\( \varepsilon \)-NFAs
Here is another finite automaton -- an $\varepsilon$-NFA. Kozen calls these "NFAs with $\varepsilon$-transitions". These allow transitions labeled "$\varepsilon$" to be followed without consuming any input. These aren't interesting in themselves but are useful for showing that DFAs and regular expressions describe the same languages.

Example: Here is an $\varepsilon$-NFA that accepts strings with 2 0's or 2 1's:
Formally, an \( \varepsilon \)-NFA is \((Q, \Sigma, \delta, s, F)\) where \(Q, \Sigma, s,\) and \(F\) are defined as with other NFAs and the inputs to \(\delta\) are a state and either a letter in \(\Sigma\) or \(\varepsilon\). This processes strings in the same way as the NFA \((\Sigma, Q, \delta\', s, F)\), where \(\delta'(q,a) = \delta(q,a) \cup \bigcup_{q' \in \delta(q,\varepsilon)} \delta'(q', a)\). (Note that this is a recursive definition of \(\delta'\).)
We are going to show that the language accepted by an \( \varepsilon \)-NFA is regular (so every \( \varepsilon \)-NFA has an equivalent DFA). To get there we need the idea of an \( \varepsilon \)-closure of a set of states. Let \((Q, \Sigma, \delta, s, F)\) be the \( \varepsilon \)-NFA we are talking about and let \( A \) be an set of states from \( Q \). \( \bar{A} \) will represent the closure of \( A \). Here are two things we want to be true:

- \( A \subset \bar{A} \)
- For each \( q \) in \( \bar{A} \) if there is an \( \varepsilon \)-transition from \( q \) to \( q_1 \) then \( q_1 \) should be in \( \bar{A} \).

This gives us an algorithm: to compute \( \bar{A} \) start with \( A \) and add the destinations of \( \varepsilon \)-transitions until nothing else can be added.
Example:

This accepts $1^*0+01^*$ Here are some $\varepsilon$-closures:

\[
\{T\} = \{T, U\} \quad \{U\} = \{U\} \\
\{S\} = \{S, T, U\} \\
\{S, W\} = \{S, W, T, U, X\}
\]
Theorem: Any language accepted by an $\varepsilon$-NFA is regular.

Construction: Let $(Q, \Sigma, \delta, s, F)$ be an $\varepsilon$-NFA. We will construct an equivalent DFA $(Q', \Sigma, \delta', s', F')$:

- $Q'$ consists of subsets of $Q$
- $s' = \{s\}$
- If $P=\{q_0, q_1, \ldots q_k\}$ is a state in $Q'$ and $a$ is in $\Sigma$ then $P'=\bigcup_{i=0}^{k} \delta(q_i, a)$ is also a state in $Q'$ and $\delta'(P, a)=P'$
- If $P=\{q_0, q_1, \ldots q_k\}$ is a state in $Q'$ and if any of the $q_i$ are in $F$, then $P$ is in $F'$. 
Example:

$\varepsilon$-NFA:

Equivalent DFA:
Example:

\[ S \xrightarrow{\varepsilon} T \xrightarrow{0} U \xrightarrow{0} V \xrightarrow{0,1} \] strings with 2 0's

\[ W \xrightarrow{1} X \xrightarrow{1} Y \xrightarrow{0,1} \] strings with 2 1's

\[ S,T,W \xrightarrow{0} U,W \xrightarrow{0} V,W \xrightarrow{1} V,X \xrightarrow{1} V,Y \xrightarrow{0,1} \] Equivalent DFA:

\[ S \xrightarrow{1} T,X \xrightarrow{0} T,Y \xrightarrow{1} T,Y \xrightarrow{1} T,Y \xrightarrow{1} T,Y \xrightarrow{1} T,Y \xrightarrow{1} T,Y \xrightarrow{1} T,Y \]
Note that this can be simplified to:
We still need to prove that the DFA of this construction accepts exactly the same strings as the original $\varepsilon$-NFA. The proof is almost exactly the same as the proof that NFAs are equivalent to DFAs. If a string is accepted by the $\varepsilon$-NFA, processing the string takes the automaton through states $q_0$, $q_1$, ..., $q_k$, where $q_k$ is final. The $q_i$ will be elements of states through which the DFA will pass while processing the string. The DFA will end in a state containing $q_k$, which is final, so it will accept the string.

Alternatively, if $\alpha$ is a prefix of the string at it takes the DFA to state \{ $q_0$, ...$q_j$\} then on input $\alpha$ the $\varepsilon$-NFA could be in any of the $q_i$ states. If the full string is accepted by the DFA it will also take the $\varepsilon$-NFA to an accept state.