Closure Claim Review, Turing Machines

CSE 105 Week 6 Discussion

Deadlines and Logistics

- Test 1 this week
- Do review quizzes on PrairieLearn
- HW 4 due Thur 2/20/25 at 5pm (late submission open until 8am next morning)

Closure claims
for regular and context-free languages

Closure claim summary

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

Idea (informal):

- DFA flip accept/non-accept status of the states (week 2 notes page 7-8)

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

$Q_1 \times Q_2$

Idea (informal):

$F_1 \times Q_2 \cup Q_1 \times F_2$

- DFA "parallel computation" (week 3 notes page 4)
- NFA construction

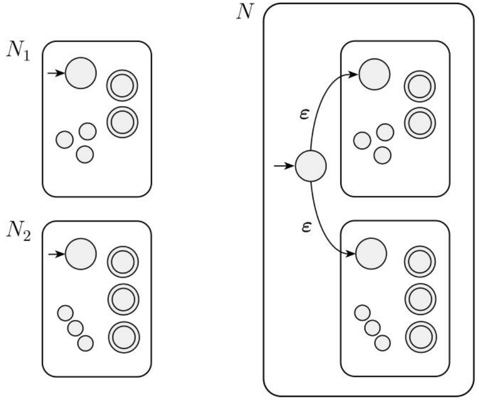


FIGURE 1.46
Construction of an NFA N to recognize $A_1 \cup A_2$

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

Sipser Figure 1.46, Pg 59

Idea (informal):

- DFA “parallel computation” (week 3 notes page 5)

$Q_1 \times Q_2$

$F_1 \times F_2$

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

Idea (informal):

- NFA construction

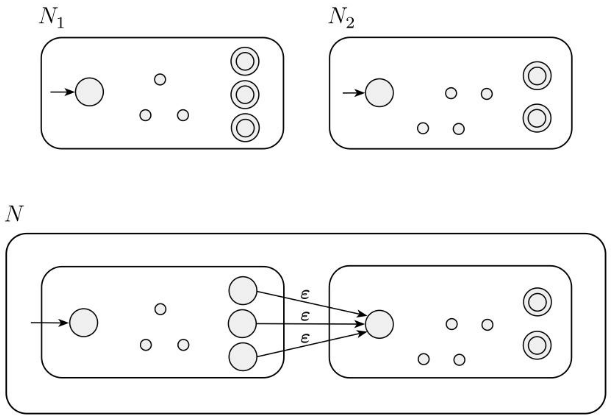


FIGURE 1.48
Construction of N to recognize $A_1 \circ A_2$

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

Sipser Figure 1.48, Pg 61

Idea (informal):

- NFA construction

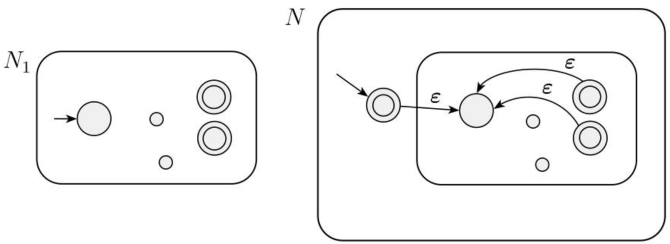


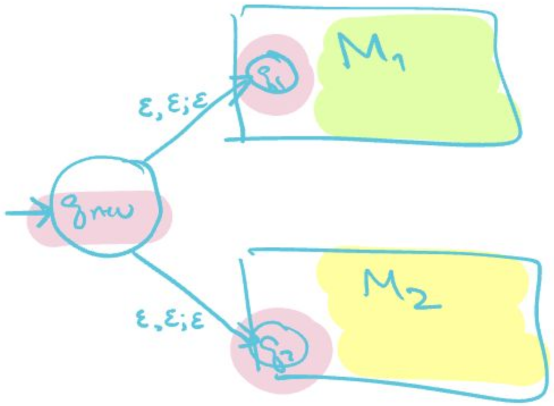
FIGURE 1.50
Construction of N to recognize A^*

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

Sipser Figure 1.50, Pg 62

Idea (informal):

- CFG $S_{new} \to S_1 \mid S_2$
- PDA construction



True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

Idea (informal):

- CFG $S_{new} \to S_1 S_2$
- PDA construction, similar to the NFA union idea but need "stack clean-up" (HW4 Q2c)

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

Idea (informal):

- CFG

(V, Σ, R, S)

$S_{new} \notin V$

$(V \cup \{S_{new}\}, \Sigma, R \cup \{S_{new} \to \varepsilon \mid S_{new} S\}, S_{new})$

True	The class of regular languages over Σ is closed under complementation.
True	The class of regular languages over Σ is closed under union.
True	The class of regular languages over Σ is closed under intersection.
True	The class of regular languages over Σ is closed under concatenation.
True	The class of regular languages over Σ is closed under Kleene star.
FALSE	The class of context-free languages over Σ is closed under complementation.
True	The class of context-free languages over Σ is closed under union.
FALSE	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation.
True	The class of context-free languages over Σ is closed under Kleene star.

FALSE	The class of context-free languages over Σ is closed under intersection.

Idea (informal):

- Counterexample:

Over the alphabet $\{a, b, c\}$, we have languages

$$A = \{a^n b^n c^m \mid m, n \geq 0\}$$

$$B = \{a^m b^n c^n \mid m, n \geq 0\}$$

Both A and B are context-free languages. However,

$$A \cap B = \{a^n b^n c^n \mid n \geq 0\}$$

is not context-free (which can be proved by using the pumping lemma for context-free languages)

FALSE	The class of context-free languages over Σ is closed under complementation.

Idea (informal):

- De Morgan's Law & proof by contradiction:

Assume context-free languages are closed under complementation. Consider context-free languages A and B. Then $\overline{A}$ and $\overline{B}$ should also be context-free. Since context-free languages are closed under union, we further get $\overline{A} \cup \overline{B}$ should also be context-free. Then $\overline{\overline{A} \cup \overline{B}}$ should also be context-free. By De Morgan's Law,

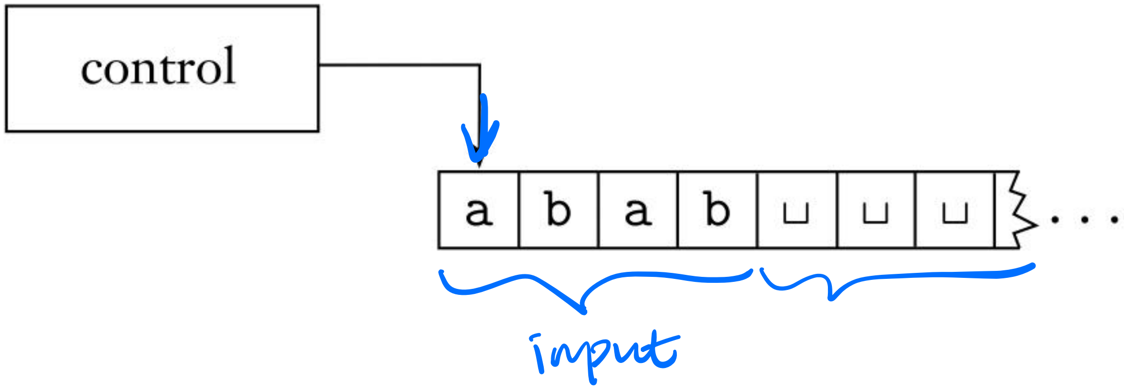
$$A \cap B = \overline{\overline{A} \cup \overline{B}}$$

, thus we conclude that $A \cap B$ is context-free. However, we proved that context-free languages are not closed under intersection, so here we arrive at a contradiction. Therefore context-free languages are not closed under complementation.

Turing Machines

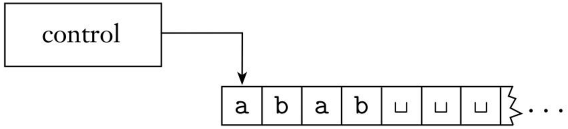
Turing machines

- Uses an infinite tape as its **unlimited memory**
- Has a tape head that can **read** and **write** symbols and **move around** on tape
- A Turing machine may loop on some input



Sipser Figure 3.1, Pg 166

Formal definition of Turing machines



DEFINITION 3.3

A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta \colon Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{L}, \text{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

deterministic:

state, read symbol

↓

next state, write symbol, L/R

→ take effect immediately!

Sipser Definition 3.3, Pg 168

Conventions

- State diagram edge labels meaning and abbreviations L/R
  - $b \rightarrow X, R$ means "read b from tape, write X onto tape, and move tape head Right"
  - you can either use ; or $\rightarrow$ in edge labels
  - $b \rightarrow R$ means $b \rightarrow b, R$
  - $a, b, c \rightarrow L$ means $a \rightarrow a, L$ and $b \rightarrow b, L$ and $c \rightarrow c, L$
- You can either use ⊔ or □ for blank symbol

Conventions continued

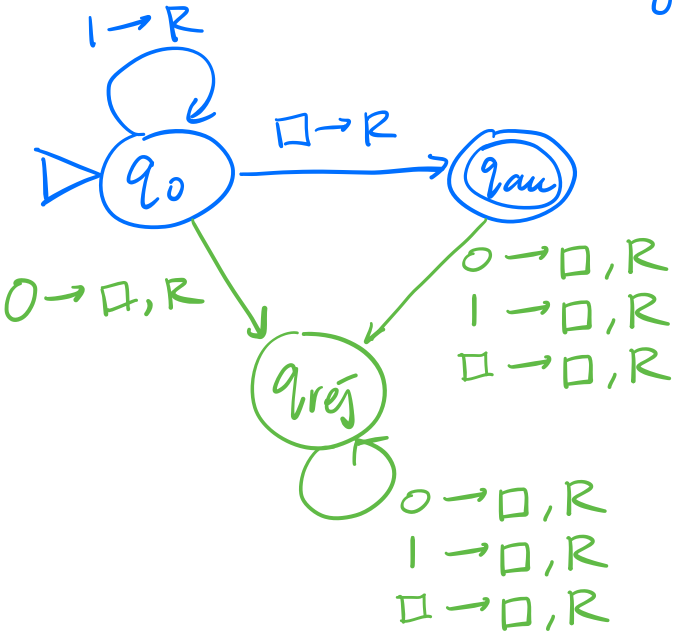
- The reject state qrej might not appear in the state diagram, and any missing transitions in the state diagram have value (qrej, □, R).

next state, write, move

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \square\}$

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

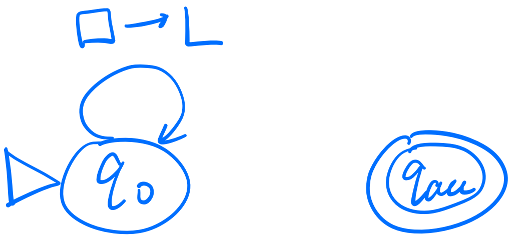


Conventions continued

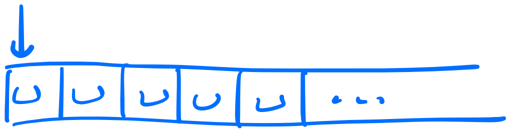
- If the tape head is already at the leftmost position on the tape, and a transition says to move **L** (left), we do the transition and stay at the leftmost position

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \square\}$



input: ε



loop on ε

reject others

Turing machine computation

Now we give the formal description of $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0\}$, and
- $\Gamma = \{0, x, ␣\}$.
- We describe δ with a state diagram (see Figure 3.8).
- The start, accept, and reject states are q_1, q_{accept}, and q_{reject}, respectively.

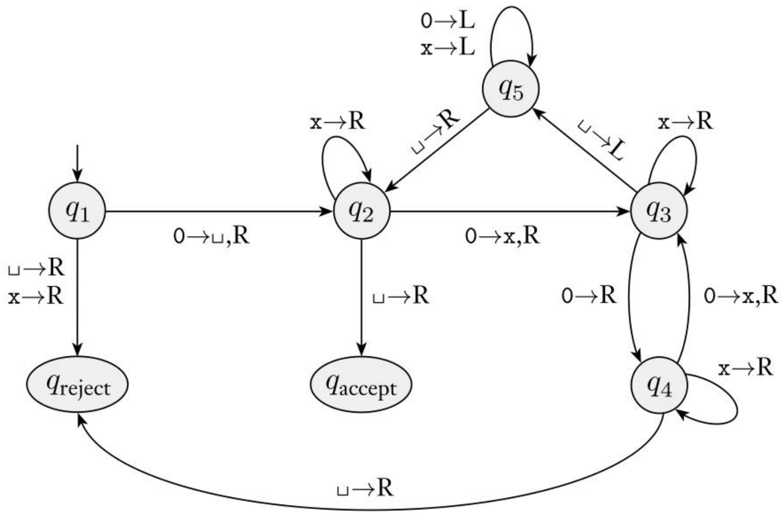


FIGURE 3.8
State diagram for Turing machine M_2

Sipser Pg 171-172

Turing machine computation

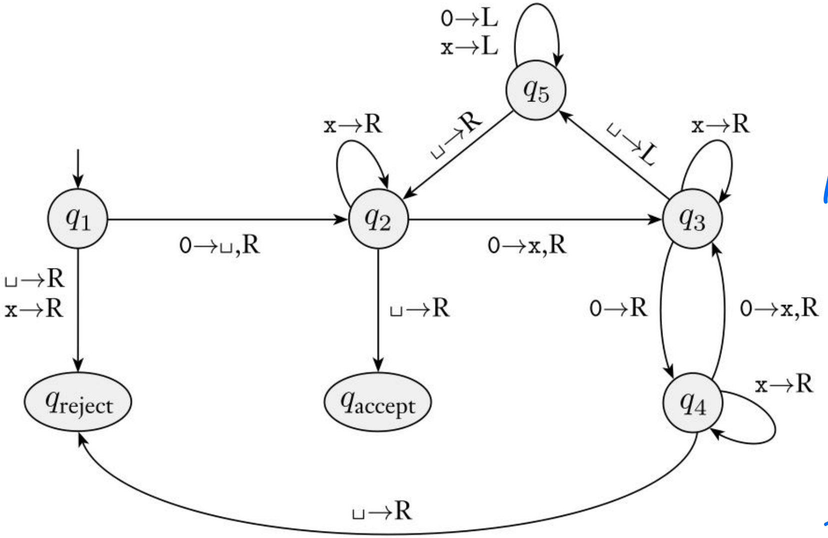
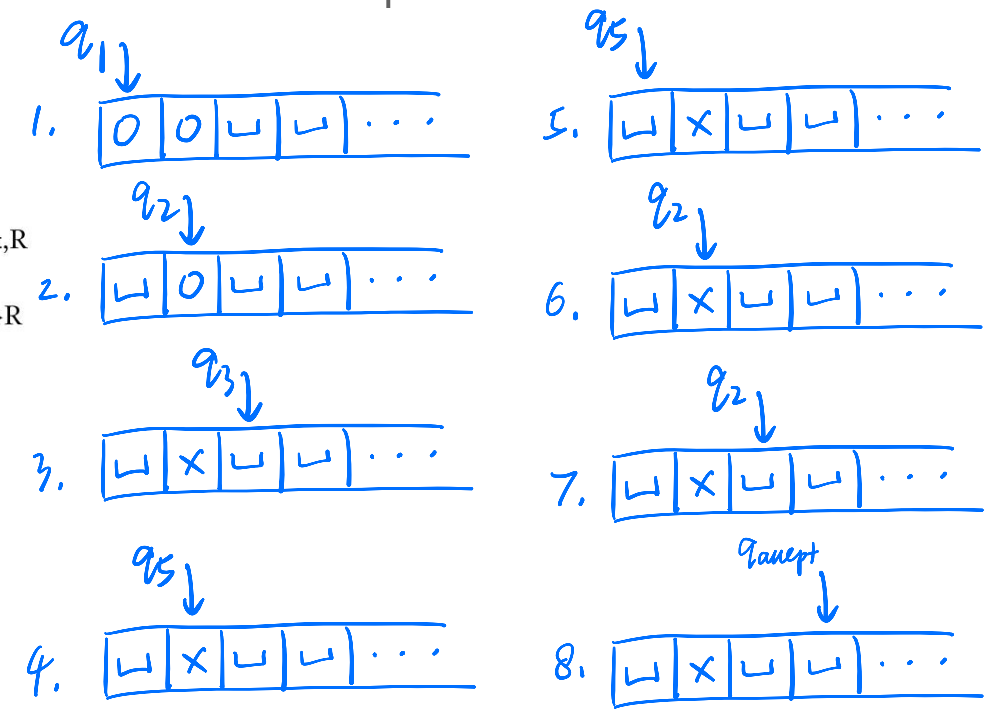


FIGURE 3.8
State diagram for Turing machine M_2

Trace the computation on 00



00 accepted

Sipser Pg 171-172

Describing Turing Machines

Describing Turing machines (Sipser p. 185) To define a Turing machine, we could give a

- **Formal definition**: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or,
- **Implementation-level definition**: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.
- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

Implementation-level description

Here we describe a Turing machine (TM) M_2 that decides $A = \{0^{2^n} | n \geq 0\}$, the language consisting of all strings of 0s whose length is a power of 2.

M_2 = "On input string w:

1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1 the tape contained a single 0, *accept*.
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, *reject*.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1."

Sipser Example 3.7, Pg 171

High-level description

Suppose M_1 and M_2 are Turing machines. Consider the Turing machines given by the high-level descriptions:

M = "On input w,   Loops at step 1 if M_1 loops on w

1. Run M_1 on input w. If M_1 accepts w, accept. If M_1 rejects w, go to 2.
2. Run M_2 on input w. If M_2 accepts w, accept. If M_2 rejects w, reject."

For each of the following claims, answer **Always true** if the statement is true for all possible M_1 and M_2; answer **Always false** if the statement is false for all possible M_1 and M_2; and answer Neither otherwise.

M_2 accepts w.

- If $w \in L(M_1)$ then $w \in L(M)$.
- If $w \in L(M_2)$ then $w \in L(M)$.
- If $w \notin L(M_1)$ then $w \notin L(M)$.
- If $w \notin L(M_2)$ then $w \notin L(M)$.

If M_1 loops on w, M loops on w. $w \notin L(M)$

Turing-recognizable and Turing decidable

- **Deciders** are Turing machines that halt on all inputs; they never loop; they always make a decision to accept or reject
- Call a language **Turing-recognizable** if some Turing machine recognizes it
- Call a language **Turing-decidable** if some decider decides it