Closure Properties for DFA

Lecture 2

Binghui Peng

Recap from last lecture

Formal Definition of DFAs

Def: A **DFA** M is a 5-tuple $(Q, \Sigma, \delta, s, F)$ where:

1. Q is a finite set of **states**.
2. Σ is a finite **alphabet**.
3. $\delta : Q \times \Sigma \to Q$ is the **transition function**.
4. $s \in Q$ is the **start state**.
5. $F \subseteq Q$ is the set of **final states**.

Def: If $x = x_1 x_2 \cdots x_n \in \Sigma^*$, M **accepts** x if there is a sequence of states $q_0, q_1, \ldots, q_n$ such that:

- $q_0 = s$
- $q_i = \delta(q_{i-1}, x_i)$ for $1 \leq i \leq n$
- $q_n \in F$

Def: A language $L \subseteq \Sigma^*$ is **regular** if there exists a DFA M such that $L(M) = L$.

Computer Implementation of DFAs

DFAs are efficient to implement. The transition table can be represented as a 2D array.

- States: $\{0, 1, 2, 3, 4\}$
- Alphabet: $\{0, 1\}$
- Start state: 0
- Final states: $\{1, 3\}$

	0 (a)	1 (b)
0	1	4
1	0	2
2	4	3
3	4	2
4	4	4

- Time Complexity: $O(n)$ to process a string of length n.
- Space Complexity: $O(|Q| \cdot |\Sigma|)$ to store the transition table.

Given the description of DNA, a string can be decided in linear time!

This lecture: What kind of language is regular?

By the end of the lecture...

We should know how to construct a DFA for the set of numbers

$$S := \{x \in \mathbb{N} : x \equiv x_1 \mod p_1 \wedge x \equiv x_2 \mod p_2\}$$

Set Operations on Languages

Languages can be combined using standard set operations:

- **Union:** $L_1 \cup L_2$
- **Intersection:** $L_1 \cap L_2$
- **Concatenation:** $L_1 \cdot L_2 = \{xy : x \in L_1, y \in L_2\}$
- **Kleene Star:** L^*

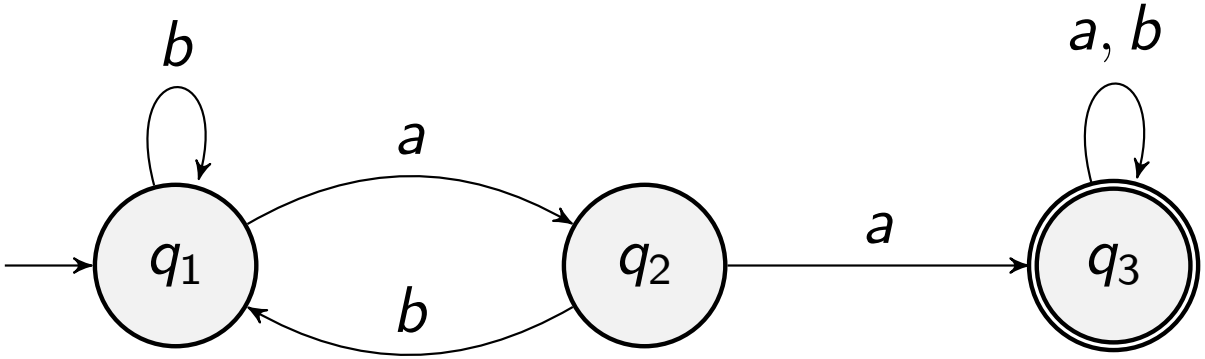
String Operations and Notation

Let $x, y \in \Sigma^*$:

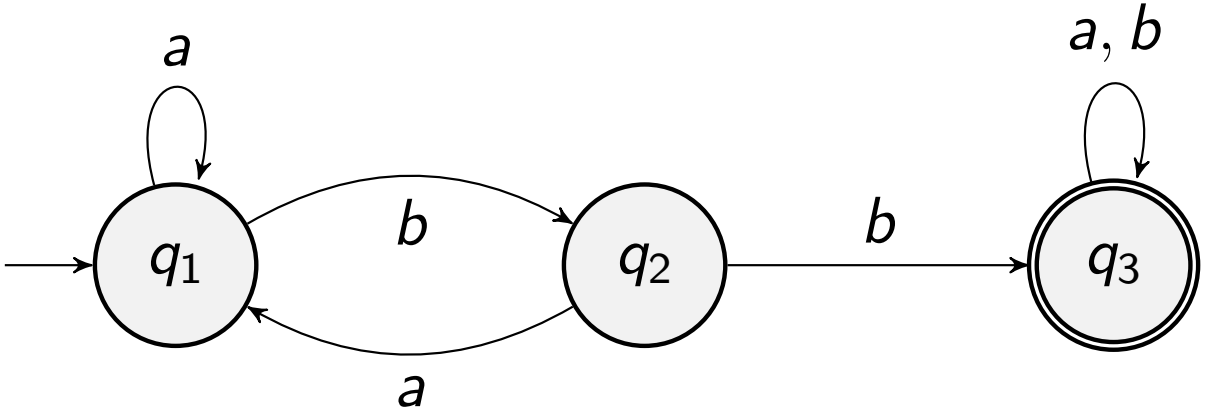
- **Concatenation:** xy (or $x \cdot y$).
- Note: $x \cdot e = e \cdot x = x$.
- **Length:** $|x|$ is the number of symbols in x.
- **Count:** $\#_a(x)$ is the number of occurrences of symbol a in x.
- **Prefix:** $x \preceq y$ if there exists z such that $xz = y$.

Example Languages

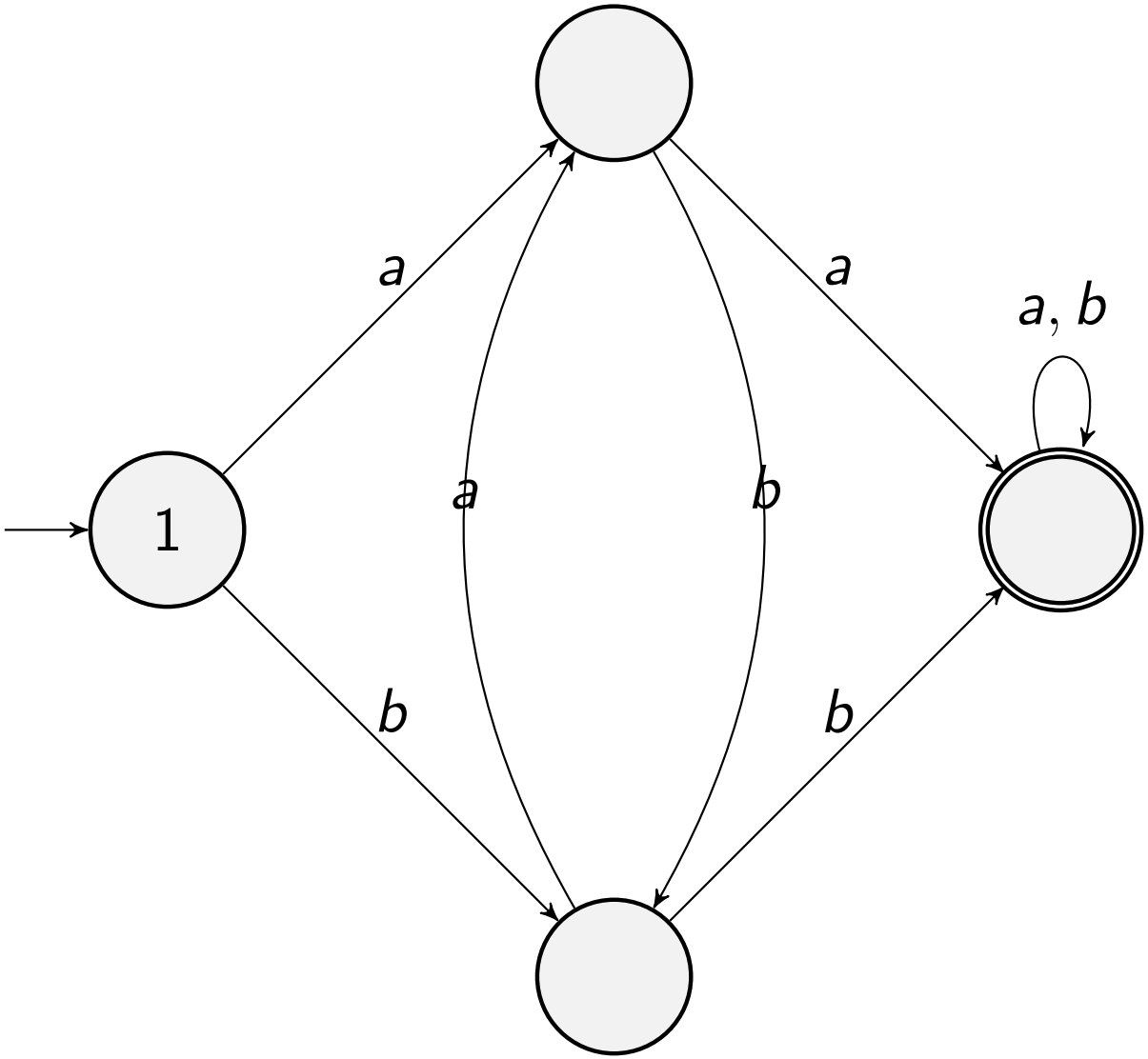
The language L_a is the set of words over $\{a, b\}$ with two consecutive a's.



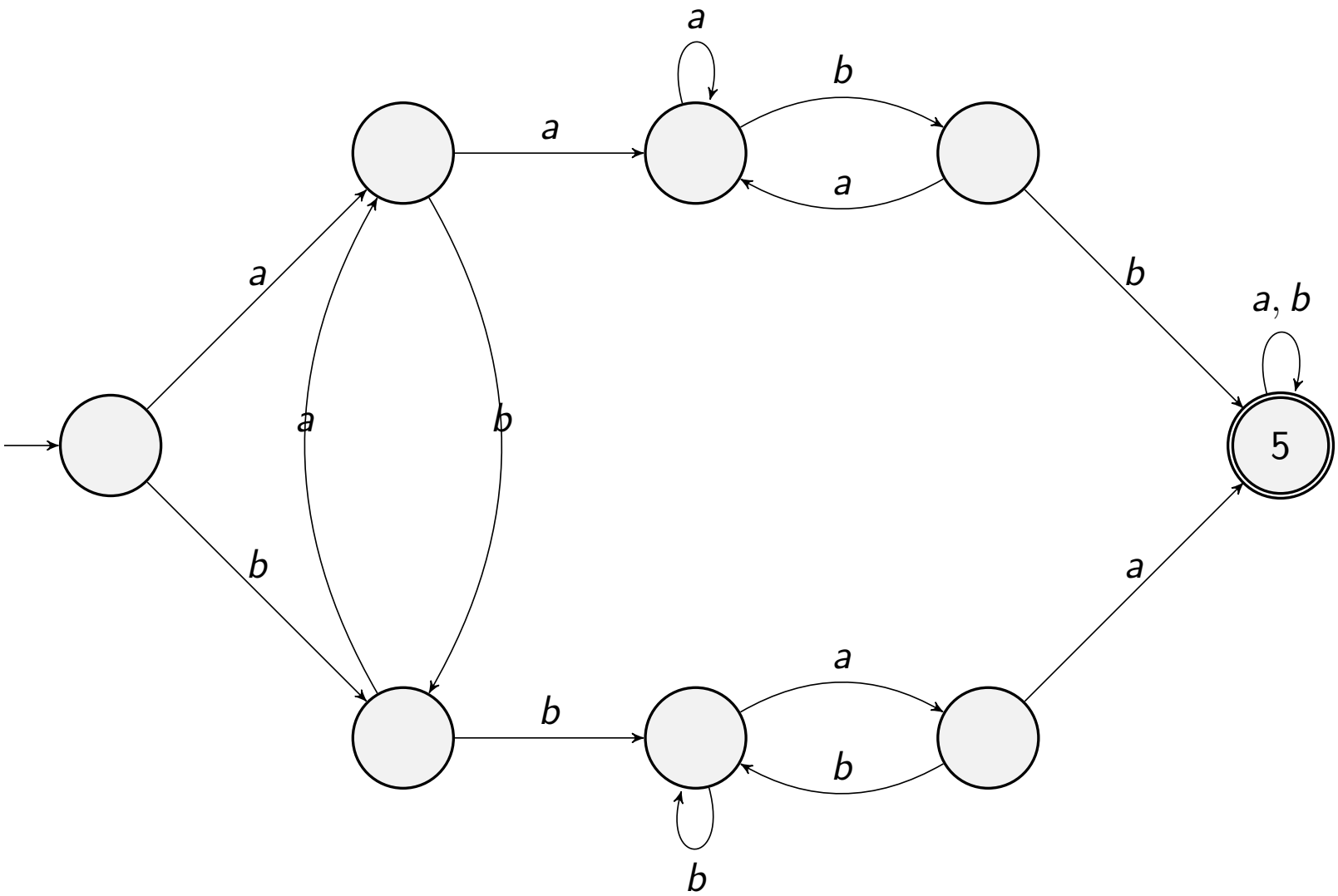
The language L_b is the set of words over $\{a, b\}$ with two consecutive b's.



Union: $L_a \cup L_b$



Intersection: $L_a \cap L_b$



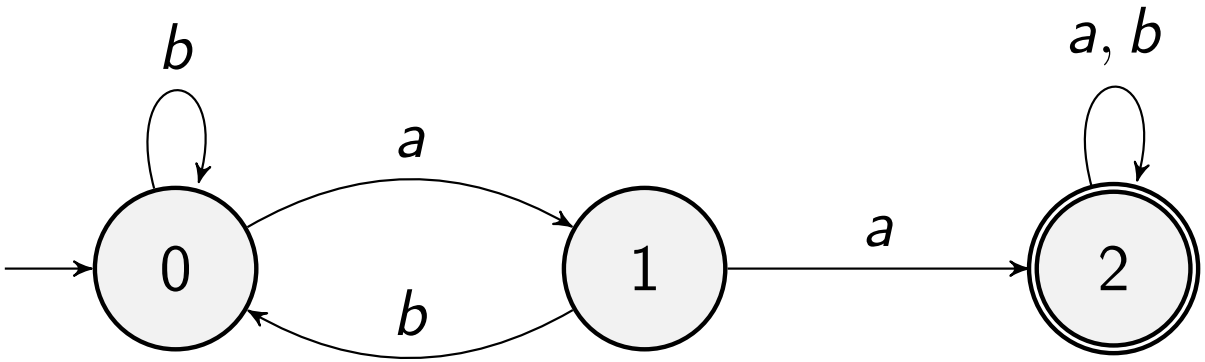
Product Construction Idea

To recognize the union or intersection of two languages, we can run both machines in parallel.

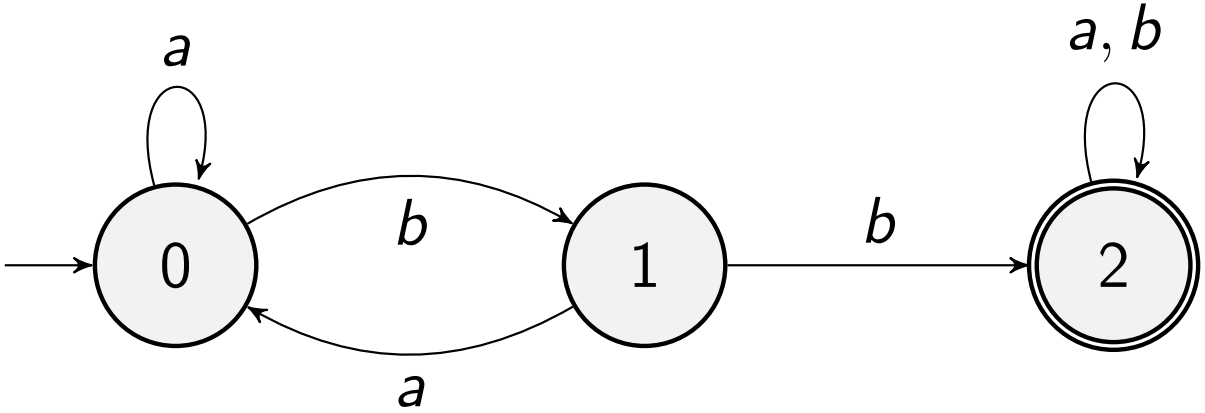
Key Concept: Each state in the new DFA will represent a pair of states (q_1, q_2), where q_1 is the current state of the first machine and q_2 is the current state of the second machine.

Component Machines: L_a and L_b

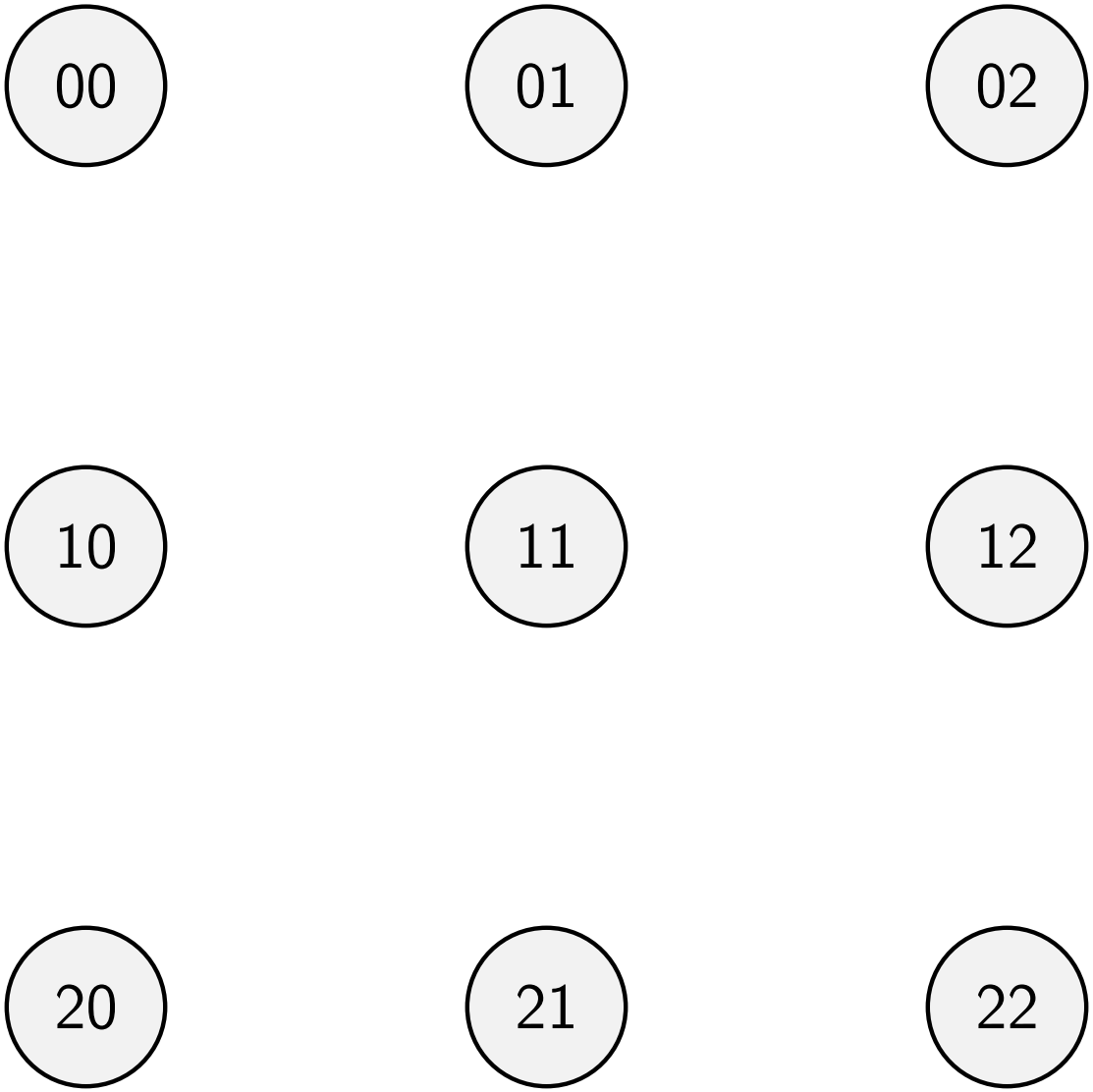
DFA for L_a (consecutive a's):



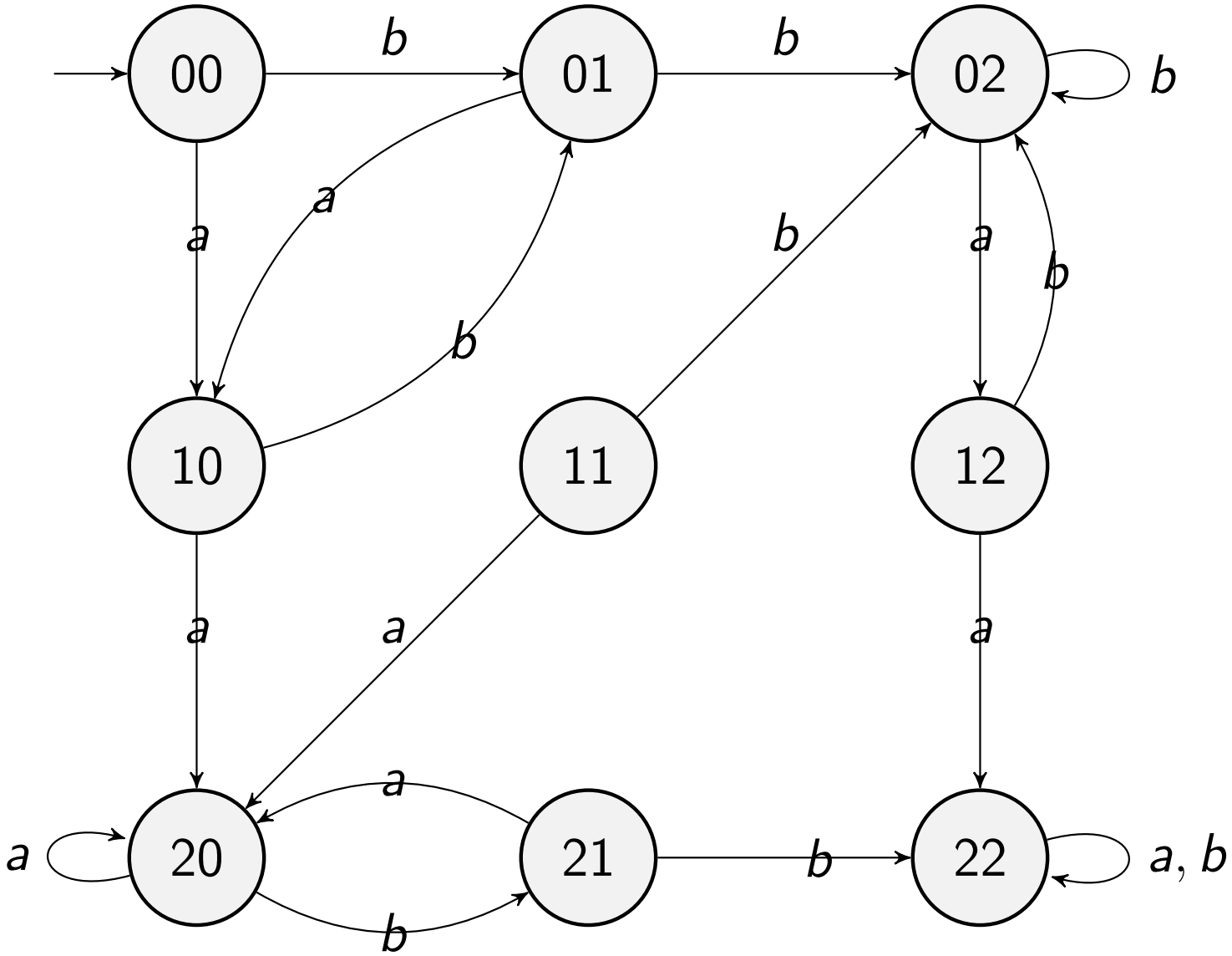
DFA for L_b (consecutive b's):



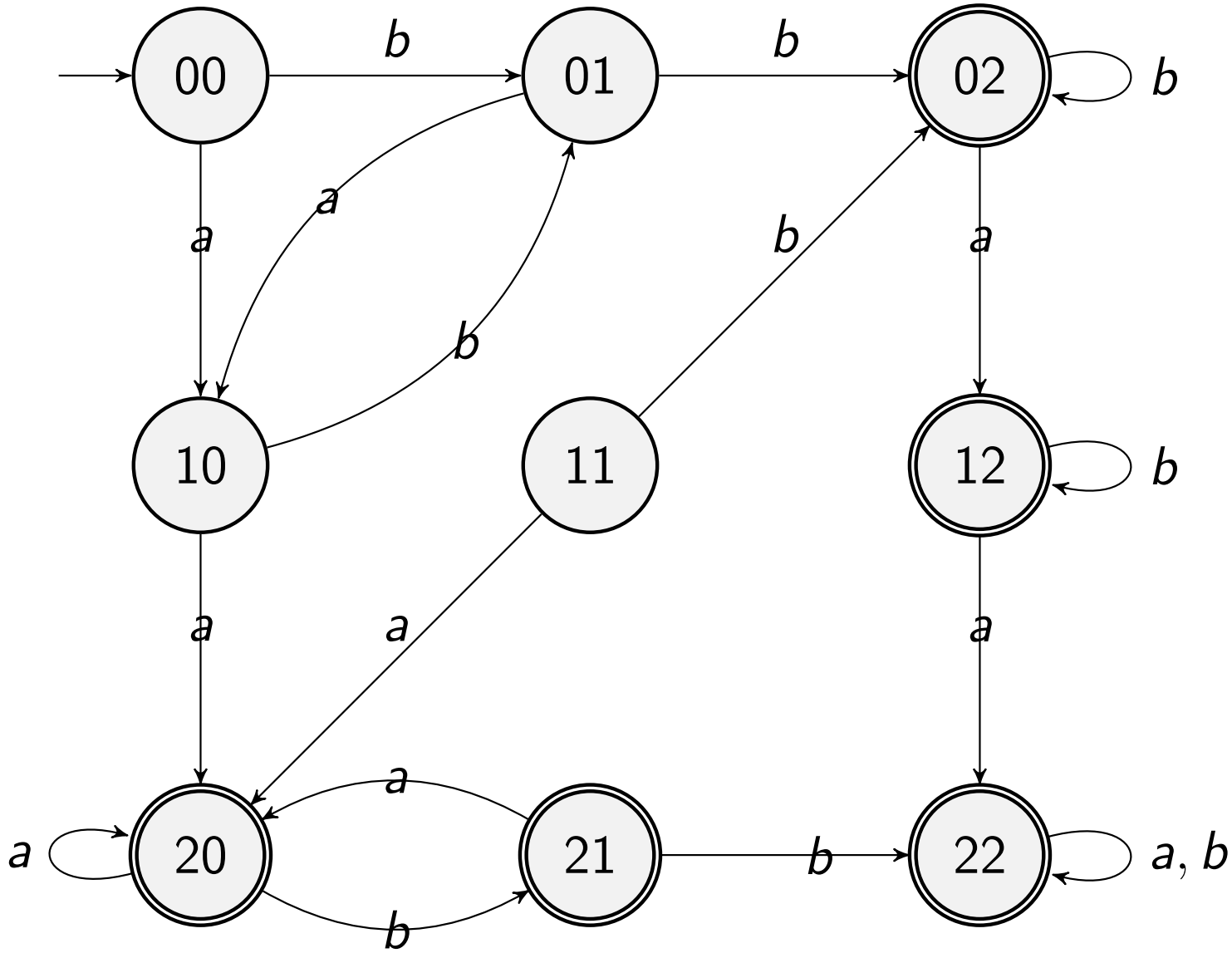
Product DFA: State Grid



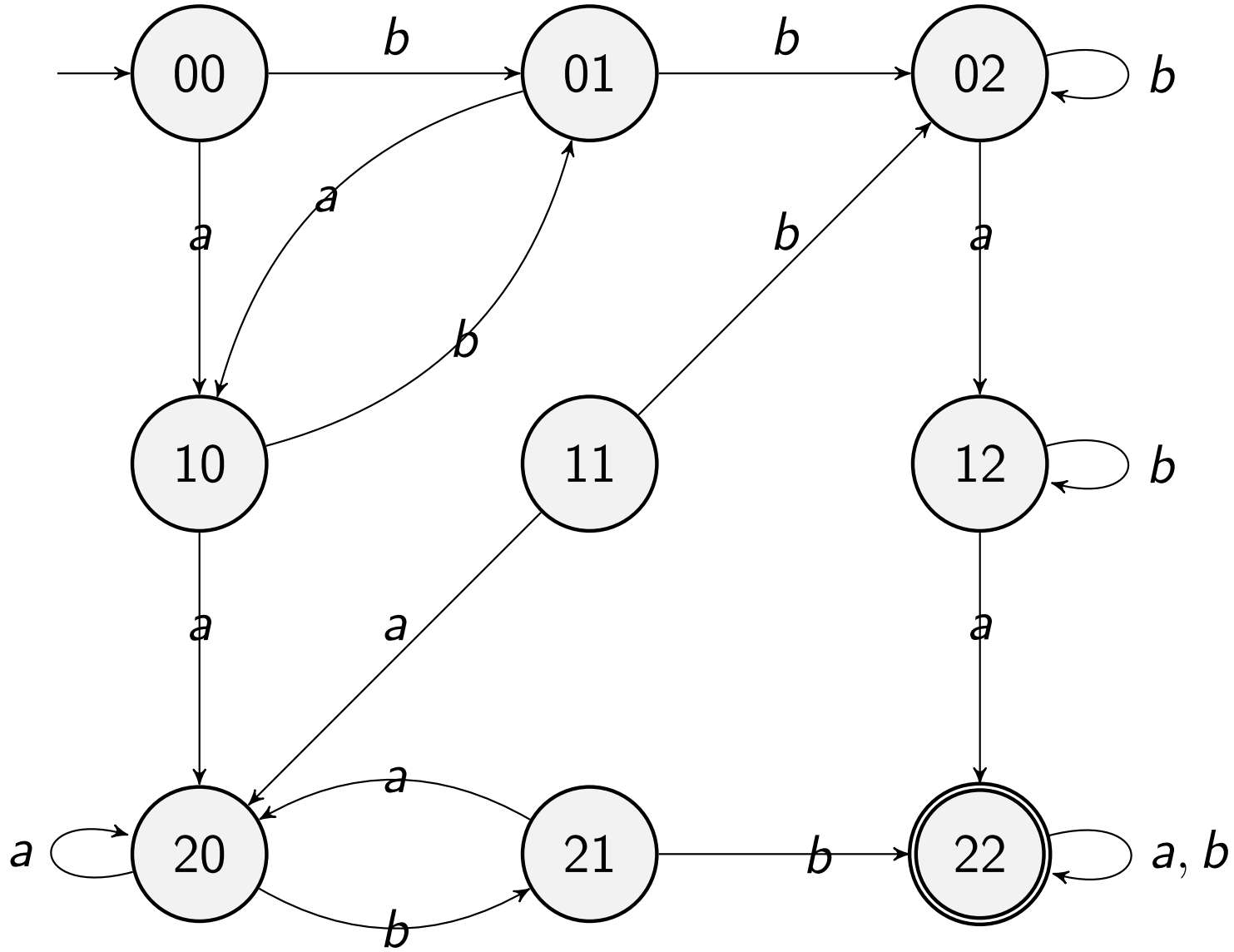
Product DFA: Transitions



Product DFA: Union $L_a \cup L_b$



Product DFA: Intersection $L_a \cap L_b$



Closure Under Union

Theorem: If L_1 and L_2 are regular languages, then $L_1 \cup L_2$ is regular.

Formal Construction: Let L_1 be recognized by $(Q_1, \Sigma, \delta_1, s_1, F_1)$ and L_2 by $(Q_2, \Sigma, \delta_2, s_2, F_2)$. Then $L_1 \cup L_2$ is recognized by:

$$(Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F)$$

where:

$$\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$$

and

$$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$$

Note: If L_i has n_i states, the product DFA has $n_1 n_2$ states.

Closure Under Intersection

Theorem: If L_1 and L_2 are regular languages, then $L_1 \cap L_2$ is regular.

Formal Construction: Let L_1 be recognized by $(Q_1, \Sigma, \delta_1, s_1, F_1)$ and L_2 by $(Q_2, \Sigma, \delta_2, s_2, F_2)$. Then $L_1 \cap L_2$ is recognized by:

$$(Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F)$$

where:

$$\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$$

and

$$F = F_1 \times F_2$$

Note: The product DFA has $n_1 n_2$ states.

Complement of a Language

Definition: The complement of a language $L \subseteq \Sigma^*$ is $\overline{L} = \Sigma^* - L$.

Theorem: If L is regular, then $\overline{L}$ is regular.

Formal Construction: If L is recognized by $(Q, \Sigma, \delta, s, F)$, then $\overline{L}$ is recognized by:

$$(Q, \Sigma, \delta, s, Q - F)$$

Note: We simply swap final and non-final states. The number of states remains n.

Closure Under Concatenation

Question: If L_1 and L_2 are regular, is $L_1 \cdot L_2$ regular?

Closure Under Concatenation

Question: If L_1 and L_2 are regular, is $L_1 \cdot L_2$ regular?

Answer: YES.

- While it is possible to prove this using DFAs, the construction is quite complex.
- We will provide a much simpler proof after introducing **Non-deterministic Finite Automata (NFA)**.

Closure Under Kleene Star

Question: If L is regular, is L^* regular?

Closure Under Kleene Star

Question: If L is regular, is L^* regular?

Answer: YES.

- Similar to concatenation, the DFA-based proof is difficult.
- A simpler proof will be presented using NFAs.

Summary of Closure Properties

The following table summarizes the closure properties for regular languages and the number of states in the resulting machine (L_i has n_i states).

Operation	**Resulting States (DFA)**
$L_1 \cup L_2$	$n_1 n_2$
$L_1 \cap L_2$	$n_1 n_2$
$\overline{L}$	n
$L_1 \cdot L_2$	(Proven later via NFA)
L^*	(Proven later via NFA)