CHEATSHEET - MATH MODELING - MIDSEM241

Chapter 1: Program verification

Hoare triples:

+) $(|\phi|)\, P\, (|\psi|)$ is considered a Hoare triple, with ϕ, ψ, P are pre-condition, post-condition, program, respectively.

+) Programs in Hoare triples are written in core languages:

$$E ::= n \mid x \mid -E \mid E+E \mid E-E \mid E*E$$
$$B ::= \texttt{true} \mid \texttt{false} \mid !B \mid \texttt{B \& B} \mid \texttt{B || B} \mid B<B \mid E<E$$
$$C ::= x = E \mid C;C \mid \texttt{if B \{C\} else \{C\}} \mid \texttt{while B \{C\}}$$

Partial correctness:

+) $\models_{\text{par}} (|\phi|)\, P\, (|\psi|)$ if for all states which satisfy ϕ, the state resulting from P's execution satisfies, provided that P terminates.

+) Rules:

1. Composition: $\dfrac{(|\phi|)\, C_1\, (|\eta|) \quad (|\eta|)\, C_2\, (|\psi|)}{(|\phi|)\, C_1; C_2\, (|\psi|)}$
2. Assignment: $(|[x \to E]\,\psi|)\; x = E\, (|\psi|)$.
   Weakest precondition: From the post-condition, push into the program.
3. Implication: $\dfrac{\vdash_{\text{AR}} \phi' \to \phi \qquad (|\phi|)\, C\, (|\psi|) \qquad \vdash_{\text{AR}} \psi \to \psi'}{(|\phi'|)\, C\, (|\psi'|)}$
4. If-statement: $\dfrac{(|\phi \wedge B|)\, C_1\, (|\psi|) \qquad (|\phi \wedge \neg B|)\, C_2\, (|\psi|)}{(\phi)\ \texttt{if B \{}C_1\texttt{\} else \{}C_2\texttt{\}}\ (|\psi|)}$
   Weakest precondition:
   $$\phi = (B \to \phi_1) \wedge (\neg B \to \phi_2) = (B \wedge \phi_1) \vee (\neg B \wedge \phi_2)$$
5. Partial-while: $\dfrac{(|\psi \wedge B|)\, C\, (|\psi|)}{(|\psi|)\ \texttt{while}\ B\ \texttt{\{}C\texttt{\}}\ (|\psi \wedge \neg B|)}$
   Weakest precondition: (use the bottom-up method)
   (a) Bottom: $\phi \to \eta$ (with η is an invariant)
   (b) Inside the loop: $(\eta \wedge \neg B) \to \psi$
   (c) Go outside the loop: $(|\eta|)\ \texttt{while (B) \{C\}}\ (|\eta \wedge \neg B|)$

Total correctness:

+) $\models_{\text{tot}} (|\phi|)\, P\, (|\psi|)$ if, for all states which satisfy ϕ, P is guaranteed to terminate and the resulting state satisfies ψ.

+) Total-while: $\dfrac{(|\psi \wedge B \wedge 0 \le E = E_0|)\, C\, (|\psi \wedge 0 \le E < E_0|)}{(|\psi \wedge 0 \le E|)\ \texttt{while B \{C\}}\ (\psi \wedge \neg B)}$ +) *Weakest precondition:* (use the bottom-up method)

1. Bottom:
   $$\phi \to (\eta \wedge 0 \le E)$$
   (with E is a decreasing variant, and η is an invariant)
2. Inside the loop:
   $$(\eta \wedge \neg B) \to \psi$$
3. Go outside the loop:
   $$\models_{\text{tot}} (|\eta \wedge 0 \le E|)\ \texttt{while (B) C}\ (|\eta \wedge \neg B|)$$

Finding loop invariants: can be found here.

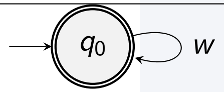
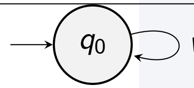
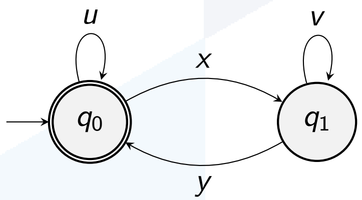
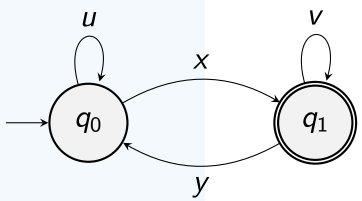
Chapter 2: Automata

Operations in formal languages:

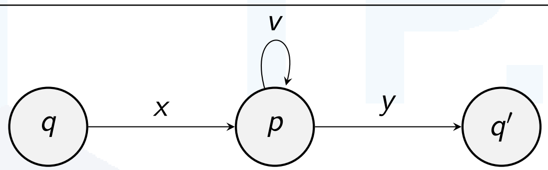
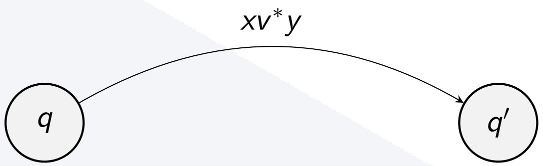
Let L, L_1, L_2 be formal languages in Σ.

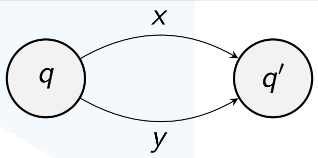
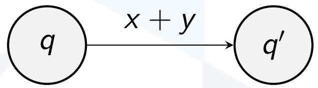
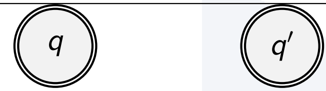
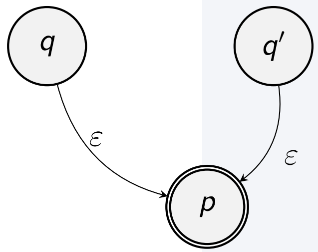
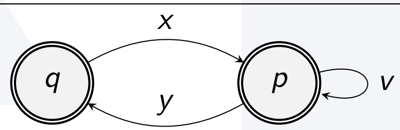
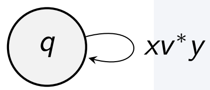
1. Union: $L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ or } u \in L_2\}$
2. Intersection: $L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ and } u \in L_2\}$
3. Difference: $L_1 \setminus L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ and } u \notin L_2\}$
4. Complement: $\overline{L} = \Sigma^* \setminus L$
5. Multiplication: $L_1 L_2 = \{uv \mid u \in L_1,\ v \in L_2\}$
6. Power: $\begin{cases} L^0 = \{\varepsilon\} & \text{with } |\varepsilon| = 0 \\ L^n = L^{n-1}L, & \forall n \ge 1 \end{cases}$
7. Star operation: $L^* = \bigcup_{i=0}^{\infty} L^i$
8. $L^+ = \bigcup_{i=1}^{\infty} L^i$

Simple automata:

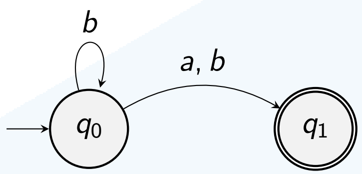
Automata	Regular expression
	w^*
	0
	$(u + xv^*y)^*$
	$u^*x\,(v + yu^*x)^*$

Special cases:

Cases	Result automata
Remove the state (p)	Before:  After: 

Join arrows	Before:  After: 
Create a single final state	Before:  After: 
Beware of loops	Before:  After: 

From finite automata to regular expression:
Arden's theorem: $R = Q + RP \Rightarrow R = QP^*$ (with P does not contain ε)
For example:

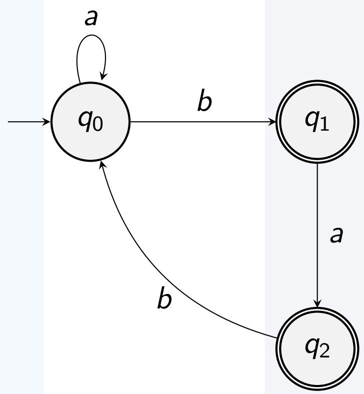


We have $q_0 = q_0 b = q_0 b + \varepsilon \Rightarrow q_0 = \varepsilon b^* = b^*$; $q_1 = q_0(a+b) = b^*(a+b)$.

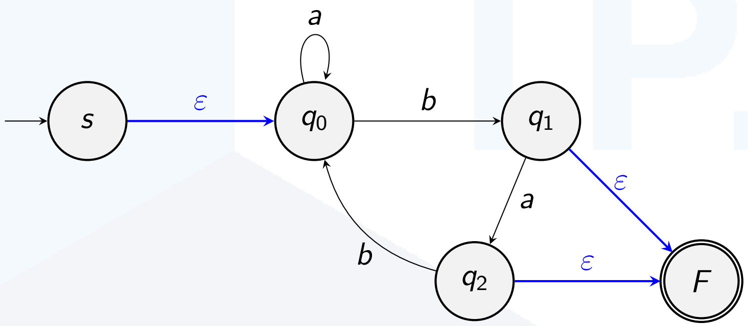
Elimination method:

1. Convert the automata to satisfy the following conditions:
   - The initial state must not have incoming edge.
   - The final state must not have outcoming edge.
   - Only one final state.
2. Eliminate intermediate states one by one.

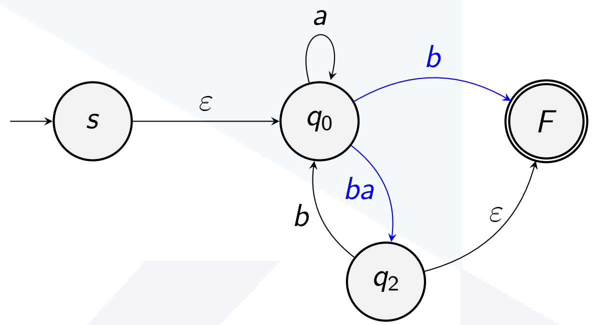
For example:



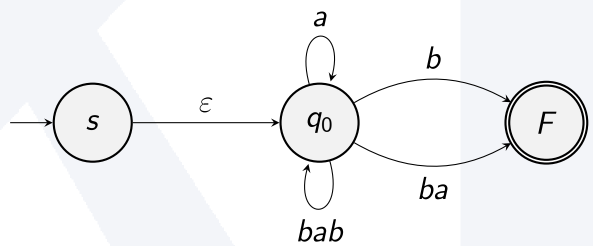
Converting the above automata, we have:



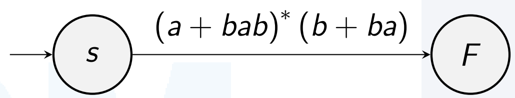
Eliminating state q_1, we have:



Eliminating state q_2, we have:



Eliminating state q_0, we have:



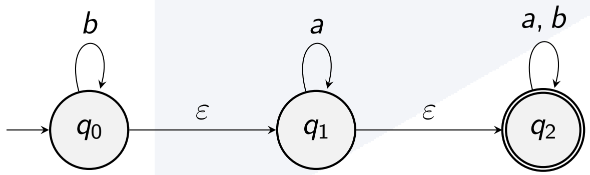
Set of problems: ε-NFA $\rightarrow$ NFA $\rightarrow$ DFA $\rightarrow$ min-DFA:

ε-NFA convert to NFA:

1. Construct the table of states, then mark the initial and final states.
2. Construct the ε-closure (which is, set of states that can be reachable from q_i through ε-transition).
3. Construct the final NFA through the formula:

$$\boxed{\delta'(q_0, a) = \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}\left(q_0\right), a\right)\right)}$$

For example:



Converting the above automata into the table of states, we have:

δ	a	b
q_0		q_0
q_1	q_1	
q_2	q_2	q_2

Construct the ε-closure table:

q_i	ε-closure
q_0	$q_0,\ q_1,\ q_2$
q_1	$q_1,\ q_2$
q_2	q_2

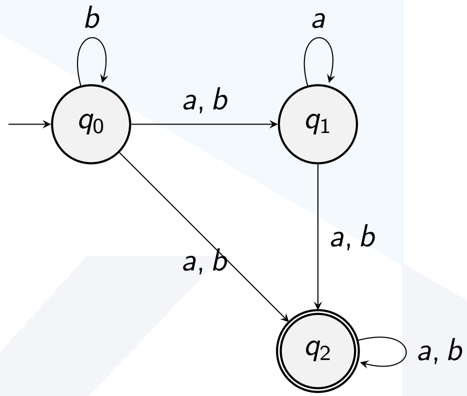
Construct the final NFA:

$$\begin{aligned}\delta'(q_0, a) &= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}\left(q_0\right), a\right)\right)\\ &= \varepsilon\text{-closure}\left(\delta\left(\left\{q_0, q_1, q_2\right\}, a\right)\right)\\ &= \varepsilon\text{-closure}\left(\delta\left(q_0, a\right) \cup \delta\left(q_1, a\right) \cup \delta\left(q_2, a\right)\right)\\ &= \varepsilon\text{-closure}\left(\varnothing \cup \left\{q_1\right\} \cup \left\{q_2\right\}\right) = q_1, q_2\end{aligned}$$

Then do the rest for all other q_i and information, we have the following table.

δ'	a	b
q_0	$q_1,\ q_2$	q_0, q_1, q_2
q_1	$q_1,\ q_2$	q_2
q_2	q_2	q_2

Construct the final NFA:



NFA → DFA:

1. Construct the table of state.
2. Construct the table of state for new automata by using the following formula, then create the new transitions for new states that do not exist in NFA (if it has).

$$\boxed{\delta'\left(q_0 q_1, a\right) = \delta'\left(q_0, a\right) \cup \left(q_1, a\right)}$$

Note that, with Q_{NFA} is set of all the states in the given NFA, then

$$Q_{\text{DFA}} \subseteq P\left(Q_{\text{NFA}}\right)$$

ε-NFA → DFA:

1. Construct the table of state and the table of ε-closure.
2. Construct the table of state for new DFA by using the formula until no new node is created:

$$\boxed{\delta'(A, a) = \varepsilon\text{-closure}\left(\delta(A, a)\right)}$$

$$\boxed{\delta'\left(q_0 q_1, a\right) = \delta'\left(q_0, a\right) \cup \left(q_1, a\right)}$$

Final states of the DFA are nodes that contain the final state(s) of NFA.

DFA → min-DFA:

1. Construct the table of state.
2. Construct the partition table like below.

s	All of states
$\mathrm{cl}(s)$	
$\mathrm{cl}(s, \cdots)$	

In this step, we partition the states as final states and non-final states at the first non-header row, then use it to fill the next rows.

3. Partition the states until $\mathrm{cl}_n = \mathrm{cl}_{n-1}$. The number of different states is the number of states in min-DFA.