3.4

Product Construction
**Question:** Are languages accepted by **DFA**s closed under union? That is, given **DFA**s $M_1$ and $M_2$ is there a **DFA** that accepts $L(M_1) \cup L(M_2)$?

How about intersection $L(M_1) \cap L(M_2)$?

Idea from programming: on input string $w$

- Simulate $M_1$ on $w$
- Simulate $M_2$ on $w$
- If both accept then $w \in L(M_1) \cap L(M_2)$. If at least one accepts then $w \in L(M_1) \cup L(M_2)$.
- **Catch:** We want a single **DFA** $M$ that can only read $w$ once.
- **Solution:** Simulate $M_1$ and $M_2$ in parallel by keeping track of states of both machines.
Question: Are languages accepted by DFA's closed under union? That is, given DFA's $M_1$ and $M_2$ is there a DFA that accepts $L(M_1) \cup L(M_2)$? How about intersection $L(M_1) \cap L(M_2)$?

Idea from programming: on input string $w$

- Simulate $M_1$ on $w$
- Simulate $M_2$ on $w$
- If both accept then $w \in L(M_1) \cap L(M_2)$. If at least one accepts then $w \in L(M_1) \cup L(M_2)$.

Catch: We want a single DFA $M$ that can only read $w$ once.

Solution: Simulate $M_1$ and $M_2$ in parallel by keeping track of states of both machines.
Union and Intersection

**Question:** Are languages accepted by DFAs closed under union? That is, given DFAs $M_1$ and $M_2$ is there a DFA that accepts $L(M_1) \cup L(M_2)$? How about intersection $L(M_1) \cap L(M_2)$?

Idea from programming: on input string $w$
- Simulate $M_1$ on $w$
- Simulate $M_2$ on $w$
- If both accept than $w \in L(M_1) \cap L(M_2)$. If at least one accepts then $w \in L(M_1) \cup L(M_2)$.
- **Catch:** We want a single DFA $M$ that can only read $w$ once.
- **Solution:** Simulate $M_1$ and $M_2$ in parallel by keeping track of states of both machines.
Union and Intersection

**Question:** Are languages accepted by DFAs closed under union? That is, given DFAs $M_1$ and $M_2$ is there a DFA that accepts $L(M_1) \cup L(M_2)$?

How about intersection $L(M_1) \cap L(M_2)$?

Idea from programming: on input string $w$

- Simulate $M_1$ on $w$
- Simulate $M_2$ on $w$
- If both accept than $w \in L(M_1) \cap L(M_2)$. If at least one accepts then $w \in L(M_1) \cup L(M_2)$.
- **Catch:** We want a single DFA $M$ that can only read $w$ once.
- **Solution:** Simulate $M_1$ and $M_2$ in parallel by keeping track of states of both machines
Example

\[ M_1 \] accepts \( \#0 = \text{odd} \)

\[ M_2 \] accepts \( \#1 = \text{odd} \)
Example

M_1 accepts #0 = odd

M_2 accepts #1 = odd

**Cross-product machine**
Example II

Accept all binary strings of length divisible by 3 and 5
Product construction for intersection

\[ M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \text{ and } M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \]

Create \( M = (Q, \Sigma, \delta, s, A) \) where

- \( Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \rightarrow Q \) where
  \[
  \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))
  \]
- \( A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\} \)

**Theorem**

\[ L(M) = L(M_1) \cap L(M_2). \]
Product construction for intersection

\( M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \) and \( M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \)

Create \( M = (Q, \Sigma, \delta, s, A) \) where

- \( Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \rightarrow Q \) where
  \[
  \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))
  \]
- \( A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\} \)

**Theorem**

\( L(M) = L(M_1) \cap L(M_2). \)
Product construction for intersection

\[ M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \] and \[ M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \]

Create \[ M = (Q, \Sigma, \delta, s, A) \] where

- \( Q = Q_1 \times Q_2 = \{ (q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2 \} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \rightarrow Q \) where
  \[
  \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))
  \]

- \( A = A_1 \times A_2 = \{ (q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2 \} \)

Theorem

\[ L(M) = L(M_1) \cap L(M_2). \]
Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \rightarrow Q$ where
  $$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$
- $A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\}$

Theorem

$L(M) = L(M_1) \cap L(M_2)$. 
Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \rightarrow Q$ where
  \[ \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \]
- $A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\}$

Theorem

$L(M) = L(M_1) \cap L(M_2)$. 
Product construction for intersection

$M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2)$

Create $M = (Q, \Sigma, \delta, s, A)$ where

- $Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $s = (s_1, s_2)$
- $\delta : Q \times \Sigma \rightarrow Q$ where
  \[
  \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))
  \]
- $A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\}$

**Theorem**

$L(M) = L(M_1) \cap L(M_2)$. 
Product construction for intersection

\[ M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \text{ and } M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \]

Create \( M = (Q, \Sigma, \delta, s, A) \) where

- \( Q = Q_1 \times Q_2 = \\{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \rightarrow Q \) where
  \[
  \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))
  \]
- \( A = A_1 \times A_2 = \\{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\} \)

Theorem

\[ L(M) = L(M_1) \cap L(M_2). \]
Product construction for intersection

\( M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \) and \( M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \)

Create \( M = (Q, \Sigma, \delta, s, A) \) where

- \( Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \rightarrow Q \) where
  \[ \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \]
- \( A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\} \)

**Theorem**

\( L(M) = L(M_1) \cap L(M_2). \)
Product construction for intersection

\[ M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \text{ and } M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \]

Create \( M = (Q, \Sigma, \delta, s, A) \) where

- \( Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \rightarrow Q \) where
  \[
  \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))
  \]
- \( A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\} \)

**Theorem**

\[ L(M) = L(M_1) \cap L(M_2). \]
Product construction for intersection

\[ M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \text{ and } M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \]

Create \( M = (Q, \Sigma, \delta, s, A) \) where

- \( Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \rightarrow Q \) where
  \[
  \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))
  \]
- \( A = A_1 \times A_2 = \{(q_1, q_2) \mid q_1 \in A_1, q_2 \in A_2\} \)

**Theorem**

\[ L(M) = L(M_1) \cap L(M_2). \]
Correctness of construction

Lemma

For each string $w$, $\delta^*(s, w) = (\delta_1^*(s_1, w), \delta_2^*(s_2, w))$.

Exercise: Assuming lemma prove the theorem in previous slide. Proof of lemma by induction on $|w|$. 
Correctness of construction

**Lemma**

For each string $w$, $\delta^*(s, w) = (\delta_1^*(s_1, w), \delta_2^*(s_2, w))$.

**Exercise:** Assuming lemma prove the theorem in previous slide. Proof of lemma by induction on $|w|$. 
Correctness of construction

**Lemma**

For each string \( w \), \( \delta^*(s, w) = (\delta_1^*(s_1, w), \delta_2^*(s_2, w)) \).

**Exercise:** Assuming lemma prove the theorem in previous slide. Proof of lemma by induction on \( |w| \)
Product construction for union

\[ M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \text{ and } M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \]

Create \( M = (Q, \Sigma, \delta, s, A) \) where

- \( Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \to Q \) where
  \[ \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \]
- \( A = \{(q_1, q_2) \mid q_1 \in A_1 \text{ or } q_2 \in A_2\} \)

**Theorem**

\[ L(M) = L(M_1) \cup L(M_2). \]
Product construction for union

\[ M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1) \text{ and } M_2 = (Q_1, \Sigma, \delta_2, s_2, A_2) \]

Create \( M = (Q, \Sigma, \delta, s, A) \) where

- \( Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\} \)
- \( s = (s_1, s_2) \)
- \( \delta : Q \times \Sigma \to Q \) where
  \[ \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \]
- \( A = \{(q_1, q_2) \mid q_1 \in A_1 \text{ or } q_2 \in A_2\} \)

**Theorem**

\[ L(M) = L(M_1) \cup L(M_2). \]
Set Difference

**Theorem**

\( M_1, M_2 \) DFA\( s. \) There is a DFA \( M \) such that \( L(M) = L(M_1) \setminus L(M_2). \)

**Exercise:** Prove the above using two methods.

- Using a direct product construction
- Using closure under complement and intersection and union
Question: Why are DFAs required to only move right? Can we allow DFA to scan back and forth? Caveat: Tape is read-only so only memory is in machine’s state.

- Can define a formal notion of a “2-way” DFA
- Can show that any language recognized by a 2-way DFA can be recognized by a regular (1-way) DFA
- Proof is tricky simulation via NFAs
**Question:** Why are DFAs required to only move right?
Can we allow DFA to scan back and forth? **Caveat:** Tape is read-only so only memory is in machine’s state.

- Can define a formal notion of a “2-way” DFA
- Can show that any language recognized by a 2-way DFA can be recognized by a regular (1-way) DFA
- Proof is tricky simulation via NFAs
THE END

... 

(for now)