600.406 — Finite-State Methods in NLP, Part II
Assignment 5: Determinization, Minimization and a Little Learning

Prof. J. Eisner — Spring 2001

Note: This assigment uses the terms "output" and "weight" interchangeably to refer to elements of the semiring K.

1. In any semiring, the **left quotient** $x^i \otimes y$ denotes an arbitrary value v having the property $x \otimes v = y$. (That is, $x \otimes (x^i \otimes y) = y$.) If there is no such value v in the semiring, we say $x^i \otimes y$ is undefined.

   Be careful: x^i is *not* an element of the semiring, the way an inverse element like x^{-1} is. Rather, for each $x \in K$, "$x^i\otimes$" is a prefix operator. However, x^i does behave in many ways like x^{-1} (and the notation "i" is meant to suggest "inverse").

   (a) How can you compute $x^i \otimes y$ in the tropical semiring $(\mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +)$?

   (b) Show that $\underline{0}^i \otimes y$ is undefined for $y \neq \underline{0}$ (in all semirings).

   (c) Give a different example where $x^i \otimes y$ is undefined (specify x, y, and the semiring you are using).

   (d) Show that $\underline{0}^i \otimes \underline{0}$ has multiple values.

   (e) Give a different example where $x^i \otimes y$ has more than one possible value. (Hint: Use the integers modulo 6 under $+, \times$ as your semiring.[1])

   We will say that a semiring **has unique left quotients** if $x^i \times y$ has at most one value for $x \neq 0$.

[1] An **integral domain** is a commutative ring (i.e., semiring with additive inverses and commutative multiplication) in which $\underline{0}$ cannot be written as a product of nonzero factors. It is easy to show that $x^i \otimes y$ has at most one value in an integral domain. The integers modulo any composite number is a standard example of a (semi)ring that is not an integral domain.

(f) The semiring multiplication $\otimes$ has higher precedence than the prefix operator $x^i\otimes$. For example, $x^i \otimes y \otimes z$ is intended to denote $x^i \otimes (y \otimes z)$. Give an example showing that this is not always the same thing as $(x^i \otimes y) \otimes z$.

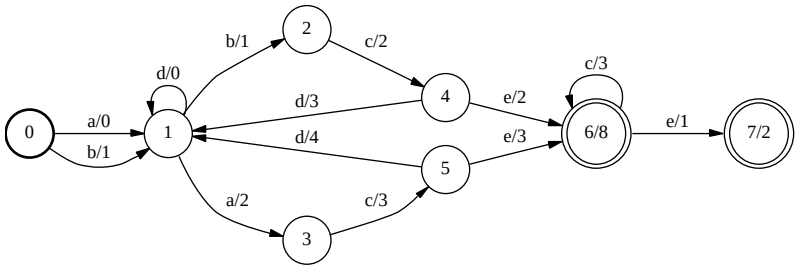
(g) Show, however, that we do still have the associative property in the following sense: *if* $(x^i \otimes y) \otimes z$ is defined, then any value v of this expression is indeed also a value of $x^i \otimes (y \otimes z)$. More generally, the possible values of the first expression form a subset of the possible values of the second expression.

(h) Show that v is a value of $w^i \otimes x^i \otimes y$ if and only if it is a value of $(x \otimes w)^i \otimes y$.

(i) We say that a semiring K **has left inverses**[2] if for all $k \in K$, except $k = \underline{0}$, there exists $k^{-1} \in K$ such that $k^{-1} \otimes k = \underline{1}$. Show that in such a semiring, $x^i \otimes y$ is defined for all $x, y \in K$ except as described in question 1b, and is unique when defined (so that K has unique left quotients).

2. Determinize the following weighted transducer by hand, over the semiring $(\mathbb{R}, +, \times)$. Each state of the resulting machine should be labeled with a *set* of triples of the form (original state, deferred output, deferred weight). (Adapted from an example of Mehyrar Mohri over a different semiring.)

3. This problem considers minimization of the following weighted automaton (adapted from an example of Mohri's), whose weights fall in the tropical semiring $(\mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +)$. The basic procedure is identical to the transducer minimization we did in class. The only difference is in the computation of the "prefix function" $P()$ (question 3a).



(a) For each state q in the machine, compute $P(q)$. This is the output that will be pushed backwards through state q (and perhaps further).
$P(q)$ is a function of the (possibly infinite) set of all paths from q to final states:

[2] This is related to the notion of **division rings**; it might be better to say K is a "left division semiring."

- For the transducer minimization we did in class, each path produced an output string, and $P(q)$ was the longest common prefix (LCP) of all those strings (Mohri (2000)).
- In the present case of the tropical semiring, each path produces an output weight, and $P(q)$ will be the minimum (i.e., semiring sum) of all those weights (Mohri (1997), who writes $d(q)$ rather than $P(q)$ in this case).

(b) "Push" the output weights (this is also known as quasi-determinization) and draw the resulting machine. An output k, on an arc from q to r, should be replaced with $P(q)^i \otimes k \otimes P(r)$. (Recall question 1a.) Check that the pushed machine seems equivalent to the original one.

(c) In the previous question, you should have thought carefully about how to push the outputs of initial arcs and final states. In particular, how did you handle pushing $P(\text{initial state}) \neq \underline{1}(= 0)$? There are at least 3 possible answers to this—can you think of the others?

(d) Minimize the pushed machine using ordinary automaton minimization. (You may want to review footnote 2 and problem 6 of Assignment 1.) First find the equivalence classes of states, showing your work, treating complex arc labels like $a/6$ as if they were single input symbols.[3] Then merge equivalent states and draw the resulting (minimal) automaton.

(e) If you hadn't pushed the outputs first, what states would have been equivalent?

(f) Sometimes changing a single weight will affect whether two states can merge. For example, changing the weight on any arc from state 3 means that nothing can merge (even after pushing). However, changing the weight on any arc from state 1 or 2 still permits the same merges as before. Why the difference?

(g) Mohri (1997) and Mohri (2000) both discuss this algorithm for minimization in the tropical semiring, including the computation of P. Both papers define that semiring as $(\mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +)$.

i. Is the minimal machine always well-defined if negative weights are allowed?

ii. How might the minimization algorithm get into trouble if applied to such a machine? (*Note:* This is a real question—the FSM tools allow negative weights but `fsmminimize` will sometimes halt with an error.)

[3] You can use the easy $O(n^2|\Sigma|)$ algorithm that appears in Hopcroft & Ullman, but be aware that there is an $O(m \log n)$ algorithm where m is the number of edges.

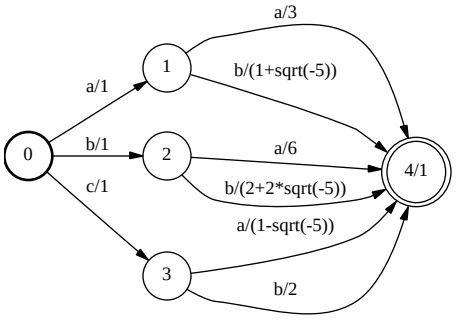
iii. The minimization algorithm does work properly on some machines with negative weights. However, the minimization of such a sequential machine might be subsequential—that is, the minimization might give it final-state weights where it had none before. Give an example.

4. ⋆ Suppose we have a machine whose outputs fall in an arbitrary semiring K. Surprisingly, there might not be a unique minimal equivalent machine! If not, the standard approach to minimization (merging equivalence classes of states) can't work. I haven't seen such cases noticed before.

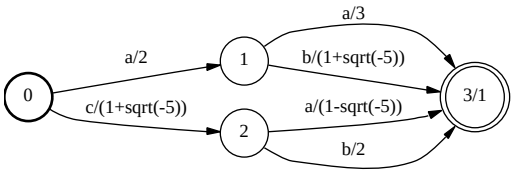
   The semiring we will consider is $\mathbb{Z}[\sqrt{-5}]$ (which is actually a ring). It is defined as $(\{a + b\sqrt{-5} : a, b \in \mathbb{Z}\}, +, \times)$ where $\mathbb{Z}$ denotes the set of integers and $+$ and $\times$ are the usual addition and multiplication of complex numbers. It is a standard example of a commutative ring that differs from the ring of integers $\mathbb{Z}$ in that factorization is not unique.[4] For example, $6 = 2 \times 3 = (1 + \sqrt{-5})(1 - \sqrt{-5})$ and these factors cannot be factored further.

   (a) The following machine takes weights in $\mathbb{Z}[\sqrt{-5}]$. Find two different minimizations. You are not working with anyone's minimization algorithm, so you are on your own! You should figure out how you can push weights back in order to make it possible to merge states, as in the previous problem.

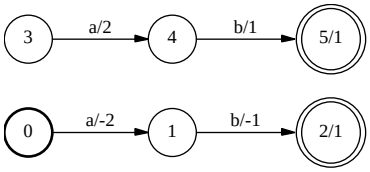
[4] Curious about how to make this claim precise? There are four kinds of elements in a ring: zero ($\underline{0}$), units, composites, and irreducibles. A **unit** is a ring element that has a multiplicative inverse in the ring; in both $\mathbb{Z}$ and $\mathbb{Z}[\sqrt{-5}]$, it's easy to show the only units are 1 and -1. A **composite** is a nonzero element that can be written as a product of 2 or more non-units. Anything else is an **irreducible** (roughly speaking, a prime number). A **Unique Factorization Domain** (UFD) is an integral domain (footnote 1) in which a composite can be factored into irreducibles in at most one way, up to permutation of the factors and multiplication of the factors by units (e.g., in $\mathbb{Z}$, (5)(-3) and (3)(-5) are considered to be the same factorization). The Fundamental Theorem of Arithmetic says precisely that $\mathbb{Z}$ is a UFD. But $\mathbb{Z}[\sqrt{-5}]$ is an integral domain that is *not* a UFD, since 6 has two obviously distinct factorizations into irreducibles.



(b) Why would a minimization algorithm be hard to design for this semiring? (*Hint:* Consider how to define $P()$. What if the machine in 4a had been missing state 3? Or, for that matter, if it had looked like the one below? I think the problem is probably NP-hard by reduction from Minimum Clique Partition, but showing this would require designing a funky semiring.)



(c) ⋆ Your minimizations in 4a compute the same function, so in what sense are they different? There is a one-sentence answer, but be precise and make sure that your definition does *not* consider the following two $\mathbb{Z}$-weighted machines to be different. (*Possible hint*: Footnote 4. *Better hint:* Mohri (2000), Theorem 3.)



(d) The set $Rat(\Sigma)$ of regular languages over $\Sigma = \{x\}$ can be defined as the closure of the set of languages $\{\emptyset, \{\epsilon\}, \{x\}\}$ under union, concatenation, and Kleene star. These languages form a semiring $(Rat(\Sigma), \cup, \cdot)$.

One can define a subsemiring $K \subseteq Rat(\Sigma)$ by considering only the closure of the set $\{\emptyset, \{\epsilon\}, \{xx\}, \{xxx\}\}$ under the same operations.

i. What strings cannot appear in any language in K?

ii. Give an element of K (i.e., a language) that has two different complete factorizations.

iii. Give a K-weighted automaton that has no unique minimization. (*Hint*: Similar to the one in question 4a.)

5. Despite the previous question, good minimization algorithms do turn out to exist for semirings besides the ones that Mohri discusses. They are identical to the algorithm in question 3 except for the definition and computation of the P function.

We'll establish the exact class of semirings where minimization "is possible," and we'll see how to define P to get minimization algorithms for "most" of these semirings. In particular we'll derive a *fast* algorithm for any semiring that has left inverses as in question 1i. Such semirings include the tropical semiring (where our algorithm has some advantages over Mohri's) and any field, including $(\mathbb{R}, +, \times)$ or $(\mathbb{C}, +, \times)$ or the field to be defined in question 6.

Notation: Σ is the alphabet and K is the weight semiring. We assume the input machine is subsequential but not necessarily complete or trim.[5] (It is certainly wise to trim the input machine: this leaves a smaller machine for the later steps to work on. But in the presentation here, I haven't required it because it's not strictly necessary and I wanted to expose what was going on.) The machine's initial state is q_0. If q is a state and $a \in \Sigma$, then the a-labeled arc from q has destination state

[5]Remember that a **subsequential** machine is one whose transition function is deterministic and which may have final-state outputs. (A sequential machine is one where there are no final-state outputs, or equivalently, these outputs are $\underline{1}$. In a p-subsequential machine, there may be up to p different final outputs per state, which allows a little bit of inexpensive nondeterminism.)

A **complete** machine is one whose transition function $\delta(q, a)$ is defined for all states q and symbols a, so there are no "missing transitions." Any machine can be completed, without adding states, by defining the missing transitions $\delta(q, a)$ arbitrarily and with weight $\sigma(q, a) = \underline{0}$.

In a **trim** machine, every state lies on at least one $\underline{0}$-free path from the initial state to a final state. A machine can be trimmed in linear time by eliminating arcs of weight $\underline{0}$ and then using DFS to identify (and eliminate) states that are not accessible from the initial state or co-accessible from any final state. Eliminating such states and arcs may lead to an incomplete machine, which can be completed as above if desired without adding states.

$\delta(q, a)$ and weight $\sigma(q, a)$. We extend this notation in the usual way to allow $\delta(q, w)$ and $\sigma(q, w)$ for $w \in \Sigma^*$.

Such a machine represents a formal power series in $K\langle\langle\Sigma\rangle\rangle$. For minimizing such machines, we will require a function P from $K\langle\langle\Sigma\rangle\rangle \to K$. P is able to "summarize" *any* formal power series as a weight (or at least, any rational one).

So instead of evaluating P on a state q, as in question 3a, we'll evaluate it on $S_q \in K\langle\langle\Sigma\rangle\rangle$. S_q is called the **suffix function** of q. It is defined as the formal power series that the machine would represent if q were the initial state. (Regarded as a function, S_q maps each $w \in \Sigma^*$ to the weight (possibly $\underline{0}$) of its accepting path from q, or $\underline{0}$ if there is no such path. So in effect it describes the set of all accepting paths from q.)

In Mohri's algorithms, $P(S)$ depends only on the range of S—it summarizes just the *weights* of the accepting paths—but we will crucially take the broader view that $P(S)$ is allowed to look at all of S. This also allows a cleaner presentation.

We will require P to have the following sufficient properties (which are more or less necessary—see question 5n and subsequent discussion). We'll show that if such a P exists for the semiring K, and also K has unique left quotients as defined in question 1e, then the basic minimization algorithm is correct. (*Note:* We'll get a more general version in question 5n.)

- The **shifting property**: If $k \in K, S \in K\langle\langle\Sigma\rangle\rangle$, then $P(k \otimes S) = k \otimes P(S)$.
  This is really the crucial property: it will ensure that when we push prefixes back, mergeable states end up with the same suffix function so that we can merge them. Notice that it prevents $P()$ from being a constant function.
- The **quotient property**: If $a \in \Sigma, S \in K\langle\langle\Sigma\rangle\rangle$, then the left quotient $P(S)^i \otimes P(a^{-1}S)$ is defined (see problem 1). Here $a^{-1}S$ is standard notation for the formal power series defined by $(a^{-1}S, w) \stackrel{\text{def}}{=} (S, aw)$.
- The **final-quotient property**: if $S \in K\langle\langle\Sigma\rangle\rangle$, then $P(S)^i \otimes (S, \epsilon)$ is defined. (See question 5l for a simpler alternative.)

The function represented by the input machine is S_{q_0}, the suffix function of the initial state. We wish to find a minimal machine for this function. Just as in question 3, the algorithm works as follows: *pushing* changes the arc weights, then *minimization* merges states by running unweighted FSA minimization as if the arc labels

were atomic, and finally *trimming* removes any useless states (and arcs).[6] No step changes the function S_{q_0} computed by the machine.

Definition of pushing: Given an arc with label $a \in \Sigma$ from q to $r = \delta(q, a)$, pushing replaces its weight $k = \sigma(q, a) \in K$ with the left quotient $P(S_q)^i \otimes P(a^{-1}S_q)$, which is defined by the quotient property.[7] Also, the initial state weight is changed from k to $k \otimes P(S_{q_0})$. And the final weight at q, namely (S_q, ϵ), is replaced with $P(S_q)^i \otimes (S_q, \epsilon)$, which is defined by the final-quotient property.[8]

We now consider minimality. Let $S \stackrel{\text{def}}{=} S_{q_0}$ be the series for which we want to construct a minimal machine. Let $D = \{u \in \Sigma^* : u^{-1}S \neq \underline{0}\}$.[9] For $u, v \in D$, let us write $u \overset{S}{\sim} v$ iff $u^{-1}S = k_u \otimes S'$ and $v^{-1}S = k_v \otimes S'$. That is, $u \overset{S}{\sim} v$ means there exist $k_u, k_v \in K$ and $S' \in \Sigma\langle\langle K \rangle\rangle$ such that we can express $(S, uw) = k_u \otimes (S', w)$ and $(S, vw) = k_v \otimes (S', w)$ for all $w \in \Sigma^*$.

For a machine M and for $u, v \in D$, write $u \overset{M}{\sim} v$ if $\delta(q_0, u) = \delta(q_0, v)$, that is, the prefixes u and v reach the same state. It is easy to see that for *any* subsequential machine that represents S, $u \overset{M}{\sim} v \Rightarrow u \overset{S}{\sim} v$ for all $u, v \in D$. Question 5e asks you to prove this. Suppose one such machine $\bar{M}$ also has the converse property that $u \overset{S}{\sim} v \Rightarrow u \overset{\bar{M}}{\sim} v$ for all $u, v \in D$. Then it is pretty easy to see that $\bar{M}$ (once it

[6] See footnote 5 for the trimming method. But if the input machine was trim to start with, then the output machine will already be trim, except possibly for one dead state that was added at the start of FSA minimization (to complete the FSA), was not merged with anything, and will have to be removed again.

[7] Notice that this equals $P(S_q)^i \otimes P(k \otimes S_r)$ which equals $P(S_q)^i \otimes k \otimes P(S_r)$ by the shifting property. This last version is easier to understand, so that's how we wrote it in question 3.

[8] Since the algorithm assumes that K has unique left quotients, these definitions are unique except for $\underline{0}^i \otimes \underline{0}$. During pushing, we can in fact choose any value for a label $\underline{0}^i \otimes \underline{0}$; these labels just don't show up in our proofs. (The consequence of arbitrary labels will be that the dead states in the pushed automaton may or may not be merged with one another during the minimization step, which is okay because we're going to trim them all anyway.) Note that by question 5c, this case ($P(S_q) = \underline{0}$) arises only when computing the output labels from a dead state q, so it doesn't even arise if we start with a trim automaton (footnote 6).

[9] We restrict our equivalence relations to this set instead of Σ^*, following Mohri, because otherwise we'd have no hope of $\overset{S}{\sim}$ being an equivalence relation: any $u \notin D$ would satisfy $(\forall v) u \overset{S}{\sim} v$ (via $k_u = \underline{0}$). Intuitively, this is because a dead state can merge with any state if we merely ensure (by pushing) that the arcs leading to it have weight $\underline{0}$. In unweighted FSA minimization, this is not possible, but here it would typically prevent $\overset{S}{\sim}$ from being an equivalence relation. Fortunately we can patch $\overset{S}{\sim}$ by limiting it to D and still get a proof that our minimized machine really is minimal.

Conceivably, similar patches might result in efficient algorithms when similar problems arise with other special values of k_u (e.g., nontrivial factors of $\underline{0}$—see footnote 1). But we will take the (slightly fishy) attitude here that semirings are beyond the pale unless they work with the minimally patched version of $\overset{S}{\sim}$ just described: see problem 5m.

is trimmed of its inaccessible and dead states) must be a minimal machine for S; question 5g asks you to prove this.

We'll say that an automaton $\bar{M}$ for S with the above property (that $u \overset{S}{\sim} v \Rightarrow u \overset{\bar{M}}{\sim} v$) is **perfectly minimal**. Our algorithm tries to find a perfectly minimal automaton. (The automaton in question 4a has no perfectly minimal equivalents, so its minimal equivalents would have to be found by some other approach, as noted in question 4b.)

The algorithm takes an input machine M, applies pushing to get M', and applies unweighted FSA minimization to get $\bar{M}$. All three machines represent the same series S (as you will partly show in question 5d).[10] So all that's left to show is that $\bar{M}$ is perfectly minimal (a trimmed version will then be minimal). Let's prove it. Let δ, σ, S_q denote the transition, output weight, and suffix functions in M, and $\delta' = \delta, \sigma', S'_q$ similarly in M'.

Given $u \overset{S}{\sim} v$, we must show $u \overset{\bar{M}}{\sim} v$. We know $u^{-1}S = k_u \otimes S', v^{-1}S = k_v \otimes S'$. Let $q = \delta(q_0, u)$, $r = \delta(q_0, v)$. Claim that $\sigma'(q, a) = \sigma'(r, a)$ for any $a \in \Sigma$. Because left quotients can have multiple values (footnote 8), be careful to read the following equations as saying that any value of an expression is also a value of the next expression:

$$
\begin{aligned}
\sigma'(q,a) &= P(S_q)^i \otimes P(a^{-1}S_q) && \text{(definition of } \sigma' \text{ via pushing)} \\
&= P(S_q)^i \otimes (k^i \otimes k) \otimes P(a^{-1}S_q) && \text{(for any } k \in K) \\
&= P(S_q)^i \otimes k^i \otimes (k \otimes P(a^{-1}S_q)) && \text{(by question 1g)} \\
&= (k \otimes P(S_q))^i \otimes (k \otimes P(a^{-1}S_q)) && \text{(by question 1h)} \\
&= P(k \otimes S_q)^i \otimes P(a^{-1}(k \otimes S_q)) && \text{(by shifting property)} \\
&= P(u^{-1}S)^i \otimes P(a^{-1}(u^{-1}S)) && \text{(instantiating } k \text{ as } \sigma(q_0, u)) \\
&= P(k_u \otimes S')^i \otimes P(a^{-1}(k_u \otimes S')) && \text{(from the definition of } u \overset{S}{\sim} v) \\
&= P(S')^i \otimes k_u^i \otimes (k_u \otimes P(a^{-1}S')) && \text{(running the previous steps backwards)}
\end{aligned}
$$

This last expression has only one value,[11] which can only be $P(S')^i \otimes P(a^{-1}S')$

[10] A subtlety: Unweighted FSA minimization insists that final and non-final states not be merged, but here it is states with different final weights that can't be merged. This can be enforced directly, or else reduced to the unweighted case by a simple transformation that Mohri describes (similar to the one you'll need for problem 5l).

[11] The algorithm assumes that left quotients are unique, but since that only means left quotients by nonzero, seeing uniqueness means checking that $P(k_u \otimes S') \neq 0$. That equals $P(u^{-1}S)$, which by question 5c is nonzero provided that $u^{-1}S \neq \underline{0}$. And we do have $u^{-1}S \neq \underline{0}$ since $u \overset{S}{\sim} v$ and therefore $u \in D$.

by question 1g. By symmetry $\sigma'(r, a)$ equals the same thing, demonstrating the claim. The rest of the proof follows Mohri exactly: It's easy to see that $ua \overset{S}{\sim} va$ (question 5f), so that the previous claim that $\sigma'(q, a) = \sigma'(r, a)$ also applies to show (e.g.) $\sigma'(\delta'(q, a), b) = \sigma'(\delta'(r, a), b)$. By induction, we see that reading any $w \in \Sigma^*$ outputs exactly the same sequence of weights whether we start at q or at r in M'. So regarding M' as an unweighted automaton whose arcs are labeled with (symbol, weight) pairs in $\Sigma \times K$, this means that the strings $(a_1, k_1)(a_2, k_2) \ldots (a_i, k_i) \in (\Sigma \times K)^*$ we can accept from q are exactly those we can accept from r. So unweighted minimization of this automaton will merge states q and r. It follows that $u \overset{\bar{M}}{\sim} v$, and therefore $\bar{M}$ is perfectly minimal. Q.E.D.

Now for the problems:

(a) We saw in question 4 that a $\mathbb{Z}[\sqrt{-5}]$-weighted automaton need not have a perfectly minimal equivalent. So it must be impossible to define P in the semiring $\mathbb{Z}[\sqrt{-5}]$. Show this directly by considering the suffix function S_2 from question 4a; in particular, consider $P(S_2), P(a^{-1}S_2), P(b^{-1}S_2)$, and $P(1)$. (*Hint:* Show that if P has the shifting property then it can't have the quotient property.)

Note: You could still define a reasonable P for this semiring that has the quotient and final-quotient properties without the shifting property. Then pushing would be well-defined, but minimization would not necessarily find a minimal version of the automaton.

> When faced with a question that requires you to manipulate symbols, it often helps to step back and think about what the question means. The quantities $P(S_2)^i P(a^{-1}S_2)$ and $P(S_2)^i P(b^{-1}S_2)$ are supposed to be output weights on the arcs from state 2. To say that they can't be well-defined means that in the minimized machine, there's no consistent way to pick output weights on the arcs from state 2. You should have already discovered this for yourself in question 4a!
>
> It simplifies matters that this semiring is commutative. The contradiction will arise when you try to find some value $P(S_2)$ that divides both $6P(1)$ and $(2 + 2\sqrt{-5})P(1)$, as required by the quotient property, and is itself divisible by both 2 and $1 + \sqrt{-5}$, as required by the shifting property.
>
> Basically, the shifting property makes sure you push back enough weight from the suffix function, and the quotient property makes sure you don't push back too much. In the absence of unique factorizations, these come into conflict.

Not sure what to do with $P(1)$ above? Hint: $P(1)$ divides $(1, \epsilon) = 1$, thanks to the final-quotient property, so in this semiring it must be either 1 or -1. (In general the final-quotient property means $P(1)$ always has a right inverse; problem 5l says that the converse is essentially true too.)

(b) Let δ' and σ' be the transition and output functions of the pushed machine M', whereas δ and σ were the corresponding functions in the original machine M. Prove (by an easy induction on w) that for any state q and for all $w \in \Sigma^*$, $\sigma'(q, w) = P(S_q)^i \otimes P(w^{-1}S_q)$.

(c) Use the previous question and the properties of P to prove that $P(S_q) = \underline{0}$ iff $S_q = \underline{0}$. (The latter "$\underline{0}$" is in $K\langle\langle\Sigma\rangle\rangle$; by the usual rules, it denotes the "empty" power series that maps all strings to $\underline{0} \in K$, i.e., accepts nothing.) In this case q is called a **dead state**.

(d) Use question 5b to prove that the pushed machine M' represents the same series as the original machine M.

(e) Prove what the text earlier said this question would ask you to prove.

(f) Show that if $u \overset{S}{\sim} v$, then $ua \overset{S}{\sim} va$.

(g) Same instructions as question 5e. :-) (*Hint:* Let $\overset{S}{\sim}{}^*$ be the transitive closure of $\overset{S}{\sim}$. Both $\overset{S}{\sim}{}^*$ and $\overset{M}{\sim}$ are equivalence relations; they partition D into n and n_M equivalence classes, respectively. Notice how n_M is related to the machine M, and compare it to n.)

(h) Now let's consider transducers whose outputs are strings in Δ^*. The nuisance is that if we take $\oplus = \cup$ as usual, our weights are sets of strings, and it's hard to see (or compute) what to push. As a simple example, $\{ac, bc, ad, bd\} = \{a, b\} \cdot \{c, d\}$, so $\{a, b\}$ is a prefix that must be pushed back by the shifting property.

However, in the usual case, our input transducer's arcs are weighted with single strings, not sets of strings. And since it is subsequential, the $\oplus$ operation doesn't really matter at all! We don't have to add together the strings on accepting paths for the input because there is at most one such path.

So let's keep $\otimes = \cdot$ (concatenation), but define a new $\oplus$ that lets us deal with string sets of size 0 and 1 only.[12] For an unambiguous machine, this change to

[12] For somewhat different reasons, Mohri defines a similar semiring, the "string semiring," with the difference that $\oplus$ is the longest common prefix operation. However, as Mohri points out, that is only a left semiring (no right distributivity $(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$). So it is less safe to define automata

$\oplus$ doesn't affect the function being computed, or the minimization algorithm. But it does let us define P over a simpler set of weights.

We take our semiring to be $K = (\Delta^* \cup \{\infty\}, \min, \cdot)$. $\underline{0} = \infty$ is a special element that represents "no output" and whose behavior is defined by the rules for $\underline{0}$. Since $\cdot$ is concatenation, $\underline{1} = \epsilon$ as usual. We define $\min(u, v)$ as whichever string is shorter, breaking ties alphabetically according to some fixed arbitrary ordering of the alphabet Δ. Notice that $\min$ gives a total ordering of Δ^* (in fact a well-ordering, assuming that Δ is well-ordered).

i. Prove that K is a semiring and has unique left quotients.

ii. Why couldn't we have simply defined $\min(u, v)$ by alphabetical order?

iii. If $S \in K\langle\langle\Sigma\rangle\rangle$, define $P(S) = \bigwedge_{w \in \Sigma^*}(S, w)$, where $\wedge$ is the longest common prefix operation (with identity ∞). Show that P has the shifting, quotient, and final-quotient properties.

This shows that Mohri's algorithm for minimization of sequential transducers is a special case of our general algorithm.

(i) Now for the punchline. Suppose the weight semiring K has left inverses (hence also unique left quotients) in the sense of question 1i. (This was *not* true in the previous question!)

i. Show that $\overset{S}{\sim}$ is guaranteed to be an equivalence relation.

ii. Suppose we have an ordering of Σ, giving us an ordering on Σ^* exactly as in the previous question.

The **support** of S is the set $\{w : (S, w) \neq \underline{0}\}$. Define $P(S) \overset{\text{def}}{=} (S, \min support(S))$, or $P(S) = \underline{0}$ when $support(S) = \emptyset$. In other words, $P(S)$ finds the least string (according to the above ordering) to which S assigns nonzero weight, if any, and returns that weight.

Show that $P(S)$ has the shifting, quotient, and final-quotient properties.

iii. Why couldn't we have simply defined $\min$ by alphabetical order? (Different answer from question 5(h)ii!)

iv. $\star$ Assume that $\otimes$ runs in constant time. Give a linear-time algorithm to compute $P(S_q)$ at every node q of a K-weighted automaton. (That is, linear on the number of nodes + edges; the automaton might not be complete and any missing edges should save you time.) *Hint*: Use breadth-first search on the reversed transition graph.

over it—although such automata would still define functions, algorithms that assume right distributivity might not work correctly with them.

(j) One semiring with left inverses is the tropical semiring $(\mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +)$. As we've seen, Mohri defines $P(S)$ in this semiring as $\bigoplus_{w \in \Sigma^*}(S, w)$, which finds the minimum-weight path in S. This definition also satisfies all the right properties.

Compare this definition of P to the new definition proposed in question 5i.

i. Which can be computed faster?
ii. Does this speed difference matter asymptotically in the context of the full minimization algorithm?
iii. Which handles negative weights better?
iv. How well does each idea generalize to $(\mathbb{R}_{\geq 0}, +, \times)$ and $(\mathbb{R}, +, \times)$?
v. Do they have the same behavior with respect to question 3(g)iii?
vi. In each algorithm, can the output weights be interpreted as probabilities if the input weights can?

(k) Propose a definition of $P(S)$ for the semiring of integers $(\mathbb{Z}, +, \times)$. Note that this semiring includes negative integers and does not have left inverses.

(l) $\star$ *Extra credit:* The final-quotient property is rather similar to the quotient property. This question exploits that fact and gives a way to avoid bothering with the final-quotient property when choosing P. Here $\Sigma' = \Sigma \cup \{\$\}$, where $\$ \notin \Sigma$ is a new symbol representing "end of string."

Suppose we can find a function $P' : K\langle\langle \Sigma' \rangle\rangle \to K$ that might not have the final-quotient property, but does have the shifting, quotient, and inverse properties:

- The **inverse property**: The particular element $P(\underline{1}) \in K$ has a right inverse $i \in K$. (Remember that by the usual rules, $\underline{1} \in K\langle\langle \Sigma \rangle\rangle$ denotes the power series $\underline{1}\epsilon$ that maps ϵ to $\underline{1} \in K$ and everything else to $\underline{0} \in K$.)

This is more convenient to check than the final-quotient property. In fact, natural definitions of P' tend to have $P'(\underline{1}) = \underline{1}$, which certainly has an inverse.

Show that $P : K\langle\langle \Sigma \rangle\rangle \to K$ defined by $P(S) \stackrel{\text{def}}{=} P'(S\$)$ has the shifting, quotient, and final-quotient properties. (*Hint*: You will have to use problem 1g.)

(m) Now, to return to the broad picture, we'll investigate which semirings K are "nice" in that the sense that they allow perfect minimization.

We've already seen that if there's a pushing function P for $K\langle\langle \Sigma \rangle\rangle$, and K has unique left quotients, then all subsequential $S \in K\langle\langle \Sigma \rangle\rangle$ must have perfectly

minimal representations derived by our algorithm. But here we're concerned with cases where we don't know whether these conditions apply.

We'll start by considering when a *given* series S has a perfectly minimal representation.

i. Show that if S has a perfectly minimal representation $\bar{M}$, then $\overset{S}{\sim}$ is an equivalence relation.

ii. Show the converse: if $\overset{S}{\sim}$ is an equivalence relation, then S has a perfectly minimal representation. No, on second thought, don't show it, I'll just tell you in the footnote.[13]

iii. Suppose $(K, \otimes)$ is any monoid. (For example, any semiring, but we don't need $\oplus$ for this definition.) Let's say that K **has unique left factorizations** iff for any Σ and $x, y \in K$ and $X, Y \in K\langle\langle\Sigma\rangle\rangle$ such that $x \otimes X = y \otimes Y \neq \underline{0}$, we can find $x', y' \in K$ and $Z \in K\langle\langle\Sigma\rangle\rangle$ such that $X = x' \otimes Z, Y = y' \otimes Z$.[14] Show that if K is a semiring with unique left factorization, then $\overset{S}{\sim}$ is an equivalence relation for any $S \in K\langle\langle\Sigma\rangle\rangle$.

iv. Show the converse: If $\overset{S}{\sim}$ is an equivalence relation for any $S \in K\langle\langle\Sigma\rangle\rangle$, then K has unique left factorizations. You should assume $|\Sigma| \geq 2$. (*Hint:* Consider $S = (aa \otimes xX) \oplus (ba \otimes X) \oplus (bb \otimes Y)$ where $a, b \in \Sigma$.)

Putting it all together, we obtain our characterization of "nice" semirings K: all subsequential series in $K\langle\langle\Sigma\rangle\rangle$ have perfectly minimal representations if and only if K has unique left factorization.

There is one exception: we didn't show that K requires unique left factorization if $|\Sigma| < 2$. There is probably a reason for this exception—I suspect that in that case, we are guaranteed perfectly minimal representations regardless of K. (This is certainly true if $|\Sigma| = 0$, and remember that all subsequential automata for $|\Sigma| = 1$ have a characteristic "lollipop" topology: see question 1g on Assignment 1.)

[13] There is a construction of $\bar{M}$ that identifies the states with the equivalence classes of $\overset{S}{\sim}$. It is analogous to a standard construction for unweighted automata. Theorem 1 in Mohri (2000) presents a version for string-to-string transducers, and this can be generalized by replacing Mohri's $g(u)$ with our $P(u^{-1}S)$. Mohri's construction uses the arc label $g(u)^i \otimes g(ua)$, which equals $P(u^{-1}S)^i \otimes P(a^{-1}u^{-1}S)$ and so is defined thanks to the quotient property. Also, his construction must be extended to allow final weights defined similarly as $g(u)^i \otimes (S, u)$, which equals $P(u^{-1}S)^i \otimes (u^{-1}S, \epsilon)$ and so is defined thanks to the final-quotient property.

[14] This definition is stated where X, Y, Z are power series over an arbitrary (perhaps infinite) alphabet. We could replace them in the definition with sequences, indexed sets, or (slightly less obviously) ordinary sets. All of these definitions pick out the same semirings K.

(n) ⋆ Having a "nice" semiring as above only guarantees that perfectly minimal automata are defined. With a slightly stronger condition we can actually obtain them! The method given here does not require unique left quotients, and it does not require us to think of a particular P. (Of course, neither does the construction in question 5(m)ii, but the following construction is much closer to practical because it only needs to consider the set of states, not the set of strings.)

We'll now define a slightly stronger condition (maybe not strictly stronger?) that lets us minimize an automaton without P or left quotients.

As before, $(K, \otimes)$ is a monoid, and lowercase letters denote values in K and uppercase ones in $K\langle\langle\Sigma\rangle\rangle$. We say that Z is a **minimal residue** of S if whenever $S = x \otimes X$, then $(\exists z)X = z \otimes Z$.

If Z is a minimal residue of S, then since $S = \underline{1} \otimes S$, we have $S = z \otimes Z$ for some z. We call such a z a **maximal left factor** of S.

We say that K **has maximal left factors** if every nonzero power series $S \in K\langle\langle\Sigma\rangle\rangle$ has at least one minimal residue and therefore at least one maximal left factor.[15]

Remark: We won't be using P below, but what $P(S)$ does in our original construction is precisely to find a maximal left factor of S.[16] In a semiring without maximal left factors, P can't be defined.

i. Prove that if T is a minimal residue of $k \otimes S$ then it is a minimal residue of S.

ii. Assume from now on that K has maximal left factors. Show that if Z is a minimal residue of S then it is a minimal residue of $k \otimes S \neq \underline{0}$. (*Hint*: Use the previous question.)

iii. Let c be an arbitrary "choice function" $c : 2^{K\langle\langle\Sigma\rangle\rangle} \to K$ that picks out an element of any nonempty collection of power series.[17] Suppose K

[15] Let $\leq$ be the partial ordering of $K\langle\langle\Sigma\rangle\rangle$ by right divisibility. Unique left factorizations says that $(\forall S)(\forall X, Y \leq S)\exists Z \leq X, Y$. Maximal left factors uses a stronger quantifier order: $(\forall S)(\exists Z)(\forall X \leq S)Z \leq X$.

[16] *Proof:* Using the quotient and final-quotient properties and induction on $|w|$, we can see for any $w \in \Sigma^*$, $P(S)$ is a left factor of (S, w). Therefore $P(S)$ is a left factor of S: we can write $S = P(S) \otimes T$ for some T. To show $P(S)$ is actually a maximal left factor, we need to show T is a minimal residue. Given $S = x \otimes X$. Then $P(S) = x \otimes P(X)$ by the shifting property, giving $x \otimes X = x \otimes P(X) \otimes T$. Assuming that K has unique left quotients, we conclude that $X = P(X) \otimes T$. This shows that $P(S)$ is a maximal left factor.

[17] Perhaps by the axiom of choice. Or if we already have well-orderings of Σ and K, then we can easily

has maximal left factors. Define $Z(S) \stackrel{\text{def}}{=} c(\{\text{minimal residues of } S\})$, or $Z(S) = \underline{0}$ if $S = \underline{0}$. Notice that $Z(S)$ divides every minimal residue of S and vice-versa.
Show that $Z(k \otimes S) = Z(S)$ if $k \neq \underline{0}$.

iv. Show that if $u \overset{S}{\sim} v$ then $Z(u^{-1}S) = Z(v^{-1}S)$.

v. Suppose $u \overset{S}{\sim} v$ and M is a machine as usual. Let $q = \delta(q_0, u)$, $r = \delta(q_0, v)$. Show that $Z(S_q) = Z(S_r)$.

vi. In the previous question, show also that $Z(a^{-1}S_q) = Z(a^{-1}S_r)$. (*Hint:* $ua \overset{S}{\sim} va$, and $a^{-1}S_q = \sigma(q, a)S_{\delta(q,a)}$.)

Notation: In general we will write $[X \otimes Y^i]$ to denote an arbitrary element of $\{k : X = k \otimes Y\}$, and $X \otimes Y^i$ to denote the canonical element of that set selected by $c()$.[18] The expected identity holds only of the arbitrary elements: $[X \otimes Y^i] \otimes [Y \otimes Z^i]$ yields $[X \otimes Z^i]$. Another useful identity is $[X \otimes Y^i] = [a^{-1}X \otimes (a^{-1}Y)^i]$ (since $X = k \otimes Y$ implies $a^{-1}x = k \otimes a^{-1}Y$).

We can now redefine the pushing construction directly in terms of Z rather than P:

As usual we define $S = S_{q_0}$. Let q be a state and $a \in \Sigma$. We can write $S_q = x \otimes Z(S_q)$ for some x, hence $a^{-1}S_q = x \otimes a^{-1}Z(S_q)$. Since that is a factorization of $a^{-1}S_q$, by the definition of $Z(a^{-1}S_q)$ we obtain $a^{-1}Z(S_q) = k \otimes Z(a^{-1}S_q)$ for at least one value of k. That means that when pushing, we can define $\sigma'(q, a)$ to equal $a^{-1}Z(S_q) \otimes Z(a^{-1}S_q)^i$, since the choice set is non-empty. We also need to give q_0 an initial weight of $S \otimes Z(S)^i$, which chooses a maximal left factor from S, and to give each state q a final weight of $(Z(S_q), \epsilon)$.

Questions 5(n)v and 5(n)vi ensure that if $u \overset{S}{\sim} v$, then all the arcs from q and r have the same labels in the pushed machine, under these definitions. Similarly q and r have the same final weights. So by the same argument made for the $P()$ algorithm, unweighted FSA minimization will merge q, r and we get $u \overset{\bar{M}}{\sim} v$ as desired.

For correctness of the algorithm, it is only left to show that the pushed machine computes the same function as the original. Following question 5b, we

define a well-ordering of $K\langle\langle\Sigma\rangle\rangle$ (compare two series by the least word where they differ), so let c take the least power series in the set.

[18] It is all right to use c to choose among elements of K since K is embedded in $K\langle\langle\Sigma\rangle\rangle$. Or if K has unique right quotients, then there is only one element to choose from (with the usual $0/0$ exception), and one can find it without a choice function as $(X, w) \otimes (Y, w)^i$ for any convenient w such that $(X, w) \neq \underline{0}$.

can show $\sigma'(q_0, w) = [w^{-1}S \otimes Z(w^{-1}S)^i]$ for any $w \in \Sigma^*$. We can do this by induction on $|w|$. By the identity mentioned above,

$$\sigma'(q_0, w) = [(wa)^{-1}S \otimes a^{-1}Z(w^{-1}S)^i]$$

Let $q = \delta(q_0, w)$. By question 5(n)iii, if $\sigma(q_0, w) \neq \underline{0}$ then

$$\begin{aligned} \sigma'(q, a) &= [a^{-1}Z(S_q) \otimes Z(a^{-1}S_q)^i] \\ &= [a^{-1}Z(\sigma(q_0, w) \otimes S_q) \otimes Z(\sigma(q_0, w) \otimes a^{-1}S_q)^i] \\ &= [a^{-1}Z(w^{-1}S) \otimes Z(a^{-1}w^{-1}S)^i] \end{aligned}$$

(else $\sigma'(q, a)$ is arbitrary). So

$$\sigma'(q_0, wa) = \sigma'(q_0, w) \otimes \sigma'(q, a) = [(wa)^{-1}S \otimes Z((wa)^{-1}S)^i]$$

proving the inductive step. To finish the proof, multiply the path weight $\sigma'(q_0, w)$ by the final weight of q, namely $(Z(S_q), \epsilon) = Z(w^{-1}S, \epsilon)$. This correctly obtains $(\sigma(q_0, w) \otimes Z(w^{-1}S), \epsilon) = (w^{-1}S, \epsilon) = (S, w)$; so the pushed machine agrees with the original.

Closing remark: I think the previous minimization algorithm could be generalized fully, that is, to all semirings with unique left factorizations. We don't really need to insist on *minimal* residues, but only residues that are minimal enough to work with the input automaton. In such semirings, having a common residue is an equivalence relation on the finite set of suffix functions $\{S_q : q \in Q\}$, with $[S_q]$ denoting the equivalence class of S_q, and furthermore each equivalence class $[S_q]$ (being finite) has some common residue $Z([S_q])$. Now just replace $Z(S_q)$ in the above algorithm with $Z([S_q])$ and proceed as before.

Practically speaking, to realize the generalized algorithm for a particular semiring, what we need is a practical **co-pushing** method for finding pairwise common residues: Given two power series expressed in terms of the input machine as $k_1^i \otimes S_q, k_2^i \otimes S_r$ where S_q, S_r are suffix functions, co-pushing finds some m_1, m_2 such that $(k_1 \otimes m_1)^i \otimes S_q$ and $(k_2 \otimes m_2)^i \otimes S_r$ are well-defined and, if possible, equal. (The previous algorithm used maximal residues as a safe way of finding m_1 and m_2 separately rather than jointly.) We can run co-pushing on all state pairs, in any order, to build up left factors that are "maximal enough." If the co-pushing routine can detect when it has achieved equality (found a common residue), then this will build up the equivalence classes as we go. If the co-pushing machine does

not know when it has succeeded in achieving equality, then unweighted FSA minimization can be used at the end to find the equivalence classes, provided that we first canonicalize the pushed weights (i.e., on an arc to state q, replace weight k with the set $\{k' : k' \otimes S_q = k \otimes S_q\}$ or a canonical member of it). Either way, we finish by merging the equivalence classes and trimming, as usual.

To summarize the main results of this problem:

- The semirings where (perfect) minimization is well-defined for $|\Sigma| \geq 2$ are precisely those with unique left factorizations.
- We have exhibited an actual minimization algorithm for all semirings in which maximal left factors can be canonically extracted at each state (by some effective procedure, of course). This is a slightly stronger condition.[19]
- I sketched a possible generalization of the algorithm to *all* semirings with unique left factorizations, i.e., all semirings where perfect minimization is possible.
- Most importantly for practical purposes, we got an especially simple and efficient version of the algorithm in any semiring that has left inverses. This includes all the usual semirings and the one we'll use in question 6.

6. In class, I introduced the V-expectation semiring and sketched how it could be used to run Expectation-Maximization (EM) quite generally on probabilistic string-to-string FSTs, including those with epsilons and interesting parameterizations that are independent of the machine topology.

   This brief problem just asks you to check the semiring properties and develop a couple of intuitions. (I've submitted a workshop paper on the full idea—feel free to ask me for a copy.)

   If V is a vector space over the reals,[20] then the V-expectation semiring is $(\mathbb{R}_{\geq 0} \times V, \oplus, \otimes)$ where

$$(p_1, v_1) \oplus (p_2, v_2) \stackrel{\text{def}}{=} (p_1 + p_2, v_1 + v_2)$$
$$(p_1, v_1) \otimes (p_2, v_2) \stackrel{\text{def}}{=} (p_1 p_2, p_1 v_2 + v_1 p_2)$$

[19] At least, I think it is—I haven't yet thought of any semirings that have unique but not maximal left factorizations. Such a monoid would have to be pretty weird, because one could keep extracting left factors from a power series forever without being able to get them all out "at once."

[20] More generally, V could be any two-sided semimodule over any semiring, not just the reals. For purposes of EM, we usually want the reals in order to represent probabilities, but other representations of probabilities might conceivably be useful, such as confidence intervals.

(a) What are $\underline{0}$ and $\underline{1}$?

(b) Prove that the semiring axioms hold.

(c) Show that the semiring has left inverses in the sense of question 1i. By question 5i, this means that it admits an efficient minimization algorithm.

(d) A semiring that has left inverses is a **field** iff $\otimes$ is commutative (so the inverses are actually two-sided) and the semiring has additive inverses too. Is the V-expectation semiring a field? If so, why? If not, how could it be made into a field?

(A semiring is a field iff the following three properties also hold: $\otimes$ is commutative, all elements have additive inverses, and all elements except $\underline{0}$ have multiplicative inverses.)

(e) Some semirings K are equipped with a "closure operator" * which is a partial function from $K \to K$. It satisfies the axiom $k^* = \underline{1} \oplus k \otimes k^*$. The idea is that $k^* = \bigoplus_{n=0}^{\infty} k^n$.

We need this to sum weights over infinitely many paths. For example, the coefficient of ϵ in the power series $(\epsilon : k)^*$ is k^*, which may or may not be defined. (The answers to assignment 3 will have more on this.)

For example, the usual closure operator for $(\mathbb{R}, +, \times)$ is $p^* \stackrel{\text{def}}{=} (1-p)^{-1}$ and is usually defined for $-1 < p < 1$ only.

What is the appropriate closure operator for $\mathbb{R} \times V$? Show that it satisfies the axiom.

(f) Write concise expressions without using $\otimes$ for the products $\bigotimes_{i=1}^{n}(p_i, p_i v_i)$ and $\bigotimes_{i=1}^{n}(p_i, v_i)$.

(g) Review question 1g of assignment 3. The automaton it proposes describes a Markov process that generates a part-of-speech tag sequence. Put another way, the automaton maps any tag sequence to its probability under a certain bigram model.

Let $V = \mathbb{R}^1$. Suppose you change the weights to fall in the V-expectation semiring, by replacing each weight $p \in \mathbb{R}$ (a probability) with $(p, \langle p \rangle) \in \mathbb{R} \times V$. The following quantities in $\mathbb{R} \times V$ now have useful interpretations: give them.

i. The total weight (using $\oplus$) of all accepting paths in the automaton, A.

ii. The total weight (using $\oplus$) of all accepting paths in $A \cap (\Sigma^* \texttt{VBD} \Sigma^*)$, where VBD is the tag for a past-tense verb.

(h) The previous two questions are part of the inspiration for a different version of the semiring, $K = \{0\} \cup (\mathbb{R}_{>0} \times V)$. There is a natural mapping f from the V-expectation semiring to K:

$$(p, v) \xmapsto{f} \begin{cases} [p, p^{-1}v] & \text{if } p \neq 0 \\ 0 & \text{if } p = 0 \end{cases}$$

(To avoid confusion, we use (p, v) to denote elements of the V-expectation semiring and $[p, v]$ to denote elements of K.)

Define $\oplus$ and $\otimes$ on K so that f is a homomorphism of semirings. That is, your definitions should ensure $f(a \oplus b) = f(a) \oplus f(b)$, $f(a \otimes b) = f(a) \otimes f(b)$, $f(\underline{0}) = \underline{0}$, $f(\underline{1}) = \underline{1}$. Also, give expressions for $[p, v]^{-1}$ and $[p, v]^*$.

It turns out that for EM, we wish to carry out a long computation in the V-expectation semiring and then apply f to the result. But because f is a homomorphism, we could get the same answer if we applied f to the inputs first and then carried out the computation in K. Which operations are easier to compute in which semiring?

7. We discussed several methods for learning FSA topology. All of them relied on greedy state merging:

   - Angluin's reversible-language learner merged states with a common suffix.
   - The Abbadingo heuristics (Lang, Perlmutter, & Price (1998)) merged states that agreed on the classification of many suffixes without disagreeing on any.
   - Stolcke and Omohundro's (1994) model merging carried out whichever merge caused the least decrease in the training perplexity.

   Angluin's learner uses *no* negative evidence; instead, it is conservative and posits the smallest language in the class that is consistent with the data. Abbadingo relies on *explicit* negative evidence. Model merging is statistical (learns from a sample), so it can learn from *implicit* negative evidence (non-occurrences).

   Sketch an approach to learning when both implicit and explicit negative evidence are available. The explicit negative evidence might be a list of excluded forms, or a sample from an "anti-language," or something else—up to you. Describe both your proposed learning method and the input data it uses.