Lecture 23: Constructing automata, proving correct

- Specifications & a correctness proof
- Strengthening the I.H.
- Automata union & intersection

Applications:

- Lessons for proving programs correct
  ... and building complex programs
Claim: $L(M) > L$ (for previous diagram)

Proof: WTS $L(M) \subseteq L$ and $L \subseteq L(M)$

- WTS $\forall x \in L(M), \exists y \in L$ (only accept state)
  - i.e. If $\delta(q_0, x) = q_0$, then $x$ has even $\#1's$
  - Proof by induction in $x$.
    - Let $P(x) = \text{WTS } P(x)$ and $P(xa)$, assuming $P(x)$.
    - $P(\epsilon)$ has $0 \#1's$, and $0$ is even.
    - WTS if $\delta(q_0, \epsilon) = q_0$, then $x$ has odd $\#1$ (this is vacuously true).
    - If $P$ is false, then "if $P$ then $Q$" is automatically true.

- $P(xa)$: Assume $P(x)$, WTS if $\delta(q_0, xa) = q_0$, then $xa$ has even $\#1's$.

Since $a \in \Sigma$, know either $a = 0$ or $a = 1$.

- If $a = 0$, then since $\delta(q_0, xa) = q_0$,
  - the only way this can happen is if $\delta(q_0, x) = q_0$, so by $P(x)$,
    - $x$ has even $\#1's$. So $xa = xa'$ also has even $\#1's$.

- If $a = 1$, since $\delta(q_0, xa) = q_0$, we must have followed "1" transition from $q_0$; i.e.
  - $\delta(q_0, x) = q_1$. By $P(x)$, $x$ has an odd $\#1's$.

- So $xa = x$ has an odd $\#1's$.

Let $P(x) = "$if $\delta(q_0, x) = q_0$ then $x$ has even $\#1's$ AND if $\delta(q_0, x) = q_1$ then $x$ has odd $\#1's$".

Last part:
- Check that if $\delta(q_0, x) \in A$ then $x \in L$, and if $\delta(q_0, x) \notin A$ then $x \notin L$.

Contrapositive of "if $x \in L$ then $x \in L(M)$".

Check A

- Check $\delta$

Check 4 transitions ($\delta$)

WTS if $\delta(q_0, xa) = q_0$, then $xa$ has odd $\#1's$.

Know $a = 0$ or $a = 1$.

If $a = 0$ and $\delta(q_0, xa) = q_0$, we have $\delta(q_0, x) = q_0$, so $P(x)$ says $x$ has odd $\#1's$, so $xa = xa'$ has an odd $\#1's$.

If $a = 1$ then $\delta(q_0, xa) = q_0$, so by $P(x)$

$x$ has even $\#1's$, so $xa = x$ has odd $\#1's$. 
Build a machine that accepts strings with an even #1's and don't contain 2 0's or more.

Check: this machine accepts strings with at most one 0.

Claim: if \( L_1 = L(M_1) \) and \( L_2 = L(M_2) \), then
\[
\exists M \text{ with } L(M) = L_1 \cap L_2.
\]

Terminology: if \( L \) is the language of a DFA, we say \( L \) is DFA-recognizable.

Restated claim: The intersection of DFA-recognizable languages is DFA-recognizable.