COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

Lec. 8 : Regular and NON-Reg. Languages
GNFA $\rightarrow$ Reg. Expression

Claim 1.65

For any GNFA $G$, $\text{CONVERT}(G)$ is equivalent to $G$.

We prove this claim by induction on $k$, the number of states of the GNFA.

“equivalent” means $L(\text{CONVERT}(G)) = L(G)$
**GNFA → Reg. Expression**

- **Induction basis**

  Let G be a GNFA with exactly $k=2$ states.

  Because of the special form of our GNFA, the two states are the start and accept states. The regular expression on the transition from $q_{start}$ to $q_{accept}$ generates the language accepted by this GNFA.
Because of the special form of our GNFA, the two states are the start and accept states. The regular expression on the transition from q_{start} to q_{accept} generates the language accepted by this GNFA.
Induction basis

Let G be a GNFA with exactly $k=2$ states.

Because of the special form of our GNFA, the two states are the start and accept states. The regular expression on the transition from $q_{\text{start}}$ to $q_{\text{accept}}$ generates the language accepted by this GNFA.
GNFA $\rightarrow$ Reg. Expression
GNFA $\rightarrow$ Reg. Expression

- Induction step
GNFA $\rightarrow$ Reg. Expression

- **Induction step**

Let $G$ be a GNFA with exactly $k>2$ states. We assume for induction hypothesis that all GNFA $G'$ of $k-1$ states accept the language defined by the regular expression obtained via CONVERT, i.e. $L(G')=L(CONVERT(G'))$. 
GNFA → Reg. Expression

Induction step

Let G be a GNFA with exactly k>2 states. We assume for induction hypothesis that all GNFA G’ of k-1 states accept the language defined by the regular expression obtained via CONVERT, i.e. \( L(G’)=L(\text{CONVERT}(G’)) \).

Since \( k>2 \) then there exists at least one state \( q_{rip} \) which is neither \( q_{start} \) nor \( q_{accept} \).
Let G be a GNFA with exactly k>2 states. We assume for induction hypothesis that all GNFA G′ of k-1 states accept the language defined by the regular expression obtained via CONVERT, i.e. L(G′)=L(CONVERT(G′)).

Since k>2 then there exists at least one state qrip which is neither qstart nor qaccept.
Let $G$ be a GNFA with exactly $k>2$ states. We assume for induction hypothesis that all GNFA $G'$ of $k-1$ states accept the language defined by the regular expression obtained via CONVERT, i.e. $L(G')=L(\text{CONVERT}(G'))$.

Since $k>2$ then there exists at least one state $q_{\text{rip}}$ which is neither $q_{\text{start}}$ nor $q_{\text{accept}}$.

Let $G'$ be, as in CONVERT, the GNFA obtained after ripping $q_{\text{rip}}$ from $G$. 
Let $G$ be a GNFA with exactly $k>2$ states. We assume for induction hypothesis that all GNFA $G'$ of $k-1$ states accept the language defined by the regular expression obtained via CONVERT, i.e. $L(G')=L(\text{CONVERT}(G'))$.

Since $k>2$ then there exists at least one state $q_{\text{rip}}$ which is neither $q_{\text{start}}$ nor $q_{\text{accept}}$.

Let $G'$ be, as in CONVERT, the GNFA obtained after ripping $q_{\text{rip}}$ from $G$. 

\[
G = \begin{array}{c}
\begin{array}{c}
q_{\text{start}} \\
\rightarrow
\end{array} \\
\begin{array}{c}
\begin{array}{c}
q_2 \\
b
\end{array} \\
\rightarrow
\end{array} \\
\begin{array}{c}
\begin{array}{c}
q_{\text{rip}} \\
ab^* \\
aa
\end{array} \\
\rightarrow
\end{array} \\
\begin{array}{c}
\begin{array}{c}
q_{\text{accept}} \\
ab \cup ba
\end{array} \\
\rightarrow
\end{array}
\end{array}
\]

\[
G' = \begin{array}{c}
\begin{array}{c}
q_{\text{start}} \\
\rightarrow
\end{array} \\
\begin{array}{c}
\begin{array}{c}
q_2 \\
b^*
\end{array} \\
\rightarrow
\end{array} \\
\begin{array}{c}
\begin{array}{c}
q_{\text{rip}} \\
ab^* \\
aa
\end{array} \\
\rightarrow
\end{array} \\
\begin{array}{c}
\begin{array}{c}
q_{\text{accept}} \\
b^* \cdot (aa)^* (aa)^* (ab \cup ba)
\end{array} \\
\rightarrow
\end{array}
\end{array}
\]
GNFA $\rightarrow$ Reg. Expression
Let $w$ be a string accepted by $G$, $w \in L(G)$. Consider an accepting sequence $q_{\text{start}}, q_1, q_2, \ldots, q_{\text{accept}}$ for string $w$. 
Let $w$ be a string accepted by $G$, $w \in L(G)$. Consider an accepting sequence $q_{\text{start}}, q_1, q_2, \ldots, q_{\text{accept}}$ for string $w$. 
Let \( w \) be a string accepted by \( G \), \( w \in L(G) \).
Consider an accepting sequence \( q_{start}, q_1, q_2, ..., q_{accept} \) for string \( w \).

If \( q_{rip} \) is not a state of the sequence, then the very same exact sequence will accept \( w \) in \( G' \) because its transitions \( R_4 \) contain all those \( R_4 \) in \( G \) (except for \( q_{rip} \)) in a union with new possibilities related to ripping \( q_{rip} \).
Let $w$ be a string accepted by $G$, $w \in L(G)$. Consider an accepting sequence $q_{\text{start}}, q_1, q_2, \ldots, q_{\text{accept}}$ for string $w$.

If $q_{\text{rip}}$ is not a state of the sequence, then the very same exact sequence will accept $w$ in $G'$ because its transitions $R_4$ contain all those $R_4$ in $G$ (except for $q_{\text{rip}}$) in a union with new possibilities related to ripping $q_{\text{rip}}$. 
If $q_{rip}$ is a state of the sequence, then the same sequence (but with all $q_{rip}$ removed) will accept $w$ in $G'$. That's because any three elements in a row $q_i,q_{rip},q_j$ ($q_i \neq q_{rip} \neq q_j$) in $G'$'s accepting sequence, will be processed identically through states $q_i,q_j$ in $G'$. Remember that the transitions for $q_i,q_j$ in $G'$ contain all those $R_1(R_2)^*R_3$ from $G$ involving $q_{rip}$ in a union with older possibilities ($R_4$). (we can deal with $q_i,q_{rip},...,q_{rip},q_j$ similarly.)
If \( q_{\text{rip}} \) is a state of the sequence, then the same sequence (but with all \( q_{\text{rip}} \) removed) will accept \( w \) in \( G' \). That's because any three elements in a row \( q_i, q_{\text{rip}}, q_j \) (\( q_i \neq q_{\text{rip}} \neq q_j \)) in \( G' \)’s accepting sequence, will be processed identically through states \( q_i, q_j \) in \( G' \). Remember that the transitions for \( q_i, q_j \) in \( G' \) contain all those \( R_1(R_2)^*R_3 \) from \( G \) involving \( q_{\text{rip}} \) in a union with older possibilities (\( R_4 \)). (we can deal with \( q_i, q_{\text{rip}}, ..., q_{\text{rip}}, q_j \) similarly.)
If $q_{\text{rip}}$ is a state of the sequence, then the same sequence (but with all $q_{\text{rip}}$ removed) will accept $w$ in $G'$. That's because any three elements in a row $q_i,q_{\text{rip}},q_j$ $(q_i \neq q_{\text{rip}} \neq q_j)$ in $G'$'s accepting sequence, will be processed identically through states $q_i,q_j$ in $G'$. Remember that the transitions for $q_i,q_j$ in $G'$ contain all those $R_1(R_2)^*R_3$ from $G$ involving $q_{\text{rip}}$ in a union with older possibilities $(R_4)$. (we can deal with $q_i,q_{\text{rip}},...,q_{\text{rip}},q_j$ similarly.)
This proved “if \( w \in L(G) \) then \( w \in L(G') \)”. We should also prove “if \( w \in L(G') \) then \( w \in L(G) \)".

Let \( w \) be a string accepted by \( G' \), i.e. \( w \in L(G') \). Consider an accepting sequence \( q_{\text{start}}, q_1, q_2, \ldots, q_{\text{accept}} \) for string \( w \). Consider any two consecutive states \( q_i, q_{i+1} \). The same portion of \( w \) is processed in \( G \) in either part of the union, \( R_1(R_2)^*R_3 \) or \( R_4 \), along the transition between \( q_i \) and \( q_{i+1} \).
This proved “if \( w \in L(G) \) then \( w \in L(G') \)”. We should also prove “if \( w \in L(G') \) then \( w \in L(G) \)

Let \( w \) be a string accepted by \( G' \), i.e. \( w \in L(G') \). Consider an accepting sequence \( q_{\text{start}}, q_1, q_2, \ldots, q_{\text{accept}} \) for string \( w \). Consider any two consecutive states \( q_i, q_{i+1} \). The same portion of \( w \) is processed in \( G \) in either part of the union, \( R_1(R_2)^*R_3 \) or \( R_4 \), along the transition between \( q_i \) and \( q_{i+1} \).
This proved “if \( w \in L(G) \) then \( w \in L(G') \)”. We should also prove “if \( w \in L(G') \) then \( w \in L(G) \).

Let \( w \) be a string accepted by \( G' \), i.e. \( w \in L(G') \). Consider an accepting sequence \( q_{\text{start}}, q_1, q_2, ..., q_{\text{accept}} \) for string \( w \). Consider any two consecutive states \( q_i, q_{i+1} \). The same portion of \( w \) is processed in \( G \) in either part of the union, \( R_1(R_2)^*R_3 \) or \( R_4 \), along the transition between \( q_i \) and \( q_{i+1} \).
If the portion of $w$ is generated by $R_4$ in $G'$ then it is also generated by $R_4$ in $G$. If the portion of $w$ is generated by $R_1(R_2)^*R_3$ in $G'$ then there exists an $m$ such that it is generated by $R_1(R_2)^mR_3$ and it is also generated in $G$ by $R_1$, going through $q_{rip}$ $m$ times via $R_2$ and finally $R_3$. Thus $q_i,q_{i+1}$ is replaced by $q_i,q_{rip},...,q_{rip},q_{i+1}$.

We conclude that if $w \in L(G')$ then $w \in L(G)$. 
If the portion of w is generated by R₄ in G' then it is also generated by R₄ in G. If the portion of w is generated by R₁(R₂)*R₃ in G' then there exists an m such that it is generated by R₁(R₂)mR₃ and it is also generated in G by R₁, going through qrip m times via R₂ and finally R₃. Thus qᵢ,qᵢ₊₁ is replaced by qᵢ,qrip,...,qrip,qᵢ₊₁.

We conclude that if w∈L(G') then w∈L(G).
If the portion of $w$ is generated by $R_4$ in $G'$ then it is also generated by $R_4$ in $G$. If the portion of $w$ is generated by $R_1(R_2)^mR_3$ in $G'$ then there exists an $m$ such that it is generated by $R_1(R_2)^mR_3$ and it is also generated in $G$ by $R_1$, going through $q_{rip}$ $m$ times via $R_2$ and finally $R_3$. Thus $q_i,q_{i+1}$ is replaced by $q_i,q_{rip},...,q_{rip},q_{i+1}$.

We conclude that if $w \in L(G')$ then $w \in L(G)$.
Combining both statements we get $L(G') = L(G)$.

By induction hypothesis $L(G') = L(CONVERT(G'))$ because $G'$ contains $k-1$ states. By construction, $CONVERT(G) = CONVERT(G')$. Therefore $L(G) = L(CONVERT(G)) = L(CONVERT(G')) = L(G')$.

QED
**FIGURE 1.62**
Typical stages in converting a DFA to a regular expression
DFA $\rightarrow$ GNFA $\rightarrow$ Reg. Exp.

Two examples
DFA $\rightarrow$ GNFA $\rightarrow$ Reg. Exp.

(a) DFA

(b) GNFA

Reg. Exp.
DFA $\rightarrow$ GNFA $\rightarrow$ Reg. Exp.
DFA $\rightarrow$ GNFA $\rightarrow$ Reg. Exp.
DFA $\rightarrow$ GNFA $\rightarrow$ Reg. Exp.
DFA $\rightarrow$ GNFA $\rightarrow$ Reg. Exp.
DFA $\rightarrow$ GNFA $\rightarrow$ Reg. Exp.

(c) DFA with transitions:
- $s$ to $a$: $ab$
- $s$ to $3$: $b$
- $3$ to $a$: $\varepsilon$
- $2$ to $s$: $\varepsilon$
- $2$ to $a$: $aa \cup b$
- $2$ to $3$: $ba \cup a$

(d) GNFA with transitions:
- $s$ to $a$: $a(aa \cup b)^*$
- $3$ to $s$: $(ba \cup a)(aa \cup b)^* \cup \varepsilon$
- $3$ to $bb$: $(ba \cup a)(aa \cup b)^* \cup ab \cup bb$
DFA $\rightarrow$ GNFA $\rightarrow$ Reg. Exp.

\[
\begin{align*}
(a(aa \cup b)*ab \cup b) & \quad (ba \cup a)(aa \cup b)* \cup \varepsilon \\
(ba \cup a)(aa \cup b)*ab \cup bb \\
\end{align*}
\]

(d)

\[
\begin{align*}
(a(aa \cup b)*ab \cup b)((ba \cup a)(aa \cup b)*ab \cup bb)*((ba \cup a)(aa \cup b)* \cup \varepsilon) & \cup a(aa \cup b)* \\
\end{align*}
\]

(e)
Multiples of 3 (base 10)

$q_0 = 0u3u6u9, \quad q_1 = 1u4u7, \quad q_2 = 2u5u8$
Multiples of 3 (base 10)

0 = 0u3, 3 = 3u6u9, 1 = 1u4u7, 2 = 2u5u8
Multiples of 3 (base 10)

$q_s$ 1u30*1 0u20*1
2u30*2
0u30*

1u20*2

20*

2u10*1

0u10*2

$q_A$

10*

$q_1$

$q_2$

0=0u3, 3=3u6u9, 1=1u4u7, 2=2u5u8
Multiples of 3 (base 10)

- $0u30*u$
- $(1u30*1)(0u20*1)*20*$
- $2u30*2u$
- $(1u30*1)(0u20*1)*(1u20*2)$

$q_s$
$q_A$
$q_2$

$0=0u3, \ 3=3u6u9,$
$1=1u4u7, \ 2=2u5u8$
Multiples of 3 (base 10)

0 = 0u3,  3 = 3u6u9,  1 = 1u4u7,  2 = 2u5u8
Multiples of 3 (base 10)

\[
\begin{align*}
3 &= 3u6u9, \\
0 &= 0u3, \\
1 &= 1u4u7, \\
2 &= 2u5u8
\end{align*}
\]
Application of the Myhill-Nerode Theorem
Application of the Myhill-Nerode Theorem

Given two regular expressions $R$ and $R'$ we can find out whether they generate the same regular language or not:
Application of the Myhill-Nerode Theorem

Given two regular expressions $R$ and $R'$ we can find out whether they generate the same regular language or not:

1. From $R$ and $R'$, compute NFAs $N$ and $N'$ accepting $L(R)$ and $L(R')$ (Lemma 1.55).
Application of the Myhill-Nerode Theorem

Given two regular expressions $R$ and $R'$ we can find out whether they generate the same regular language or not:

1. From $R$ and $R'$, compute NFAs $N$ and $N'$ accepting $L(R)$ and $L(R')$ (Lemma 1.55).

2. Compute equivalent DFAs $M$ and $M'$ (Thm 1.39).
Application of the Myhill-Nerode Theorem

Given two regular expressions $R$ and $R'$ we can find out whether they generate the same regular language or not:

1. From $R$ and $R'$, compute NFAs $N$ and $N'$ accepting $L(R)$ and $L(R')$ (Lemma 1.55).

2. Compute equivalent DFAs $M$ and $M'$ (Thm 1.39).

3. Using part (b) of Myhill-Nerode we construct minimal DFAs $W$ for $M$ and $W'$ for $M'$. 
Application of the Myhill-Nerode Theorem

Given two regular expressions $R$ and $R'$ we can find out whether they generate the same regular language or not:

1. From $R$ and $R'$, compute NFAs $N$ and $N'$ accepting $L(R)$ and $L(R')$ (Lemma 1.55).

2. Compute equivalent DFAs $M$ and $M'$ (Thm 1.39).

3. Using part (b) of Myhill-Nerode we construct minimal DFAs $W$ for $M$ and $W'$ for $M'$.

4. $L(R)=L(R')$ iff $W \approx W'$
   ($\approx$ means "identical up to state renaming").
Regular and non-Regular Languages
footnote 3 page 46:

Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ be a DFA accepting $L_A$ and $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ be a DFA accepting $L_B$. 
Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ be a DFA accepting $L_A$ and $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ be a DFA accepting $L_B$.

Consider $M_U = (Q_A \times Q_B, \Sigma, \delta_U, (q_{0A}, q_{0B}), F_U)$ where
\[
\delta_U((q, q'), s) = (\delta_A(q, s), \delta_B(q', s))
\]
for all $q, q', s$ and
\[
F_U = (F_A \times Q_B) \cup (Q_A \times F_B).
\]
Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ be a DFA accepting $L_A$ and $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ be a DFA accepting $L_B$.

Consider $M_U = (Q_A \times Q_B, \Sigma, \delta_U, (q_{0A}, q_{0B}), F_U)$ where

$$\delta_U((q, q'), s) = (\delta_A(q, s), \delta_B(q', s))$$

for all $q, q', s$ and

$$F_U = (F_A \times Q_B) \cup (Q_A \times F_B).$$

$L_U = L_A \cup L_B$. 
Let $M_A=(Q_A, \Sigma, \delta_A, q_0A, F_A)$ be a DFA accepting $L_A$ and $M_B=(Q_B, \Sigma, \delta_B, q_0B, F_B)$ be a DFA accepting $L_B$.

Consider $M_U=(Q_AQ_B, \Sigma, \delta_U, (q_0A, q_0B), F_U)$ where

$$\delta_U((q,q'),s) = (\delta_A(q,s), \delta_B(q',s))$$

for all $q, q', s$ and

$$F_U = (F_A \times Q_B) \cup (Q_A \times F_B).$$

$L_U = L_A \cup L_B$.

$F_U = F_A \times F_B$ would yield the intersection (and not the union) of $L_A$ and $L_B$. This proves that the class of regular languages is also closed under intersection.
NON-Regular Languages

B = \{ 0^n1^n \mid n \geq 0 \} 

C = \{ w \mid w \text{ contains an equal number of 0's and 1's} \} 

D = \{ w \mid w \text{ contains an equal number of occurrences of 01 and 10 as sub-strings} \}
NON-Regular Languages

- $B = \{ 0^n1^n | n \geq 0 \}$
- $C = \{ w | w \text{ contains an equal number of 0's and 1's} \}$
- $D = \{ w | w \text{ contains an equal number of occurrences of 01 and 10 as sub-strings} \}$
NON-Regular Languages

\[ B = \{ 0^n1^n \mid n \geq 0 \} \]

\[ C = \{ w \mid w \text{ contains an equal number of } 0\text{'s and } 1\text{'s} \} \]

\[ D = \{ w \mid w \text{ contains an equal number of occurrences of } 01 \text{ and } 10 \text{ as sub-strings} \} \]
NON-Regular Languages

- $B = \{ 0^n1^n \mid n \geq 0 \}$
- $C = \{ w \mid w$ contains an equal number of 0's and 1's $\}$
- $D = \{ w \mid w$ contains an equal number of occurrences of 01 and 10 as sub-strings $\}$

- NON-Regular

- NON-Regular

- NON-Regular

- Regular
All languages

Languages we can describe

Regular Languages
NON-Regular Languages

**Theorem:** Some languages are not regular.

**Proof idea:** all regular languages have certain properties. Some languages provably do not have one of these properties.
Computability Theory

- All languages
  - Regular Languages
  - NON-Regular Languages

languages we can describe

NON-Regular Languages via Pumping Lemma
NON-Regular Languages via Reductions
Reductions

If $C$ is regular then so is $B$.

Proof: Regular languages are closed under intersection (see footnote 3 page 46). Define $A = L(0^*1^*)$. Obviously $A$ is regular. If $C$ was regular then so would $C \cap A = B$.

QED

If $B$ is NON-regular then so is $C$.

$B = \{ 0^n1^n \mid n \geq 0 \}$

$C = \{ w \mid w \text{ contains an equal number of 0's and 1's} \}$
Reductions

- If $A$ is regular then so is $A'$.

- Regular languages are closed under complement (see ex. 1.14), intersection, union, concatenation and star. If there exists $R$, a regular language, such that either $A^c = A'$, $A^* = A'$, $A \cap R = A'$, $A \cup R = A'$, $A \circ R = A'$ or any combinations of these operations then $A'$ is regular as long as $A$ is.

- If $A'$ is NON-regular then so is $A$.
Simple Reductions

- If $A^*$ is NON-regular then so is $A$.
- If $A$ is NON-regular then so is $A^c$.
- If $A$ is NON-regular then so is $A^R$. 
Complex Reductions

Let \( A' = (A \cup R) \cap (A^c \cup R') \) \hspace{1cm} (R,R' regular)

Let \( A' = ((A^c \cap R) \cup (A^* \cap R')) \circ R'' \) \hspace{1cm} (R,R',R'' regular)

Let \( A' = (A \circ R) \cap (A^c \circ R') \) \hspace{1cm} (R,R' regular)

If \( A' \) is NON-regular then so is \( A \).
Non-Regular Languages

Theorem: Some languages are not regular.

Proof idea: All regular languages have certain properties. Some languages provably do not have one of these properties.

Example: A property of all regular languages = the Pumping Lemma.
COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

Lec. 8 : Regular and NON-Reg. Languages