Recall: given a weighted automaton $A$, Support $(\mathcal{L}(A)) = \{ w \mid \mathcal{L}(A)(w) \neq 0 \}$

Non-emptiness Problem for Support Languages:

Input: Weighted Automaton $A$ over $\Sigma$ and $\Sigma$

Question: Is Support $(\mathcal{L}(A)) \neq \emptyset$?

We return to the matrix representation for $A = \langle \lambda, \mu, \gamma \rangle$

$\lambda$ - $1 \times n$ row vector $\in \mathbb{R}^n$

$\mu$ - $\Sigma \rightarrow (m \times n$ real matrix over $\mathbb{R})$

$\gamma$ - $n \times 1$ column vector

Can think of $\mu a$ for $a \in \Sigma$ as a weight distortion.

With each word, the automaton $A$ associates a vector in $\mathbb{R}^n$, namely the vector $\lambda \cdot \mu(w)$ with the word $w$. Here, $\mu(w)$ is defined as in LECTURE 2.

For the sake of deriving the results that will follow, we will assume that $\mathbb{S}$ is a field, i.e., inverses of $+$ and $\times$ exist in $\mathbb{S}$.

Define the set of Reachable Vectors in $\mathbb{S}^n$ as follows:

$$\lambda \mu(\Sigma^*) = \{ \lambda \cdot \mu(w) \mid w \in \Sigma^* \} \subset \mathbb{S}^n$$

We are interested in the span of reachable vectors in $\mathbb{S}^n$, denoted $\langle \lambda \cdot \mu(\Sigma^*) \rangle$.

This is a subspace of $\mathbb{S}^n$ and we may compute its basis.
The following algorithm computes a basis for \( \langle \lambda \mu (\Sigma^k) \rangle \)

1. \[ B \leftarrow \{ \lambda \} \]
2. \[ \text{Todo} \leftarrow \{ \lambda \} \]
3. While \( \text{Todo} \neq \emptyset \):
   - Remove \( x \) from \( \text{Todo} \)
   - If \( x \mu (a) \notin \langle B \rangle \),
     - Add \( x \mu (a) \) to \( B \)
     - Add \( x \mu (a) \) to \( \text{Todo} \)
   - If \( x \mu (b) \notin \langle B \rangle \),
     - Add \( x \mu (b) \) to \( B \)
     - Add \( x \mu (b) \) to \( \text{Todo} \)
4. Output \( B \)

Now, we argue the correctness of the algorithm above.

- We observe that the set \( B \) is always linearly independent. This can be established as a loop invariant. This proves that \( B \) would be linearly independent if the algorithm halts.

- Since \( B \) is always linearly independent, there may be at most \( n \) additions to \( \text{Todo} \). After another \( n \) iterations, \( \text{Todo} \) must be empty. Therefore, the algorithm terminates.

Suppose the algorithm returns \( B = \{ x_1, \ldots, x_m \} \). We argue, by induction, that \( \lambda \mu (w) \notin \langle B \rangle \), for all \( w \in \Sigma^k \).

**Base Case** \( \lambda \mu (\varepsilon) = \lambda \) is added to \( B \) manually in the beginning.

**Inductive Step** Suppose \( \lambda \mu (w) \notin \langle B \rangle \) and in particular \( \lambda \mu (w) = \sum_{i=1}^{k} x_i \cdot x_i \).

Thus, \( (\lambda \mu (w) \cdot \mu (a)) = \sum_{i=1}^{k} x_i \cdot x_i \cdot \mu (a) \).
But the algorithm in fact that \( \lambda_i = \lambda\mu(2_i) \) for some \( 1 \leq i \leq n \).

So,

\[
(\lambda\mu(2_i)) \cdot \mu(a) = \sum_{i=1}^{k} \alpha_i (\lambda_i \mu(a))
\]

\[
= \sum_{i=1}^{k} \alpha_i \lambda_i \mu(2_i a)
\]

---

Now, either \( \lambda_\mu(2_i a) \in B \), in which case we are done.

Otherwise \( \lambda_\mu(2_i a) \notin B \). But this means we know that \( \lambda(2_i) \in B \).

So, \( \lambda(2_i) \) was an element of \( T \) at some point. But \( \lambda_\mu(2_i a) \) was not added to \( B \). This means \( \lambda_\mu(2_i a) = \sum_{i=1}^{k} \beta_i \lambda_i \).

So, (1) can be rewritten as

\[
\sum_{i=1}^{k} \alpha_i \lambda_i \mu(2_i a) = \left( \sum_{i=1}^{k} \beta_i \lambda_i \right) \lambda(2_i a)
\]

but this sum can be expressed as a sum of linear combinations of elements in \( B \).

A similar argument holds for \( \lambda_\mu(2_6) \).

Q.E.D.

Checking whether a given vector \( v \) is in the span of \( B \) can be done using Gaussian Elimination which requires \( O(n^3) \) operations.

Thus, the algorithm terminates in \( O(|E| \times n \times n^2) \) time.

Claim: Suppose \( B = \{ x_1, \ldots, x_k \} \) is a basis of \( \langle \lambda\mu(\Sigma^*) \rangle \) obtained by the algorithm above. Then \( \text{Support} (\langle \lambda, \mu, y \rangle) \) is empty iff \( \lambda_i : y = 0 \) for each \( i \).

Proof:\n
\((\Rightarrow)\) Given any word \( w \in \Sigma^* \), \( \lambda\mu(w) \) can be realized as a linear combination of \( B \), say \( \sum_{i=1}^{k} \alpha_i x_i \). Therefore, \( \lambda\mu(w) y = \sum \alpha_i x_i y_i \).

Thus \( w \notin \text{Supp}(\langle \lambda, \mu, y \rangle) \).
So, we have a polynomial time algorithm for checking the non-emptiness of the support language.

**Equivalence Problem for Weighted Automata**

**Input:** \( \mathcal{A}_1, \mathcal{A}_2 \) - weighted automata

**Question:** Do \( \llbracket \mathcal{A}_1 \rrbracket = \llbracket \mathcal{A}_2 \rrbracket \)?

Observe that if \( \mathcal{A}_1 = \langle \lambda_1, \mu_1, \nu \rangle \) and \( \mathcal{A}_2 = \langle \lambda_2, \mu_2, \nu_2 \rangle \), we may consider

\[
\mathcal{A} = \langle \lambda, \mu, \nu \rangle \quad \text{with} \quad \lambda = \begin{bmatrix}
\lambda_1 \\
\vdots \\
\lambda_2
\end{bmatrix}
\]

\[
\mu(a) = \begin{bmatrix}
\mu_1(a) & 0 \\
0 & \mu_2(a)
\end{bmatrix}
\]

for \( a \in \Sigma \)

\[
\nu = \begin{bmatrix}
\nu_1 \\
\nu_2
\end{bmatrix}
\]

This is simply the disjoint union of \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) with regarded initial weights for \( \mathcal{A}_2 \), \( \llbracket \mathcal{A} \rrbracket \) then computes \( \llbracket \mathcal{A}_1 \rrbracket - \llbracket \mathcal{A}_2 \rrbracket \).

Thus, to answer the equivalence problem, we may construct \( \mathcal{A} \) as above and check if \( \text{Support}(\llbracket \mathcal{A} \rrbracket) = \emptyset \).

**Exercise**

Assume that \( \mathbb{R} \) is an ordered field. We say that \( \llbracket \mathcal{A}_1 \rrbracket \preceq \llbracket \mathcal{A}_2 \rrbracket \) if \( \llbracket \mathcal{A}_1 \rrbracket (w) \preceq \llbracket \mathcal{A}_2 \rrbracket (w) \), read as \( \mathcal{A}_1 \) is dominated by \( \mathcal{A}_2 \). Given \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \), check algorithmically if \( \mathcal{A}_1 \) dominates \( \mathcal{A}_2 \).
The arguments above hold if $S$ is embedded in some field.

For example, $\mathbb{N}$ can be embedded in $\mathbb{Q}$ and so can be $\mathbb{Z}$.

In general, this may not be possible. For example, consider $(\mathbb{N} \cup \{\infty\}, \min, +)$.

Suppose we could embed this in a ring with elements $n'$ for each $n \in \mathbb{N}$ s.t.

\[ \min(n, n') = \infty. \]

Then, we could get a contradiction as follows:

\[ \min(1, 2) = \min(1, 3) \]

\[ \Rightarrow \min(1', \min(1, 2)) = \min(1', \min(1, 3)) \]

\[ \Rightarrow \min(\min(1, 1'), 2) = \min(\min(1, 1'), 3) \]

\[ \Rightarrow \min(\infty, 2) = \min(\infty, 3) \]

\[ \Rightarrow 2 = 3 \]

Thus, in general, cancellativity is necessary for a semiring to be embedded in a field. In the semiring $(\{0, 1\}, \oplus, \cdot)$, we have $0 \cdot 1 = 1 \cdot 1 = 1$.

Hence, these arguments do not hold. Indeed, checking equivalence for NFA is PSPACE-complete.

However, if a semiring can be embedded in a ring without any zero-divisors, (i.e., the ring maintains $m \cdot n = 0 \Rightarrow m = 0 \lor n = 0$), then it may be embedded in its corresponding field of fractions.