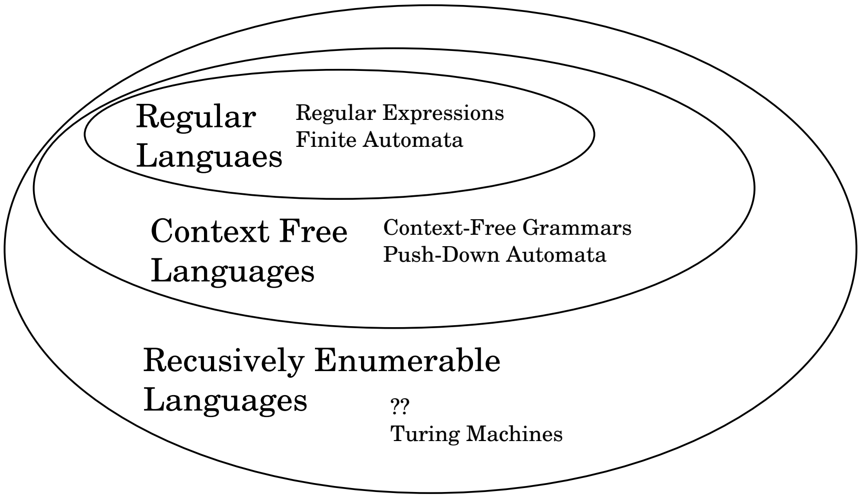
13-0: **Language Hierarchy**



13-1: **CFG Review**

$G = (V, \Sigma, R, S)$

- V = Set of symbols, both terminals & non-terminals
- $\Sigma \subset V$ set of terminals (alphabet for the language being described)
- $R \subset ((V - \Sigma) \times V^*)$ Set of rules
- $S \in (V - \Sigma)$ Start symbol

13-2: **Unrestricted Grammars**

$G = (V, \Sigma, R, S)$

- V = Set of symbols, both terminals & non-terminals
- $\Sigma \subset V$ set of terminals (alphabet for the language being described)
- $R \subset (V^*(V - \Sigma)V^* \times V^*)$ Set of rules
- $S \in (V - \Sigma)$ Start symbol

13-3: **Unrestricted Grammars**

- $R \subset (V^*(V - \Sigma)V^* \times V^*)$ Set of rules
- In an Unrestricted Grammar, the left-hand side of a rule contains a string of terminals and non-terminals (at least one of which must be a non-terminal)
- Rules are applied just like CFGs:
  - Find a substring that matches the LHS of some rule
  - Replace with the RHS of the rule

13-4: **Unrestricted Grammars**

- To generate a string with an Unrestricted Grammar:

- Start with the initial symbol
- While the string contains at least one non-terminal:
  - Find a substring that matches the LHS of some rule
  - Replace that substring with the RHS of the rule

13-5: **Unrestricted Grammars**

- Example: Grammar for $L = \{a^n b^n c^n : n > 0\}$
  - First, generate $(ABC)^*$
  - Next, non-deterministically rearrange string
  - Finally, convert to terminals ($A \rightarrow a$, $B \rightarrow b$, etc.), ensuring that string was reordered to form $a^*b^*c^*$

13-6: **Unrestricted Grammars**

- Example: Grammar for $L = \{a^n b^n c^n : n > 0\}$

$$\begin{aligned}
S &\rightarrow ABCS \\
S &\rightarrow T_C \\
CA &\rightarrow AC \\
BA &\rightarrow AB \\
CB &\rightarrow BC \\
CT_C &\rightarrow T_C c \\
T_C &\rightarrow T_B \\
BT_B &\rightarrow T_B b \\
T_B &\rightarrow T_A \\
AT_A &\rightarrow T_A a \\
T_A &\rightarrow \epsilon
\end{aligned}$$

13-7: **Unrestricted Grammars**

$$\begin{aligned}
S &\Rightarrow ABCS \\
&\Rightarrow ABCABCS \\
&\Rightarrow ABACBCS \\
&\Rightarrow AABCBCS \\
&\Rightarrow AABBCCS \\
&\Rightarrow AABBCCT_C \\
&\Rightarrow AABBCT_C c \\
&\Rightarrow AABBT_C cc \\
&\Rightarrow AABBT_B cc \\
&\Rightarrow AABT_B bcc \\
&\Rightarrow AAT_B bbcc \\
&\Rightarrow AAT_A bbcc \\
&\Rightarrow AT_A abbcc \\
&\Rightarrow T_A aabbcc \\
&\Rightarrow aabbcc
\end{aligned}$$

13-8: **Unrestricted Grammars**

$$\begin{aligned}
S &\Rightarrow ABCS \\
&\Rightarrow ABCABCS \\
&\Rightarrow ABCABCABCS \\
&\Rightarrow ABACBCABCS \\
&\Rightarrow AABCBCABCS \\
&\Rightarrow AABCBACBCS \\
&\Rightarrow AABCABCBCS \\
&\Rightarrow AABACBCBCS \\
&\Rightarrow AAABCBCBCS \\
&\Rightarrow AAABBCCBCS \\
&\Rightarrow AAABBCBCCS \\
&\Rightarrow AAABBBCCCS \\
&\Rightarrow AAABBBCCCT_C \\
&\Rightarrow AAABBBCCT_C c \\
&\Rightarrow AAABBBCT_C cc \\
&\Rightarrow AAABBBT_C ccc \\
&\Rightarrow AAABBBT_B ccc \\
&\Rightarrow AAABBT_B bccc \\
&\Rightarrow AAABT_B bbccc \\
&\Rightarrow AAAT_B bbbccc \\
&\Rightarrow AAAT_A bbbccc \\
&\Rightarrow AAT_A abbbccc \\
&\Rightarrow AT_A aabbbccc \\
&\Rightarrow T_A aaabbbccc \Rightarrow aaabbbccc
\end{aligned}$$

13-9: **Unrestricted Grammars**

- Example: Grammar for $L = \{ww : w \in a, b^*\}$

13-10: **Unrestricted Grammars**

- Example: Grammar for $L = \{ww : w \in a, b^*\}$
- Hints:
  - What if we created a string, and then rearranged it (like $(abc)^* \rightarrow a^n b^n c^n$)

13-11: **Unrestricted Grammars**

- Example: Grammar for $L = \{ww : w \in a, b^*\}$
- Hints:
  - What if we created a string, and then rearranged it (like $(abc)^* \rightarrow a^n b^n c^n$)
  - What about trying ww^R ...

13-12: **Unrestricted Grammars**

- $L = \{ww : w \in a, b^*\}$

$$\begin{array}{rl} S & \rightarrow S'Z \\ S' & \rightarrow aS'A \\ S' & \rightarrow bS'B \\ S' & \rightarrow \epsilon \end{array}$$

$$\begin{array}{rlcrl} AZ & \rightarrow XZ & \quad & BZ & \rightarrow YZ \\ AX & \rightarrow XA & & AY & \rightarrow YA \\ BX & \rightarrow XB & & BY & \rightarrow YB \\ aX & \rightarrow aa & & aY & \rightarrow ab \\ bX & \rightarrow ba & & bY & \rightarrow bb \end{array}$$

13-13: **Unrestricted Grammars**

- L_{UG} is the set of languages that can be described by an Unrestricted Grammar:
  - $L_{UG} = \{L : \exists$ Unrestricted Grammar $G, L[G] = L\}$
- Claim: $L_{UG} = L_{re}$
- To Prove:
  - Prove $L_{UG} \subseteq L_{re}$
  - Prove $L_{re} \subseteq L_{UG}$

13-14: $L_{UG} \subseteq L_{re}$

- Given any Unrestricted Grammar G, we can create a Turing Machine M that semi-decides $L[G]$

13-15: $L_{UG} \subseteq L_{re}$

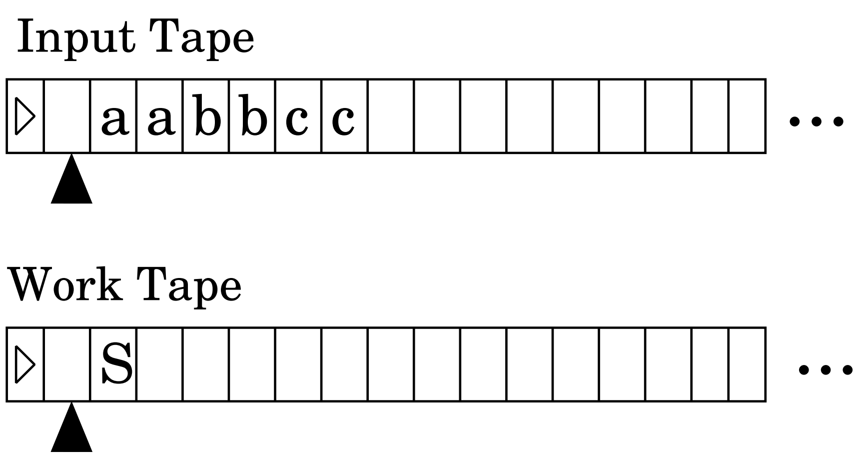
- Given any Unrestricted Grammar G, we can create a Turing Machine M that semi-decides $L[G]$
- Two tape machine:

- One tape stores the input, unchanged
- Second tape implements the derivation
- Check to see if the derived string matches the input, if so accept, if not run forever

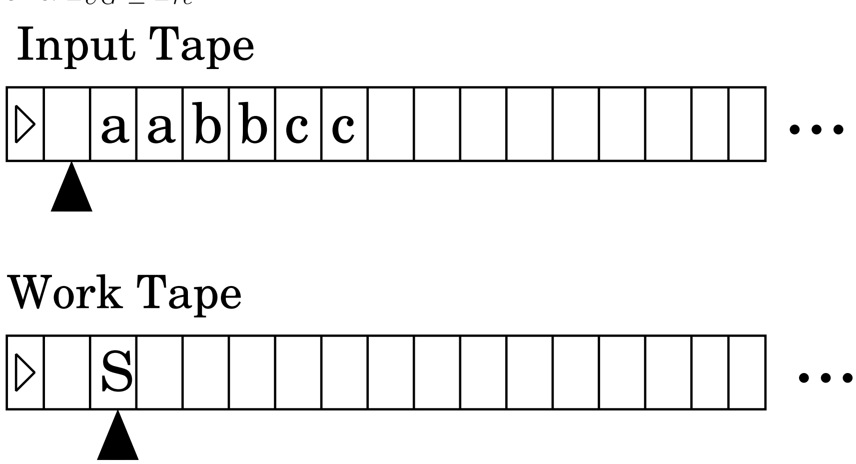
13-16: $L_{UG} \subseteq L_{re}$

- To implement the derivation on the second tape:
  - Write the initial symbol on the second tape
  - Non-deterministically move the read/write head to somewhere on the tape
  - Non-deterministically decide which rule to apply
  - Scan the current position of the read/write head, to make sure the LHS of the rule is at that location
  - Remove the LHS of the rule from the tape, and splice in the RHS

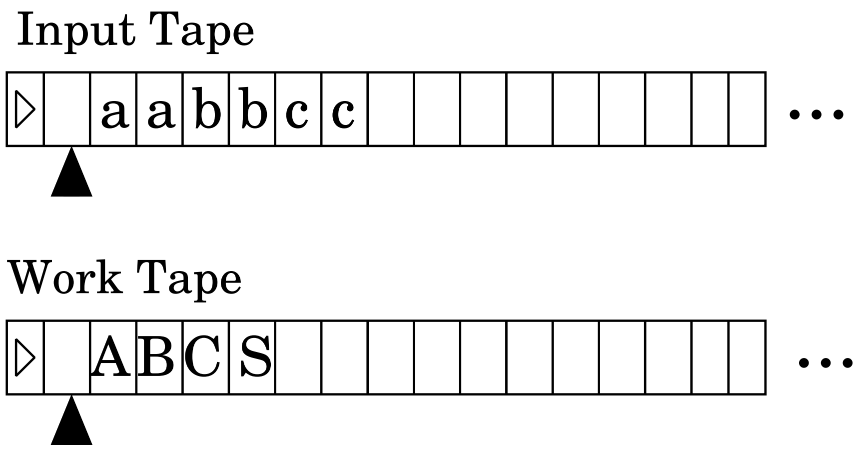
13-17: $L_{UG} \subseteq L_{re}$



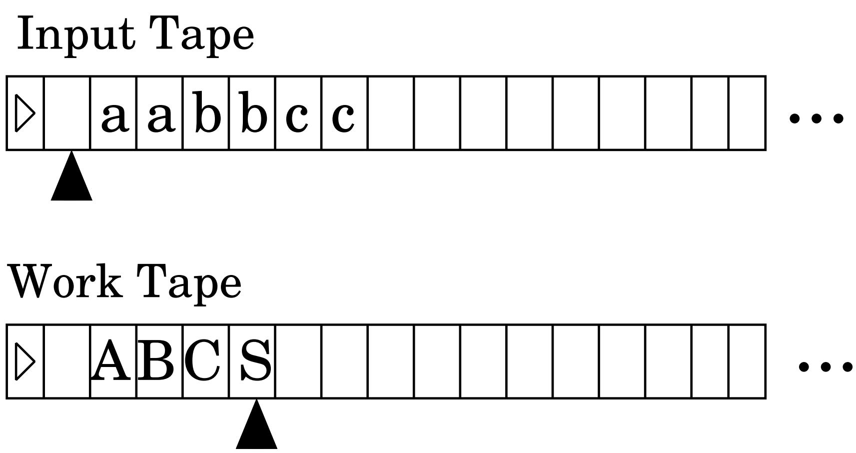
13-18: $L_{UG} \subseteq L_{re}$



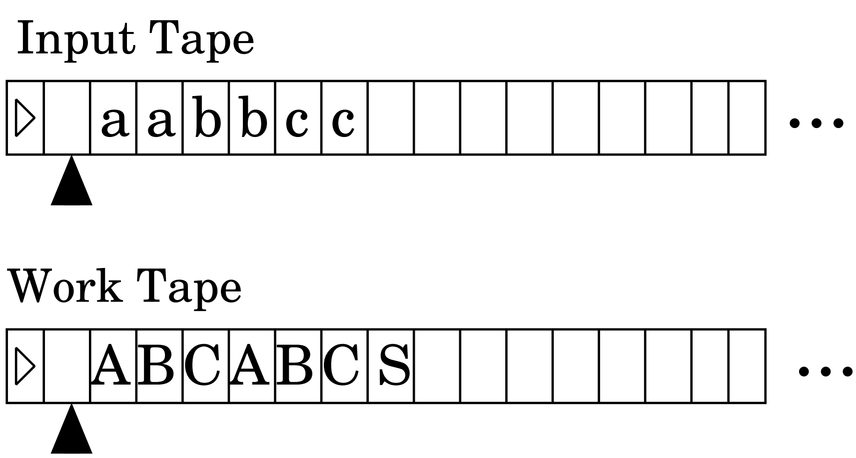
13-19: $L_{UG} \subseteq L_{re}$



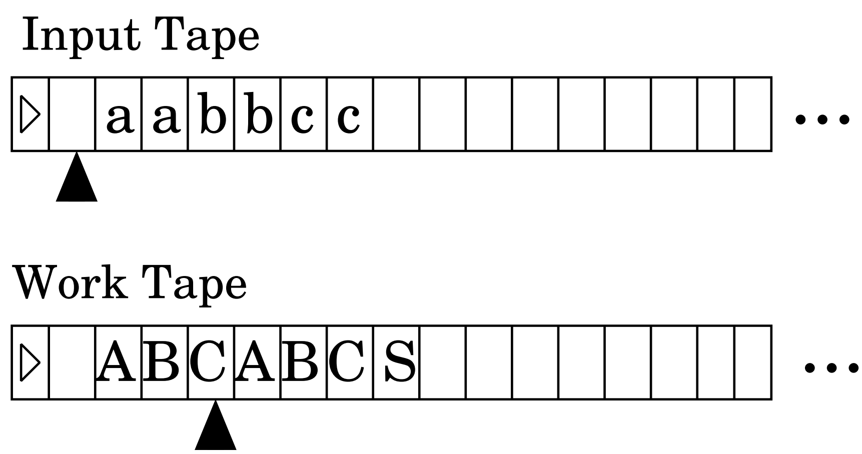
13-20: $L_{UG} \subseteq L_{re}$



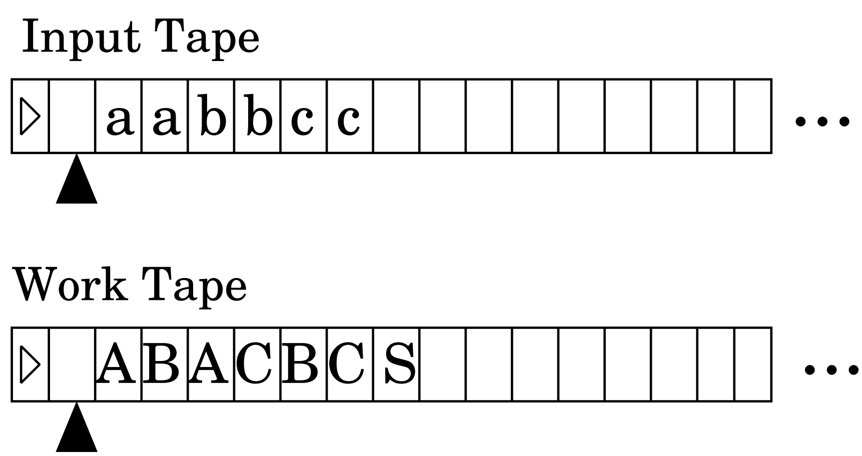
13-21: $L_{UG} \subseteq L_{re}$



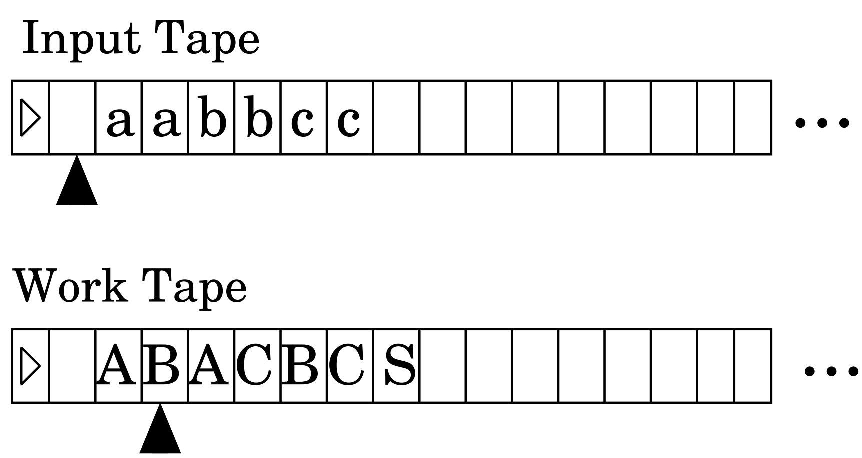
13-22: $L_{UG} \subseteq L_{re}$



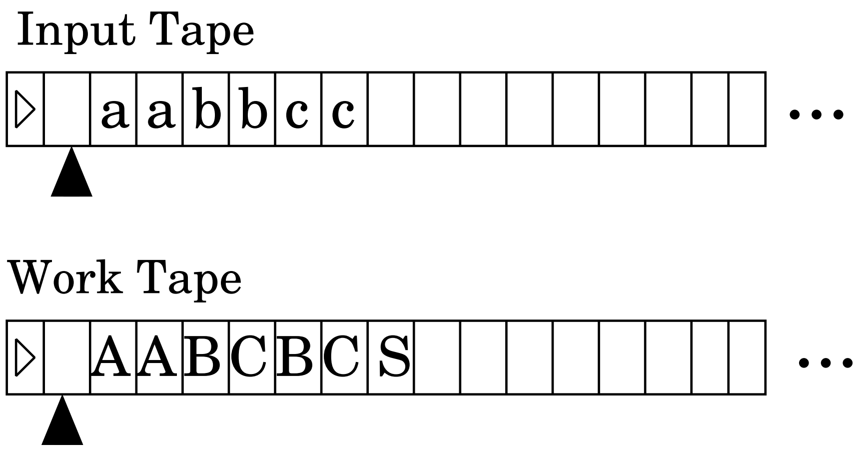
13-23: $L_{UG} \subseteq L_{re}$



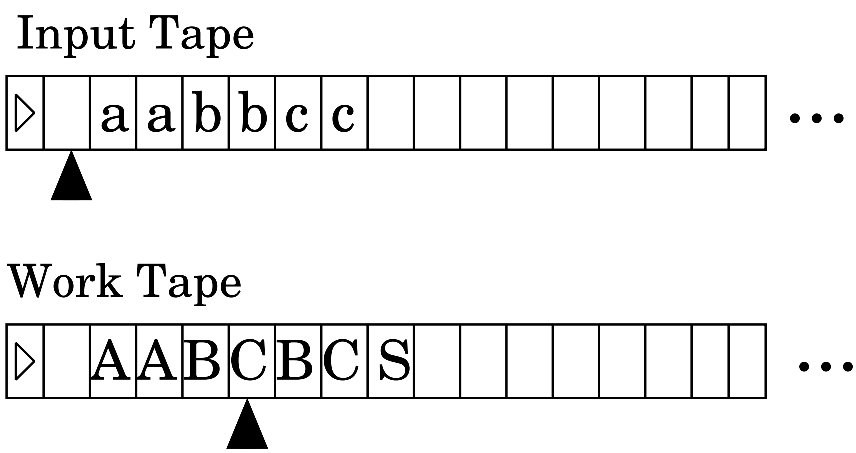
13-24: $L_{UG} \subseteq L_{re}$



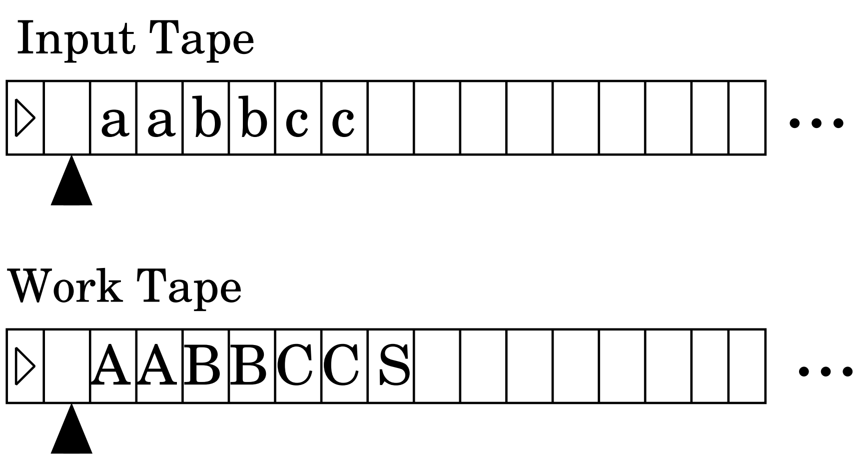
13-25: $L_{UG} \subseteq L_{re}$



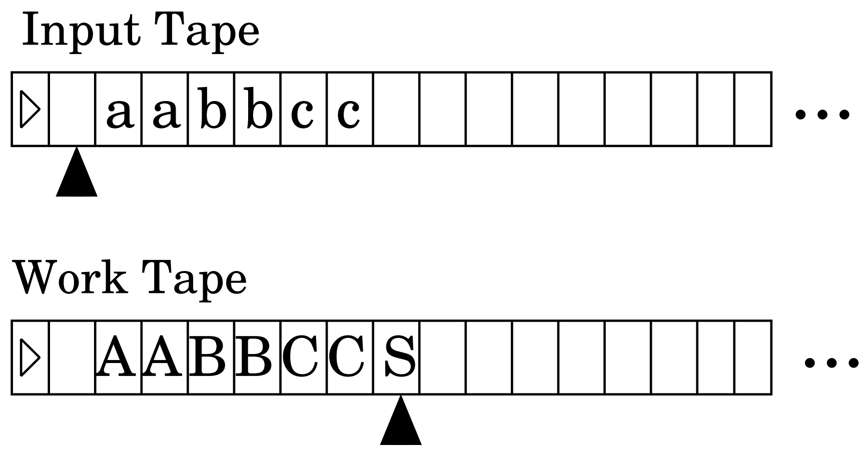
13-26: $L_{UG} \subseteq L_{re}$



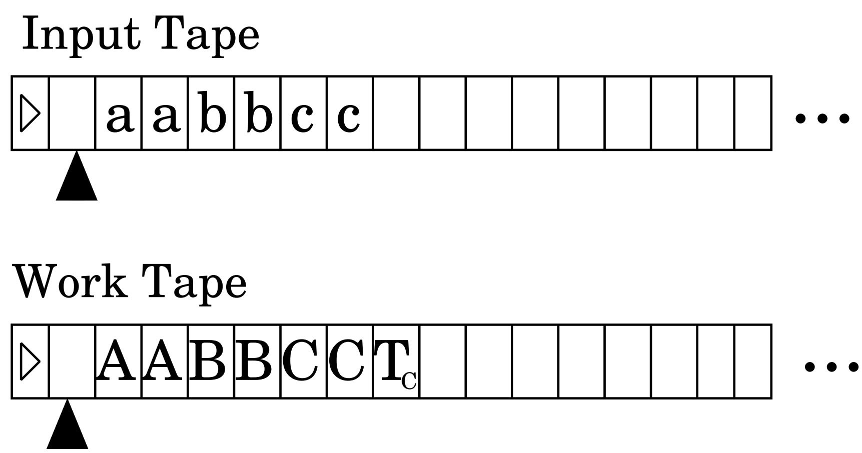
13-27: $L_{UG} \subseteq L_{re}$



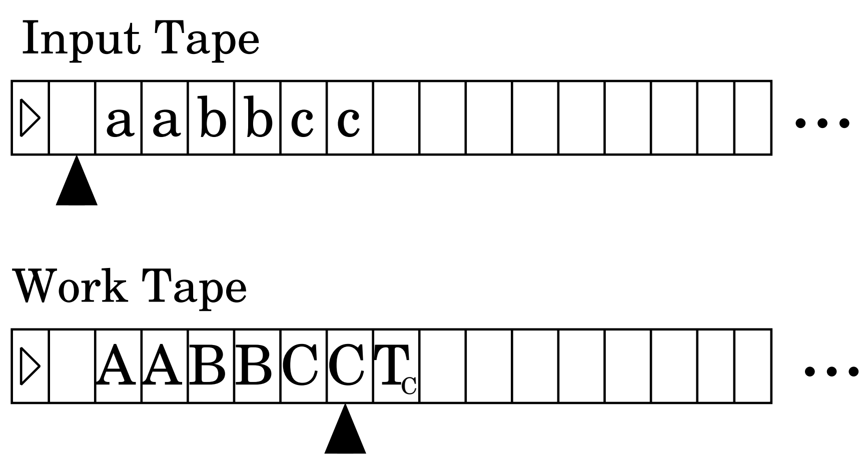
13-28: $L_{UG} \subseteq L_{re}$



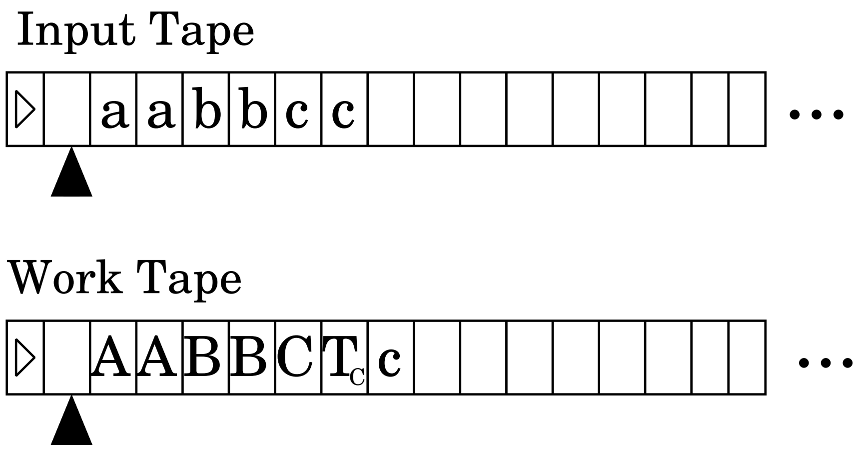
13-29: $L_{UG} \subseteq L_{re}$



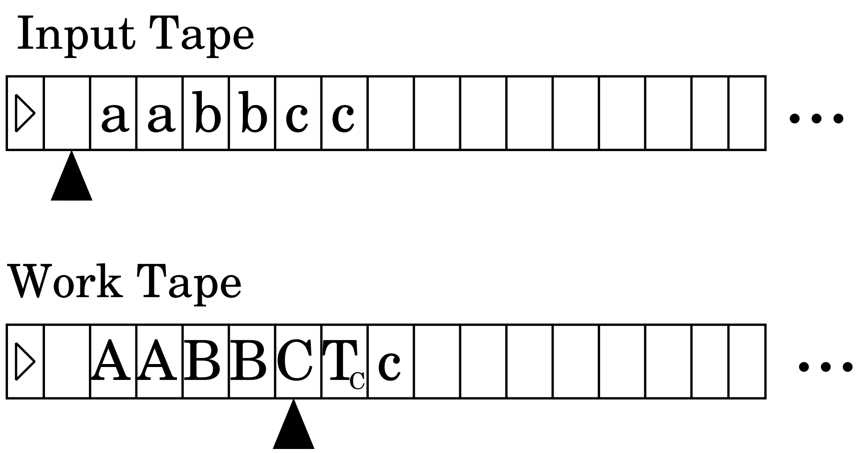
13-30: $L_{UG} \subseteq L_{re}$



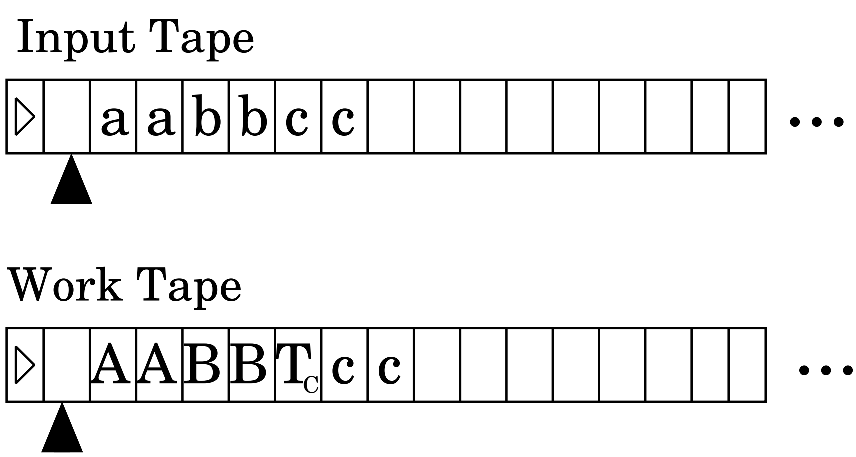
13-31: $L_{UG} \subseteq L_{re}$



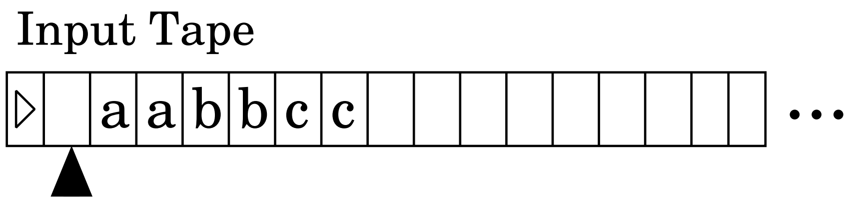
13-32: $L_{UG} \subseteq L_{re}$



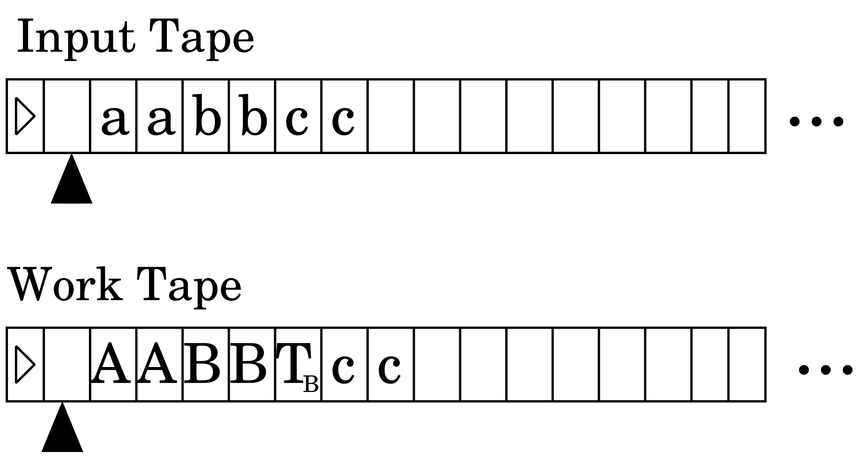
13-33: $L_{UG} \subseteq L_{re}$



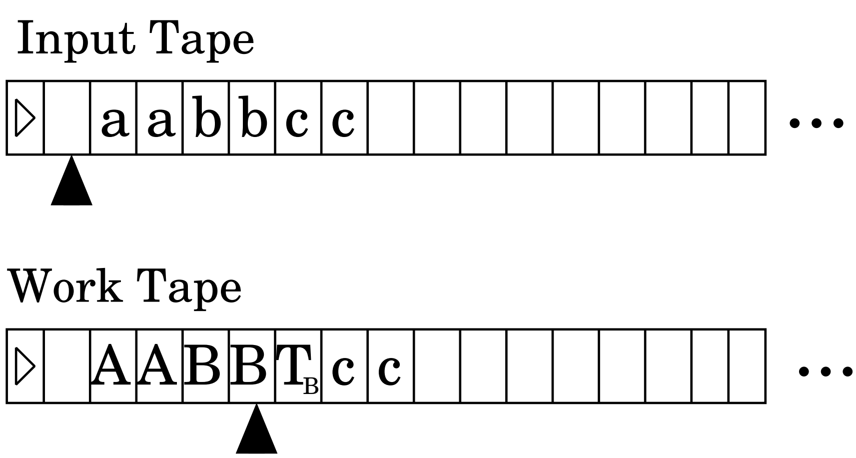
13-34: $L_{UG} \subseteq L_{re}$



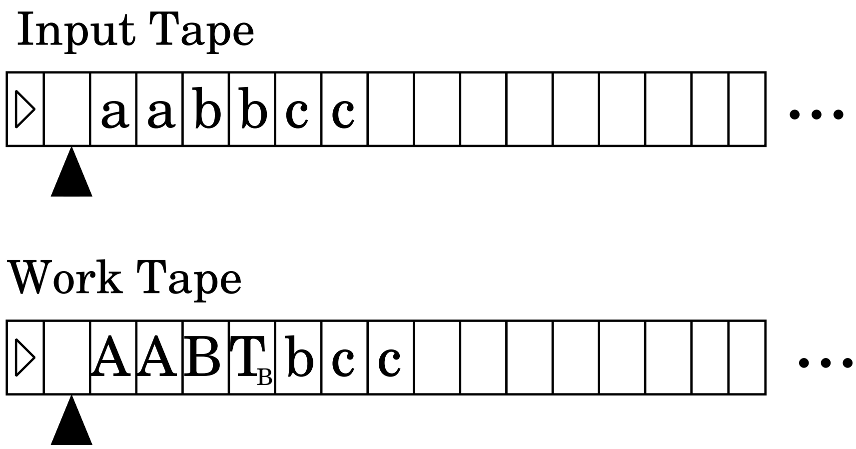
13-35: $L_{UG} \subseteq L_{re}$



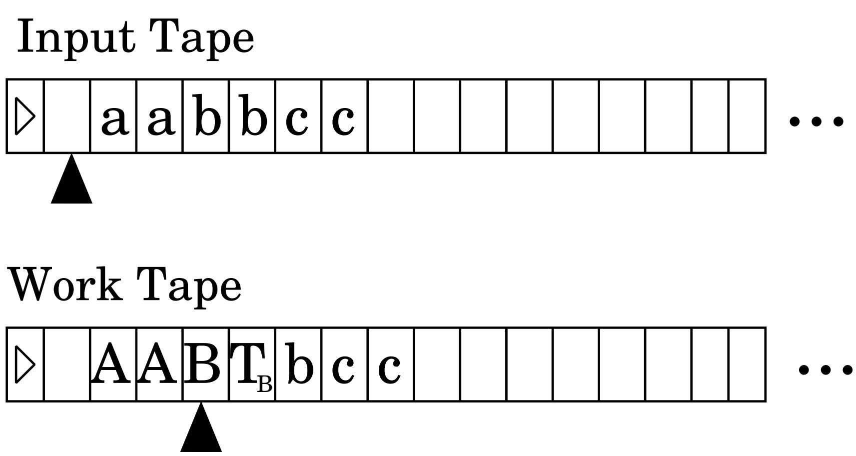
13-36: $L_{UG} \subseteq L_{re}$



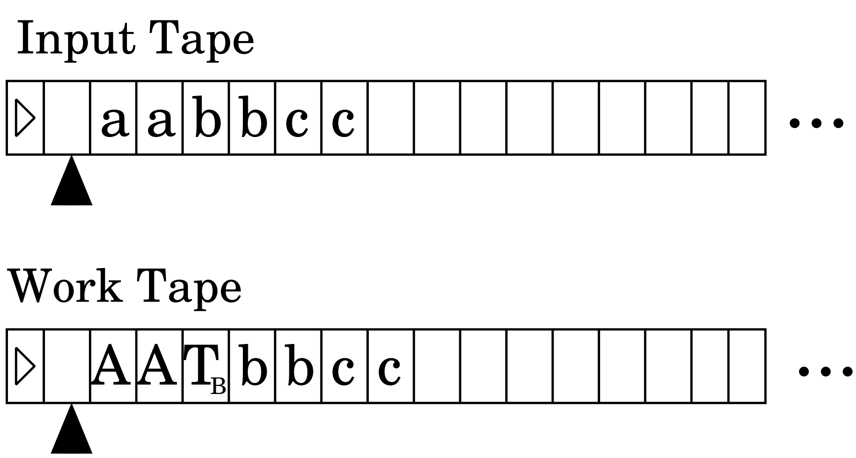
13-37: $L_{UG} \subseteq L_{re}$



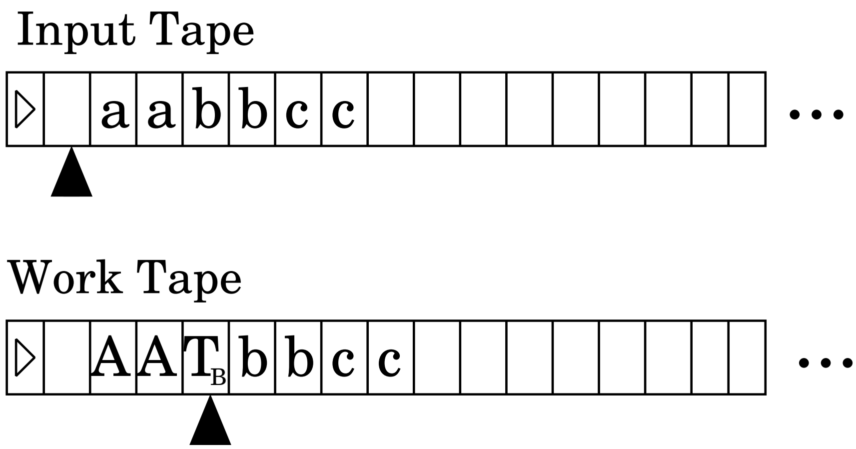
13-38: $L_{UG} \subseteq L_{re}$



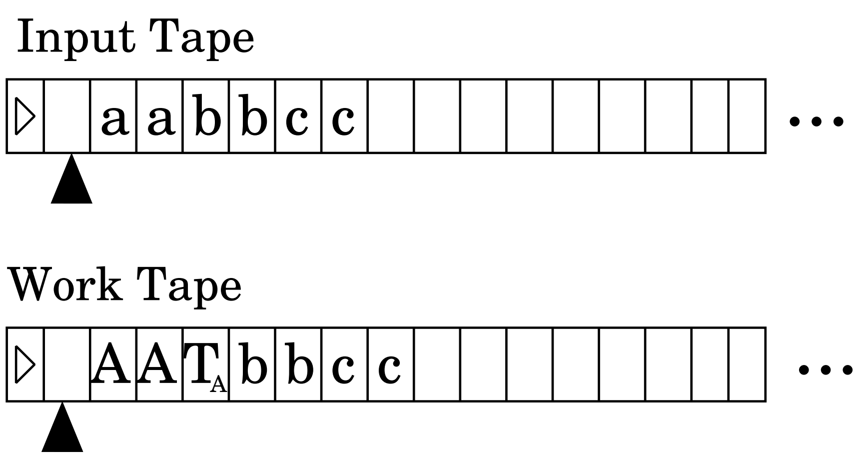
13-39: $L_{UG} \subseteq L_{re}$



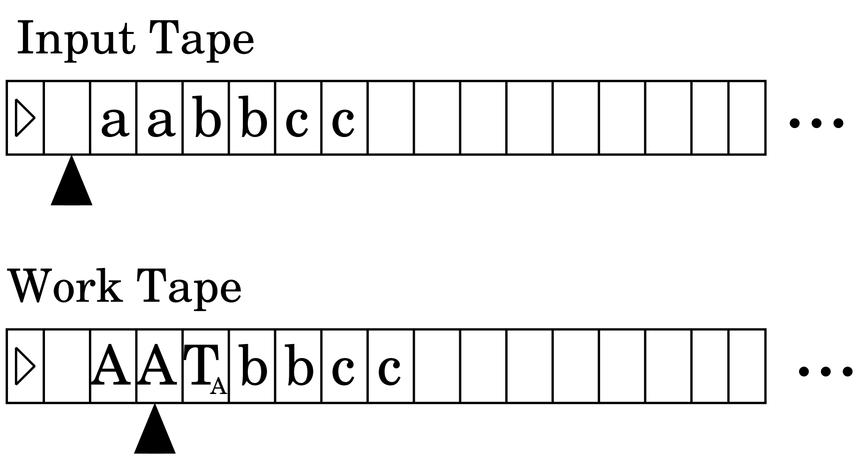
13-40: $L_{UG} \subseteq L_{re}$



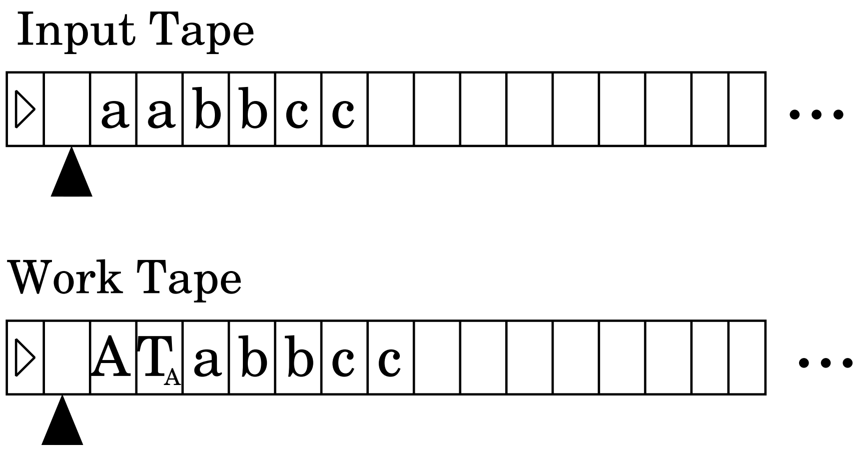
13-41: $L_{UG} \subseteq L_{re}$



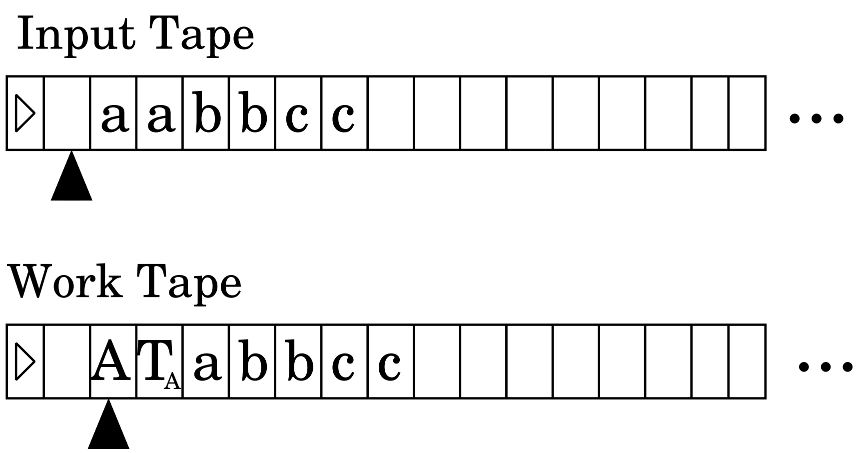
13-42: $L_{UG} \subseteq L_{re}$



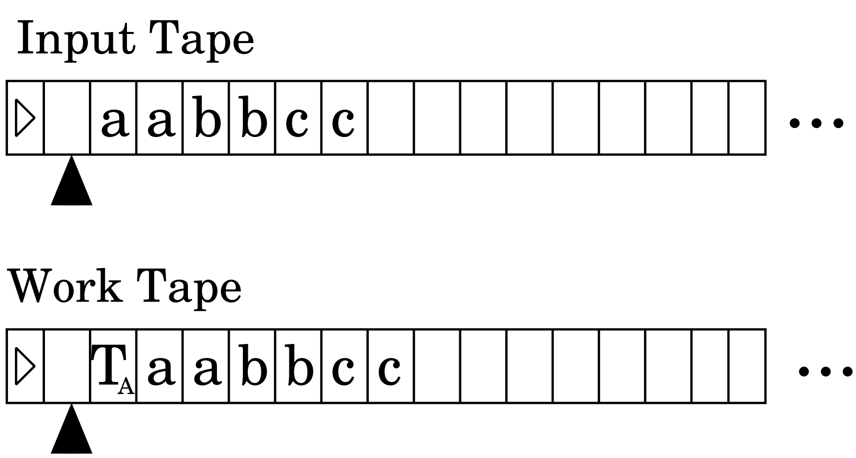
13-43: $L_{UG} \subseteq L_{re}$



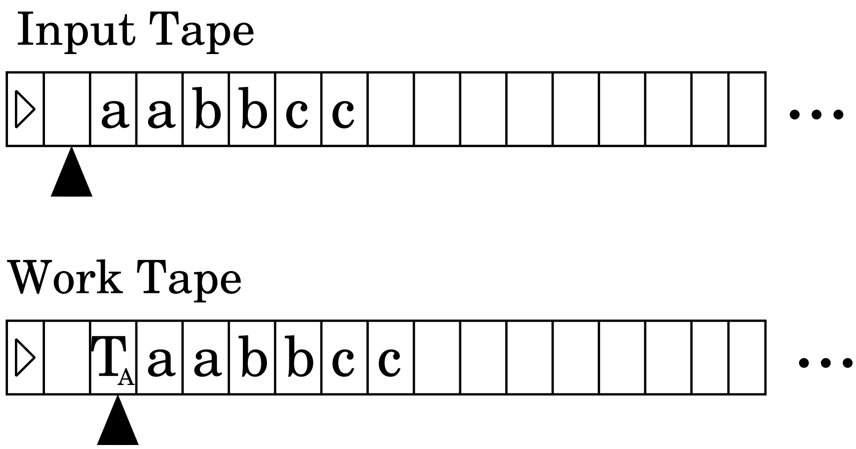
13-44: $L_{UG} \subseteq L_{re}$



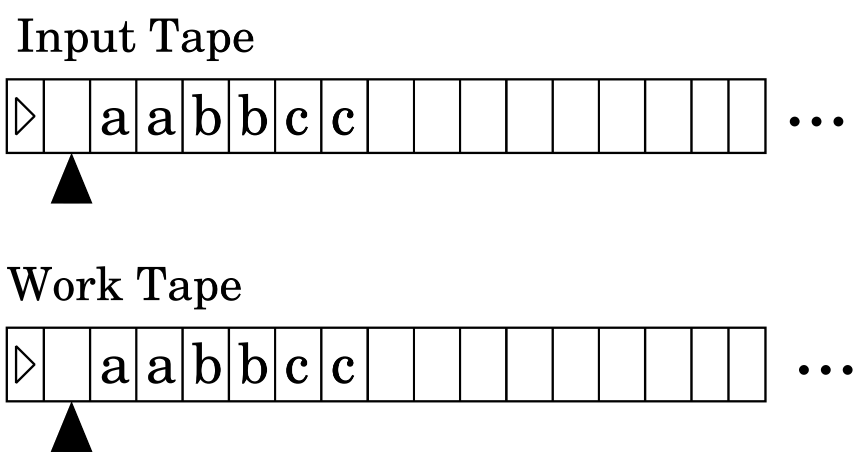
13-45: $L_{UG} \subseteq L_{re}$



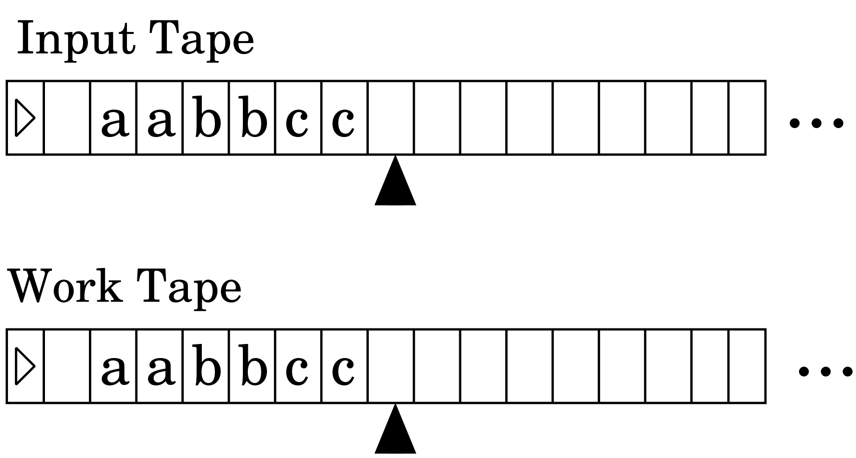
13-46: $L_{UG} \subseteq L_{re}$



13-47: $L_{UG} \subseteq L_{re}$



13-48: $L_{UG} \subseteq L_{re}$



13-49: $L_{re} \subseteq L_{UG}$

- Given any Turing Machine M that semi-decides the language L, we can create an Unrestricted Grammar G such that $L[G] = L$

13-50: $L_{re} \subseteq L_{UG}$

- Given any Turing Machine M that semi-decides the language L, we can create an Unrestricted Grammar G such that $L[G] = L$
  - Will assume that all Turing Machines accept in the same configuration: $(h, \triangleright\underline{\sqcup})$
  - Not a major restriction – why?

13-51: $L_{re} \subseteq L_{UG}$

- Given any Turing Machine M that semi-decides the language L, we can create an Unrestricted Grammar G such that $L[G] = L$
  - Will assume that all Turing Machines accept in the same configuration: $(h, \triangleright\underline{\sqcup})$
  - Not a major restriction – why?
  - Add a "tape erasing" machine right before the accepting state, that erases the tape, leaving the read/write head at the beginning of the tape

13-52: $L_{re} \subseteq L_{UG}$

- Given any Turing Machine M that semi-decides the language L, we can create an Unrestricted Grammar G such that $L[G] = L$
  - Grammar: Generates a string
  - Turing Machine: Works from string to accept state
- Two formalisms work in different directions
- Simulating Turing Machine with a Grammar can be difficult ..

13-53: $L_{re} \subseteq L_{UG}$

- Two formalisms work in different directions
  - Simulate a Turing Machine – in reverse!
  - Each partial derivation represents a configuration
  - Each rule represents a *backwards* step in Turing Machine computation

13-54: $L_{re} \subseteq L_{UG}$

- Given a TM M, we create a Grammar G:
  - Language for G:
    - Everything in Σ_M
    - Everything in K_M
    - Start symbol S
    - Symbols $\triangleright$ and $\triangleleft$

13-55: $L_{re} \subseteq L_{UG}$

- Configuration $(Q, \triangleright u\underline{a}w)$ represented by the string:

  $\triangleright uaQw\triangleleft$

For example, $(Q, \triangleright \sqcup ab\underline{c} \sqcup a)$ is represented by the string $\triangleright \sqcup abcQ \sqcup a\triangleleft$

13-56: $L_{re} \subseteq L_{UG}$

- For each element in δ_M of the form:
  - $((Q_1, a), (Q_2, b))$
- Add the rule:
  - $bQ_2 \to aQ_1$
- Remember, simulating *backwards* computation

13-57: $L_{re} \subseteq L_{UG}$

- For each element in δ_M of the form:
  - $((Q_1, a), (Q_2, \leftarrow))$
- Add the rule:
  - $Q_2 a \to aQ_1$

13-58: $L_{re} \subseteq L_{UG}$

- For each element in δ_M of the form:
  - $((Q_1, \sqcup), (Q_2, \leftarrow))$
- Add the rule
  - $Q_2 \triangleleft \to \sqcup Q_1 \triangleleft$
- (undoing erasing extra blanks)

13-59: $L_{re} \subseteq L_{UG}$

- For each element in δ_M of the form:
  - $((Q_1, a), (Q_2, \rightarrow))$
- Add the rule
  - $abQ_2 \to aQ_1 b$
- For all $b \in \Sigma$

13-60: $L_{re} \subseteq L_{UG}$

- For each element in δ_M of the form:
  - $((Q_1, a), (Q_2, \rightarrow))$
- Add the rule
  - $a \sqcup Q_2 \triangleleft \to aQ_1 \triangleleft$
- (undoing moving to the right onto unused tape)

13-61: $L_{re} \subseteq L_{UG}$

- Finally, add the rules:

- $S \rightarrow \triangleright \sqcup h \triangleleft$
- $\triangleright \sqcup Q_s \rightarrow \epsilon$
- $\triangleleft \rightarrow \epsilon$

13-62: $L_{re} \subseteq L_{UG}$

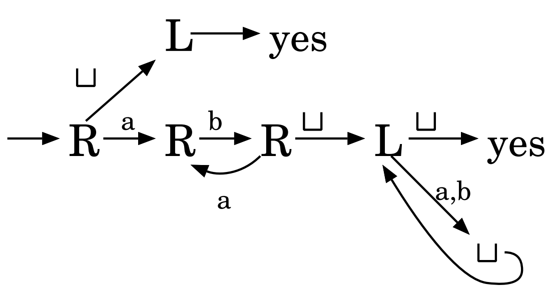
- If the Turing machine can move from
  - $\triangleright\underline{\sqcup}w$ to $\triangleright h\underline{\sqcup}$
- Then the Grammar can transform
  - $\triangleright \sqcup Q_h \triangleleft$ to $\triangleright \sqcup Q_s w \triangleleft$
- Then, remove $\triangleright \sqcup Q_s$ and $\triangleleft$ to leave w

13-63: $L_{re} \subseteq L_{UG}$

- Example:
  - Create a Turing Machine that accepts $(ab)^*$, halting in the configuration $(h, \triangleright\underline{\sqcup})$
  - (assume tape starts out as $\triangleright\underline{\sqcup}w$)

13-64: $L_{re} \subseteq L_{UG}$

- Example:
  - Create a Turing Machine that accepts $(ab)^*$, halting in the configuration $(h, \triangleright\underline{\sqcup})$



13-65: $L_{re} \subseteq L_{UG}$

	a	b	$\sqcup$
q_0	$(q_1, \rightarrow)$	$(q_1, \rightarrow)$	$(q_1, \rightarrow)$
q_1	$(q_2, \rightarrow)$		$(q_h, \leftarrow)$
q_2		$(q_3, \rightarrow)$	
q_3	$(q_2, \rightarrow)$		$(q_4, \leftarrow)$
q_4	$(q_5, \sqcup)$	$(q_5, \sqcup)$	$(q_h, \sqcup)$
q_5			$(q_4, \leftarrow)$

13-66: $L_{re} \subseteq L_{UG}$

- $((q_0, a), (q_1, \rightarrow))$
  - $aaQ_1 \rightarrow aQ_0a$
  - $abQ_1 \rightarrow aQ_0b$
  - $a \sqcup Q_1 \rightarrow aQ_0\sqcup$

- $a \sqcup Q_1 \triangleleft \rightarrow aQ_0 \triangleleft$

13-67: $L_{re} \subseteq L_{UG}$

- $((q_0, b), (q_1, \rightarrow))$
  - $baQ_1 \rightarrow bQ_0a$
  - $bbQ_1 \rightarrow bQ_0b$
  - $b \sqcup Q_1 \rightarrow bQ_0 \sqcup$
  - $b \sqcup Q_1 \triangleleft \rightarrow bQ_0 \triangleleft$

13-68: $L_{re} \subseteq L_{UG}$

- $((q_0, \sqcup), (q_1, \rightarrow))$
  - $\sqcup aQ_1 \rightarrow \sqcup Q_0a$
  - $\sqcup bQ_1 \rightarrow \sqcup Q_0b$
  - $\sqcup \sqcup Q_1 \rightarrow \sqcup Q_0 \sqcup$
  - $\sqcup \sqcup Q_1 \triangleleft \rightarrow \sqcup Q_0 \triangleleft$

13-69: $L_{re} \subseteq L_{UG}$

- $((q_1, a), (q_2, \rightarrow))$
  - $aaQ_2 \rightarrow aQ_1a$
  - $abQ_2 \rightarrow aQ_1b$
  - $a \sqcup Q_2 \rightarrow aQ_1 \sqcup$
  - $a \sqcup Q_2 \triangleleft \rightarrow aQ_1 \triangleleft$

13-70: $L_{re} \subseteq L_{UG}$

- $((q_1, \sqcup), (q_h, \leftarrow))$
  - $h \sqcup \rightarrow \sqcup Q_1$

13-71: $L_{re} \subseteq L_{UG}$

- $((q_2, b), (q_3, \rightarrow))$
  - $baQ_3 \rightarrow bQ_2a$
  - $bbQ_3 \rightarrow bQ_2b$
  - $b \sqcup Q_3 \rightarrow bQ_2 \sqcup$
  - $b \sqcup Q_3 \triangleleft \rightarrow bQ_2 \triangleleft$

13-72: $L_{re} \subseteq L_{UG}$

- $((q_3, a), (q_4, \rightarrow))$
  - $aaQ_4 \rightarrow aQ_3a$
  - $abQ_4 \rightarrow aQ_3b$
  - $a \sqcup Q_4 \rightarrow aQ_3 \sqcup$

- $a \sqcup Q_4 \triangleleft \rightarrow aQ_3 \triangleleft$

13-73: $L_{re} \subseteq L_{UG}$

- $((q_4, a), (q_5, \sqcup))$
  - $\sqcup Q_5 \rightarrow aQ_4$
- $((q_4, b), (q_5, \sqcup))$
  - $\sqcup Q_5 \rightarrow bQ_4$
- $((q_4, \sqcup), (q_h, \sqcup))$
  - $\sqcup h \rightarrow \sqcup Q_4$
- $((q_5, \sqcup), (q_4, \leftarrow))$
  - $Q_4 \sqcup \rightarrow \sqcup Q_5$

13-74: $L_{re} \subseteq L_{UG}$

$S \rightarrow \triangleright \sqcup h \triangleleft$	$\sqcup aQ_1 \rightarrow \sqcup Q_0 a$	$b \sqcup Q_3 \rightarrow bQ_2 \sqcup$
$\triangleright \sqcup Q_0 \rightarrow \epsilon$	$\sqcup bQ_1 \rightarrow \sqcup Q_0 b$	$b \sqcup Q_3 \triangleleft \rightarrow bQ_2 \triangleleft$
$\triangleleft \rightarrow \epsilon$	$\sqcup \sqcup Q_1 \rightarrow \sqcup Q_0 \sqcup$	$aaQ_4 \rightarrow aQ_3 a$
$aaQ_1 \rightarrow aQ_0 a$	$\sqcup \sqcup Q_1 \triangleleft \rightarrow \sqcup Q_0 \triangleleft$	$abQ_4 \rightarrow aQ_3 b$
$abQ_1 \rightarrow aQ_0 b$	$aaQ_2 \rightarrow aQ_1 a$	$a \sqcup Q_4 \rightarrow aQ_3 \sqcup$
$a \sqcup Q_1 \rightarrow aQ_0 \sqcup$	$abQ_2 \rightarrow aQ_1 b$	$a \sqcup Q_4 \triangleleft \rightarrow aQ_3 \triangleleft$
$a \sqcup Q_1 \triangleleft \rightarrow aQ_0 \triangleleft$	$a \sqcup Q_2 \rightarrow aQ_1 \sqcup$	$\sqcup Q_5 \rightarrow aQ_4$
$baQ_1 \rightarrow bQ_0 a$	$a \sqcup Q_2 \triangleleft \rightarrow aQ_1 \triangleleft$	$\sqcup Q_5 \rightarrow bQ_4$
$bbQ_1 \rightarrow bQ_0 b$	$h \sqcup \rightarrow \sqcup Q_1$	$\sqcup h \rightarrow \sqcup Q_4$
$b \sqcup Q_1 \rightarrow bQ_0 \sqcup$	$baQ_3 \rightarrow bQ_2 a$	$Q_4 \sqcup \rightarrow \sqcup Q_5$
$b \sqcup Q_1 \triangleleft \rightarrow bQ_0 \triangleleft$	$bbQ_3 \rightarrow bQ_2 b$	

13-75: $L_{re} \subseteq L_{UG}$

- Generating $abab$

$$\begin{array}{rcl}
S & \Rightarrow & \triangleright \sqcup h \triangleleft \\
\triangleright \sqcup h \triangleleft & \Rightarrow & \triangleright \sqcup Q_4 \triangleleft \\
\triangleright \sqcup Q_4 \triangleleft & \Rightarrow & \triangleright \sqcup \sqcup Q_5 \triangleleft \\
\triangleright \sqcup \sqcup Q_5 \triangleleft & \Rightarrow & \triangleright \sqcup aQ_4 \triangleleft \\
\triangleright \sqcup aQ_4 \triangleleft & \Rightarrow & \triangleright \sqcup a \sqcup Q_5 \triangleleft \\
\triangleright \sqcup a \sqcup Q_5 \triangleleft & \Rightarrow & \triangleright \sqcup abQ_4 \triangleleft \\
\triangleright \sqcup abQ_4 \triangleleft & \Rightarrow & \triangleright \sqcup ab \sqcup Q_5 \triangleleft \\
\triangleright \sqcup ab \sqcup Q_5 \triangleleft & \Rightarrow & \triangleright \sqcup abaQ_4 \triangleleft \\
\triangleright \sqcup abaQ_4 \triangleleft & \Rightarrow & \triangleright \sqcup aba \sqcup Q_5 \triangleleft \\
\triangleright \sqcup aba \sqcup Q_5 \triangleleft & \Rightarrow & \triangleright \sqcup ababQ_4 \triangleleft
\end{array}$$

13-76: $L_{re} \subseteq L_{UG}$

- Generating $abab$

$$\begin{array}{rcl}
\triangleright \sqcup ababQ_4 \triangleleft & \Rightarrow & \triangleright \sqcup abab \sqcup Q_3 \triangleleft \\
\triangleright \sqcup abab \sqcup Q_3 \triangleleft & \Rightarrow & \triangleright \sqcup ababQ_2 \triangleleft \\
\triangleright \sqcup ababQ_2 \triangleleft & \Rightarrow & \triangleright \sqcup abaQ_3 b \triangleleft \\
\triangleright \sqcup abaQ_3 b \triangleleft & \Rightarrow & \triangleright \sqcup abQ_2 ab \triangleleft \\
\triangleright \sqcup abQ_2 ab \triangleleft & \Rightarrow & \triangleright \sqcup aQ_1 bab \triangleleft \\
\triangleright \sqcup aQ_1 bab \triangleleft & \Rightarrow & \triangleright \sqcup Q_0 abab \triangleleft \\
\triangleright \sqcup Q_0 abab \triangleleft & \Rightarrow & abab \triangleright \\
abab \triangleleft & \Rightarrow & abab
\end{array}$$