Imagine you are given an automaton $M$, and you want to decide something about the language $L(M)$ it recognizes.

For example, you might want to know if $L(M) = \emptyset$, or if $L(M)$ is infinite.

Or, given $M_1$ and $M_2$, you might want to know if $L(M_1) = L(M_2)$.

We are interested in efficient (to the extent possible!) methods for deciding these questions.
There are two different models we could consider.

In the *clear box* model, we are given *complete information* about $M$.

We know all its parts $(Q, \Sigma, \delta, q_0, F)$ explicitly.

The name *clear box* comes from imagining $M$ as an electronic circuit where all components are completely visible to the analyst.
In the *black box* model, we are only given very limited information about $M$, and we can only use the automaton as a “black box” where we feed in inputs to $M$ and see if they are accepted or not.

We are told what its input alphabet $\Sigma$ is, and we are told what the number of states $n = |Q|$ is.

But nothing else.

We do not know $\delta$ or $F$.

This model was first proposed by Edward F. Moore in his 1956 paper “Gedanken-experiments on sequential machines”.
Not surprisingly, in the clear box model, where we have complete information about $M = (Q, \Sigma, \delta, q_0, F)$, there are efficient algorithms to decide these questions, in most cases.

These algorithms are based on considering the transition diagram of the automaton as a labeled, directed graph $G$, and then using standard graph algorithms to search $G$.

Let’s start with answering questions about DFA’s.
Deciding if $L(M) = \emptyset$ in the clear box model

Here we look at the corresponding digraph $G$ based on $M$: nodes are states, transitions are edges. The source is the start state and the sinks are the final states.

If $M$ accepts some string, then there must be a path from the start state $q_0$ to a final state of $F$.

Checking for the existence of such a path is just graph reachability, which can be solved in linear time (in the number of nodes and edges) using depth-first search (DFS) or breadth-first search (BFS).

Since the transition diagram of a DFA of alphabet size $k$ and state size $n$ has $kn$ edges and $n$ nodes, this check is $O(n)$ or linear time if $k$ is fixed (which is the usual convention).
Deciding if $L(M)$ is infinite in the clear box model

Here again we look at the corresponding digraph $G$ based on $M$.

We claim that $L(M)$ is infinite iff there is a *fruitful cycle* in $G$; that is, a cycle $C$ that has the property that some node in the cycle is reachable from $q_0$, and from which one can reach some final state.

To see this, note that if there is a fruitful cycle, we can go around it as many times as we want, getting longer and longer strings accepted by $M$.

So $L(M)$ is infinite.
Deciding if $L(M)$ is infinite in the clear box model

On the other hand, suppose $M$ has $n$ states and $L(M)$ is infinite.

Since $L(M)$ is infinite, we know $M$ accepts some string of length $\geq n$.

By the proof of the pumping lemma there exists a state $q$ and $u, v, w$ such that $\delta(q_0, u) = q$, $\delta(q, v) = q$, and $\delta(q, w) \in F$.

Let $\delta(q_0, u) = q$. Then the cycle of states reached by starting at $q$ and reading $v$ is fruitful.
Deciding if $L(M)$ is infinite in the clear box model

To get an efficient algorithm, start with the transition diagram graph $G$.

Use DFS or BFS to identify all states not reachable from $q_0$, and remove them.

Next, use DFS or BFS to identify all states from which one cannot reach a final state, and remove them.

This can be done in linear time using DFS or BFS. (Think about it; there is an easy trick to avoid doing a DFS/BFS for each final state.)

Any cycle that remains will be fruitful.

Then we can detect the existence of a cycle using DFS, in linear time.
Deciding whether $L(M_1) = L(M_2)$ in the clear box model

Given two DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, of $m$ and $n$ states respectively, we can check if $L(M_1) = L(M_2)$ in $O(mn)$ time by reducing it to the problem of checking if $L(M) = \emptyset$.

But for which $M$?

We use the fact that $L_1 \Delta L_2 = \emptyset$ iff $L_1 = L_2$.

Recall that $L_1 \Delta L_2 := (L_1 - L_2) \cup (L_2 - L_1)$.

So given $M_1$ and $M_2$ we just construct the automaton of $mn$ states recognizing $L(M_1) \Delta L(M_2)$ and use the algorithm we already saw for checking if $L(M) = \emptyset$. 
Deciding whether \( L(M_1) = L(M_2) \) in the clear box model

What is this automaton \( M = (Q, \Sigma, \delta, q_0, F) \)?

We saw it already in Week 3.

On input \( x \), \( M \) simulates the automata \( M_1 \) and \( M_2 \) in parallel, using the state set \( Q_1 \times Q_2 \).

The first component of states simulates \( M_1 \) and the second component simulates \( M_2 \).
Deciding whether $L(M_1) = L(M_2)$ in the clear box model

Formally we define

- $Q = Q_1 \times Q_2$;
- $q_0 = [q_1, q_2]$;
- $F = F_1 \times (Q_2 - F_2) \cup (Q_1 - F_1) \times F_2$;
- $\delta([p, q], a) = [\delta_1(p, a), \delta_2(q, a)]$.

One can now prove by induction on $|x|$ that

$\delta^*(q_0, x) = [\delta_1^*(q_1, x), \delta_2^*(q_2, x)]$ and hence $x$ is accepted by $M$ iff $x \in L(M_1) \Delta L(M_2)$.

The bottom line: in the clear box model, we can decide if $L(M_1) = L(M_2)$ in $O(mn)$ time, if $M_1$ has $m$ states and $M_2$ has $n$ states.
As it turns out, the algorithms we gave above for DFA’s for checking whether \( L(M) = \emptyset \) or \( L(M) \) infinite also work for NFA’s, pretty much unchanged. (Why?)

However, for the third problem, checking whether \( L(M_1) = L(M_2) \) for NFA’s, we need a new algorithm.

It turns out this problem is PSPACE-complete, which means that nobody currently knows an efficient algorithm, and probably there is not one.

(Exercise: why does the construction we gave for DFA’s fail in this case? And how would you check \( L(M_1) = L(M_2) \) for NFA’s?)
Now we turn to black box algorithms for these problems.

How do we check if $L(M) = \emptyset$ in the black box model?

Here we use the following theorem:

**Theorem.** Suppose $M$ is a DFA with $n$ states. Then $L(M) \neq \emptyset$ iff $M$ accepts a string of length $< n$. 
Theorem. Suppose $M$ is a DFA with $n$ states. Then $L(M) \neq \emptyset$ iff $M$ accepts a string of length $< n$.

Proof. One direction is clear.

For the other direction, assume $L(M) \neq \emptyset$, and let $z$ be a shortest string accepted.

If $|z| < n$ we’re done, so assume $|z| \geq n$.

But if $|z| \geq n$, we can apply the pumping lemma to it, obtaining some decomposition $z = uvw$ with $|v| \geq 1$ such that $uv^i w \in L(M)$ for all $i \geq 0$.

Choose $i = 0$. 
Checking if $L(M) = \emptyset$ in the black box model

Then $uw \in L(M)$ and $|uw| = |z| - |v| < |z|$, a contradiction (because $z$ was a shortest string accepted).

To use this theorem, we feed our black box with every possible string of length $< n$.

If it accepts at least one string, then $L(M) \neq \emptyset$.

Otherwise $L(M) = \emptyset$.

This gives an algorithm, but not an efficient one!
Checking if $L(M)$ is infinite in the black box model

Here we use the following theorem:

**Theorem.** Suppose $M$ is a DFA with $n$ states. Then $L(M)$ is infinite iff $M$ accepts a string $z$ with $n \leq |z| < 2n$.

**Proof.** Just as in the previous proof, if $M$ accepts a $z$ with $|z| \geq n$, then the pumping lemma implies that there are $u, v, w$ with $|v| \geq 1$ such that $uv^i w \in L$ for all $i \geq 0$.

So $L(M)$ contains, as a subset, the infinite language $uv^* w$. 
Now let’s prove the other direction. Suppose $L(M)$ is infinite.

Since $L(M)$ is infinite, $M$ must accept arbitrarily long strings.

Let $z$ be a string that $M$ accepts, of length $\geq n$, but as short as possible subject to this constraint. (*)

Assume, to get a contradiction, that $|z| \geq 2n$.

Apply the pumping lemma to $z$.

We get a decomposition $z = uvw$ with $|v| \geq 1$ such that $uv^i w \in L(M)$ for all $i \geq 0$. 
Checking if $L(M)$ is infinite in the black box model

Choose $i = 0$. Then $uw \in L(M)$ and $|uw| < |z|$.

Furthermore, since $|z| \geq 2n$ and $|v| \leq |uv| \leq n$, we have $|uw| \geq n$.

So $uw$ is a string in $L$, of length $\geq n$, but shorter than $z$, contradicting (*).

This contradiction shows that $|z| < 2n$ as desired. ■

To use this theorem in an algorithm, we feed our black box for $M$ with all strings of lengths $\ell$ with $n \leq \ell < 2n$.

If the box accepts at least one such string, then $L(M)$ is infinite; otherwise it is finite.
Checking if $L(M_1) = L(M_2)$

To check if $L(M_1) = L(M_2)$, we use the same idea as in the clear box model: $L(M_1) = L(M_2)$ iff $L(M_1) \Delta L(M_2) = \emptyset$.

The only problem is, we don’t actually know $M_1$ or $M_2$; we just have black boxes for them!

But this is no problem.

We can simulate the automaton $M$ for $L(M_1) \Delta L(M_2)$ on $x$ by running the black boxes for $M_1$ and $M_2$ in parallel on $x$ and observing the results. Then $M$ would have accepted $x$ iff exactly one of the boxes for $M_1$ and $M_2$ accepts $x$. 
Checking if $L(M_1) = L(M_2)$

This gives us (essentially) a black box $B$ for an automaton $M$ recognizing $L(M_1) \Delta L(M_2)$.

So it remains to use our black box algorithm for checking if $L(M) = \emptyset$ on this new black box $B$.

We just need to check all strings up to (but not including) the number of states in $M$.

If $M_1$ has $m$ states and $M_2$ has $n$ states, then we saw above that a construction for $L(M_1) \Delta L(M_2)$ has $mn$ states.
Checking if $L(M_1) = L(M_2)$

So it suffices to check all strings of length $< mn$.

This gives the following algorithm for checking if $L(M_1) = L(M_2)$: feed black boxes for $M_1$ and $M_2$ with all strings of length $< mn$.

If for at least one string $x$, one box accepts $x$ but the other rejects $x$, then $L(M_1) \neq L(M_2)$.

Otherwise $L(M_1) = L(M_2)$. 
You can now easily check that the two theorems we proved for DFA’s also apply to NFA’s. This gives analogous black box algorithms for checking if $L(M) = \emptyset$ or $L(M)$ infinite.

For checking if $L(M_1) = L(M_2)$, a little more work is needed, and how to do this is left as an exercise.