Chomsky hierarchy

- Regular languages
  - Expressive power
- Context-free languages
  - Easy to process
- Context-sensitive languages
- Recursively enumerable languages
Two characterisations of regular languages

A “regular language” means a language that can be defined using a regular expression.
A “regular language” means a language that can be defined using a regular expression.

**Kleene’s theorem**

Regular languages are those languages that can be accepted by finite automata.
A “regular language” means a language that can be defined using a regular expression.

**Kleene’s theorem**

Regular languages are those languages that can be accepted by finite automata.

We have already seen that any NFA has an equivalent DFA. So when we talk about languages that can be accepted by finite automata, we don’t need to specify whether we are talking about DFAs or NFAs.
A “regular language” means a language that can be defined using a regular expression.

**Kleene’s theorem**

Regular languages are those languages that can be accepted by finite automata.

We have already seen that any NFA has an equivalent DFA. So when we talk about languages that can be accepted by finite automata, we don’t need to specify whether we are talking about DFAs or NFAs.

We prove Kleene’s theorem by showing how to convert from a NFA to a reg expr, and vice versa.
Convert regular expression to NFA

Recall: we have seen to convert NFA to DFA!

Recall how r.e’s are defined. To show that any r.e. has an equivalent NFA:

1. Construct a NFA that accepts any single one-letter word (easy!)
2. Given two NFAs, construct a new one that accepts the concatenation of their languages
3. Given two NFAs, construct a new one that accepts the union of their languages
4. Given any single NFA, construct a new one that accepts the closure of its language
recall: we have seen to convert NFA to DFA!
recall: we have seen to convert NFA to DFA!
Recall how r.e.’s are defined. To show that any r.e. has an equivalent NFA:

1. Construct a NFA that accepts any single one-letter word (easy!)
2. Given two NFAs, construct a new one that accepts the concatenation of their languages
3. Given two NFAs, construct a new one that accepts the union of their languages
4. Given any single NFA, construct a new one that accepts the closure of its language
Claim 1

Let $L_1$ and $L_2$ be languages over an alphabet $A$. If there are finite automata accepting $L_1$ and $L_2$ then there is a finite automaton accepting $L_1 L_2$. 

Proof.

Assume $L_1$ and $L_2$ are accepted by DFAs $A_1 = (Q_1, A, \phi_1, i_1, T_1)$ and $A_2 = (Q_2, A, \phi_2, i_2, T_2)$.

Define $A$ to be the automaton $(Q_1 \cup Q_2, A, \psi, i_1, T_2)$ where

$\psi$ is defined for $q_1 \in Q_1$ by 

$\psi(q_1, a) = \begin{cases} \phi_1(q_1, a) & \text{if } \phi_1(q_1, a) \not\in T_1 \\ \phi_1(q_1, a), i_2 & \text{if } \phi_1(q_1, a) \in T_1 \end{cases}$

and for $q_2 \in Q_2$ by 

$\psi(q_2, a) = \phi_2(q_2, a)$.

Paul Goldberg  COMP 218; slides 5, 2012-13 4 / 19
Claim 1

Let $L_1$ and $L_2$ be languages over an alphabet $A$. If there are finite automata accepting $L_1$ and $L_2$ then there is a finite automaton accepting $L_1L_2$.

Proof. Assume $L_1$ and $L_2$ are accepted by DFAs $A_1 = (Q_1, A, \phi_1, i_1, T_1)$ and $A_2 = (Q_2, A, \phi_2, i_2, T_2)$. 
Claim 1

Let $L_1$ and $L_2$ be languages over an alphabet $A$. If there are finite automata accepting $L_1$ and $L_2$ then there is a finite automaton accepting $L_1 L_2$.

Proof. Assume $L_1$ and $L_2$ are accepted by DFAs $A_1 = (Q_1, A, \phi_1, i_1, T_1)$ and $A_2 = (Q_2, A, \phi_2, i_2, T_2)$. Define $A$ to be the automaton $(Q_1 \cup Q_2, A, \psi, i_1, T_2)$ where $\psi$ is defined for $q_1 \in Q_1$ by

$$\psi(q_1, a) = \{\phi_1(q_1, a)\} \text{ if } \phi_1(q_1, a) \notin T_1$$
$$\psi(q_1, a) = \{\phi_1(q_1, a), i_2\} \text{ if } \phi_1(q_1, a) \in T_1.$$

and for $q_2 \in Q_2$ by

$$\psi(q_2, a) = \{\phi_2(q_2, a)\}.$$
Example

\{aa,aaa\}

\[\rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ\]

\textit{ab(a*)}

\[\rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ\]

\{aa,aaa\}ab(a*)

\[\rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ\]
Next: verify that $L_1L_2$ is the language accepted by $A$. First, prove that $L_1L_2 \subseteq L(A)$. 
Next: verify that $L_1L_2$ is the language accepted by $A$.
First, prove that $L_1L_2 \subseteq L(A)$.
Suppose $w_1 \in L_1$ and $w_2 \in L_2$. Then $\phi_1(i_1, w_1) \in T_1$ and $\phi_2(i_2, w_2) \in T_2$. By the definition of $\psi$ we see that $i_2 \in \psi(i_1, w_1)$ and $\phi_2(i_2, w_2) \in \psi(i_1, w_1w_2)$. Now in $A$ the path with label $w_1w_2$ has the form

$$i_1 \rightarrow^{w_1} \cdots \rightarrow \{\ldots, i_2, \ldots\} \rightarrow^{w_2} \cdots \rightarrow \{\ldots, \phi_2(i_2, w_2), \ldots\}$$

so the final set $\psi(i_1, w_1w_2)$ contains a state from $T_2$, i.e. an accepting state of $A$. Hence $w_1w_2$ is accepted by $A$. This shows that $L_1L_2 \subseteq L(A)$.
Then, prove that $L(A) \subseteq L_1L_2$. 
Then, prove that $L(A) \subseteq L_1L_2$.
Proof sketch (see textbook for full proof):
Suppose that $w \in L(A)$.
There is an accepting path in $A$ labelled by the letters of $w$. From each set of states on the path, choose one that is “the right choice” for the non-deterministic machine $A$.
At some point the path passes from states in $Q_1$ to states in $Q_2$, and never returns to $Q_1$.
This point gives the right point to split $w$ into two words $w_1$ and $w_2$ accepted by $A_1$ and $A_2$ respectively.
Claim 2

Let $L_1$ and $L_2$ be languages over an alphabet $A$. If there are finite automata accepting $L_1$ and $L_2$ then there is a finite automaton accepting $L_1 \cup L_2$.

We will show how to do it on the above example, then give a general description.
Example

Make the 2 original DFAs have complete transition functions.
New state for each *pair* of old states...
Initial state: pair of initial states...
Example

Accepting states: pairs containing accepting states
Example

Transitions mimic the original transitions
Example of accepting path for \texttt{aaa}.
Claim 2

Let $L_1$ and $L_2$ be languages over an alphabet $A$. If there are finite automata accepting $L_1$ and $L_2$ then there is a finite automaton accepting $L_1 \cup L_2$. 
Claim 2

Let $L_1$ and $L_2$ be languages over an alphabet $A$. If there are finite automata accepting $L_1$ and $L_2$ then there is a finite automaton accepting $L_1 \cup L_2$.

Proof. Let $A_1 = (Q_1, A, \phi_1, i_1, T_1)$ be an automaton accepting $L_1$ and $A_2 = (Q_2, A, \phi_2, i_2, T_2)$ be an automaton accepting $L_2$, where we may assume both are deterministic. Define $A = (Q, A, \phi, i, T)$ as follows:

\[
\begin{align*}
Q &= Q_1 \times Q_2 \\
\phi((q_1, q_2), a) &= (\phi_1(q_1, a), \phi_2(q_2, a)) \\
&\quad \text{for all } q_1 \in Q_1; \ q_2 \in Q_2; \ a \in A \\
i &= (i_1, i_2). \\
T &= \{(p, q) : p \in T_1 \text{ or } q \in T_2\}
\end{align*}
\]

Then

\[
\phi(i, w) = (\phi_1(i_1, w), \phi_2(i_2, w)) \text{ for all } w \in A^*.
\]
Now we have

\[
\begin{align*}
    w \in L_1 \cup L_2 & \iff \phi_1(i_1, w) \in T_1 \text{ or } \phi_2(i_2, w) \in T_2 \\
    & \iff \phi(i, w) = (p, q) \\
    & \text{where } p \in T_1 \text{ or } q \in T_2 \text{ (or both)}
\end{align*}
\]

So from the definition of the set \( T \) of terminal states of \( A \) we have

\[
    w \in L_1 \cup L_2 \iff \phi(i, w) \in T \iff w \text{ is accepted by } A.
\]

Hence the language accepted by \( A \) is \( L_1 \cup L_2 \).
Observation

The 2 constructions so far give a general way of constructing a finite automaton that accepts any finite language.

A word in a finite language is built from a sequence of concatenations.

The language is built from a sequence of unions of sets of words.
Claim 3

If language $L$ is accepted by some finite automaton, then so is language $L^*$. 
Claim 3

*If language $L$ is accepted by some finite automaton, then so is language $L^*$.  

Make new initial state $i'$, with $i' \in T$.
Let $\phi'$ be the new transition function, where $\phi'$ contains all transitions of $\phi$, and in addition:
If $\phi(i, a) = q$ for $a \in A$, $q \in Q$ then $\phi'(i', a) = q$.
If $\phi(q, a) = t \in T$ for $q \in Q$, then let $\phi'(q, a) = \{i', t\}$. (making the automaton non-deterministic)
“union” construction could be modified in obvious (?) way to get an “intersection” result; could also do symmetric difference, or complicated boolean combinations of more than 2 languages. (eg: given 4 languages $L_1, L_2, L_3, L_4$, define a new language $L_{new}$ as $w \in L_{new}$ iff $w \in$ exactly 2 of $L_1, L_2, L_3, L_4$.) But it’s not so obvious how to do that with regular expressions! Next: conversion from any finite automaton to an equivalent r.e. (which completes Kleene’s theorem)