleene algebra — wikipedia, the free encyclopedia, 2014. [Online; accessed 8-May-2015].

[9] Wikipedia. Kleene's algorithm — wikipedia, the free encyclopedia, 2014. [Online; accessed 8-May-2015].

[10] Wikipedia. Kleene star — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-May-2015].

[11] Wikipedia. Partially ordered set — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-May-2015].

[12] Wikipedia. Regular expression — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-May-2015].

[13] Wikipedia. Semiring — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-May-2015].

[14] Wikipedia. Thompson's construction algorithm — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-May-2015].

[15] Wikipedia. Tropical geometry — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-May-2015].