Theory of Automata

Lecture #17-18-19

Transition Graph (T.G)

- Like an NFA except for start states and transitions
- Transitions
  - Go from ***some*** states to some other states.
  - Labeled by ***strings*** of letters from the alphabet.
  - The labels may include the empty word Λ.
- Start states
  - Maybe ***more*** than one.

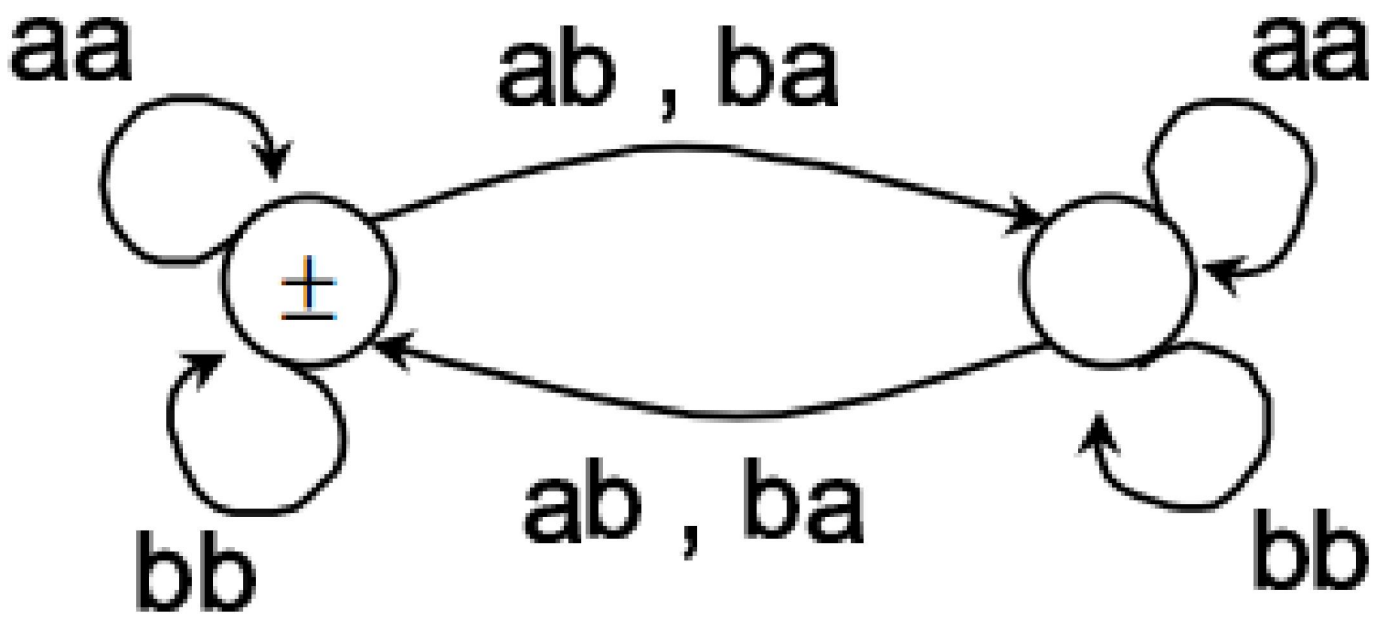
Transition Graph (T.G)

- A finite set of states **at least one** of which is designated as the **start state** (-) and **some** (maybe none) of which are **designated as final states** (+).
- An alphabet Σ of possible input letters from which input strings are formed.
- A **finite set of transitions** that show how to go from one state to another based on reading specified substrings of input letters (possibly even the null string λ).

Transition Graph (T.G)

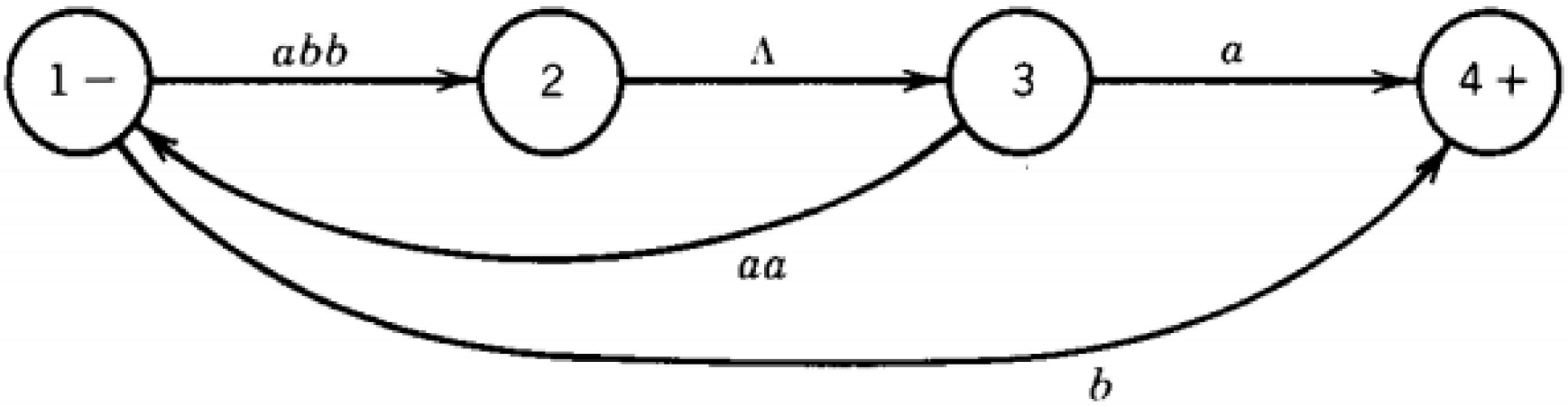
- A **successful path** through a transition graph is a series of edges forming a path beginning at some start state **(there may be several)** and ending at a final state. If we concatenate in order the strings of letters that label each edge in the path, we produce a word that is accepted by this machine.

Transition Graph (T.G)-Even Even



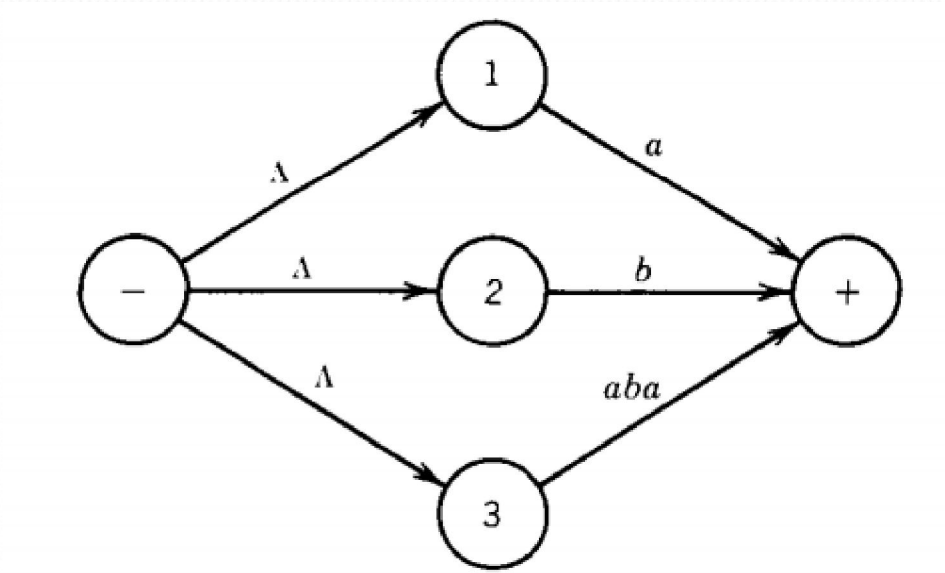
Transition Graph (T.G)

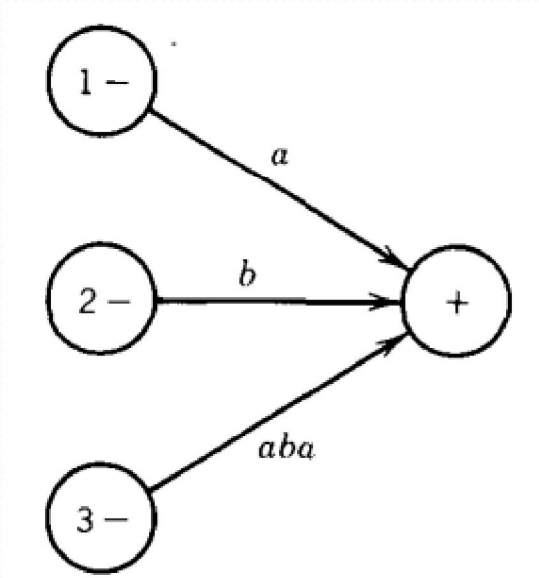
- For example, consider the following TG



- Some accepted words are **abba**, **abbaaabba**, and **b etc**

Transition Graph (T.G)





Both are Same

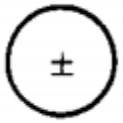
Transition Graph (T.G)

- It is extremely important for us to understand that every FA is also a TG.
- This means that any picture that represents an FA can be interpreted as a picture of a TG. Of course, not every TG satisfies the definition of an FA.
- Let us consider some more examples of TG's.

Transition Graph (T.G)

- The picture above represents a TG that accepts nothing, not even the null string Λ. To be able to accept anything, it must have a final state.

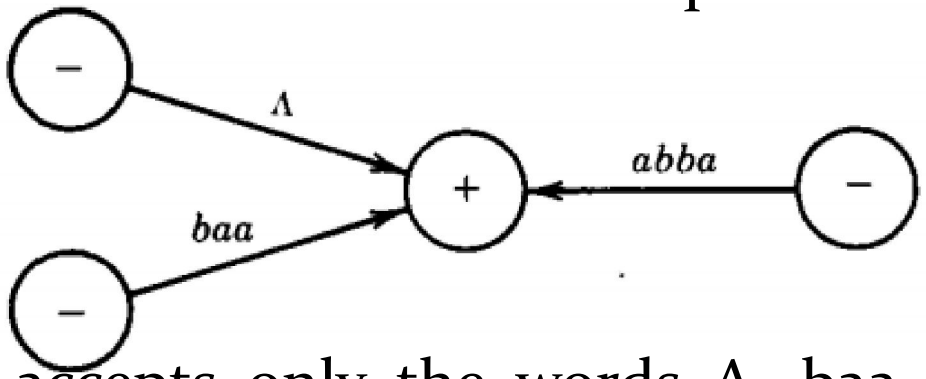
- The machine



accepts only the string λ

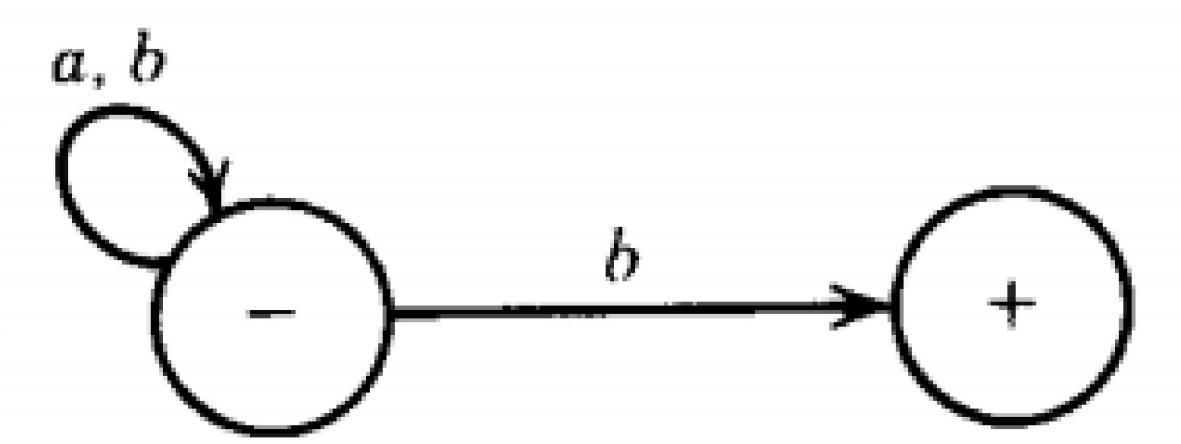
Transition Graph (T.G)

- Any TG in which some start state is also a final state will always accept the string λ; this is also true of FA's
- There are some other TG's that accept the word λ, for example:



- This machine accepts only the words A, baa, and abba. Anything read while in the + state will cause a crash, since the + state has no outgoing edges

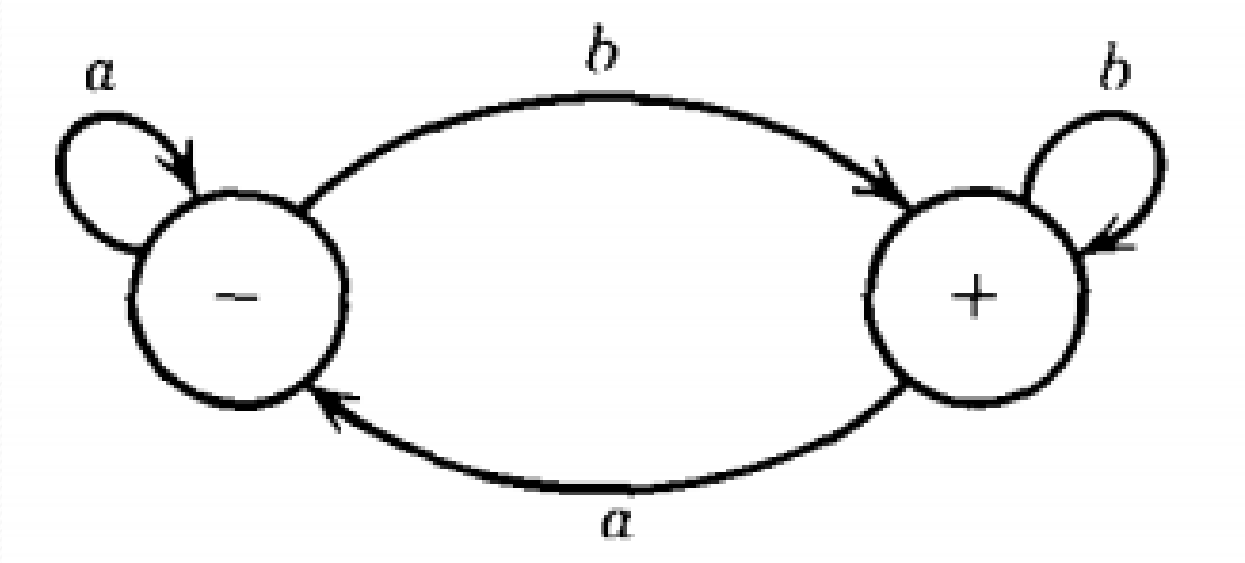
Transition Graph (T.G)



- We can read all the input letters one at a time and stay in the left-side state. When we read a b in the - state there are two possible edges we can follow. If the very last letter is a b, we can use it to go to the + state

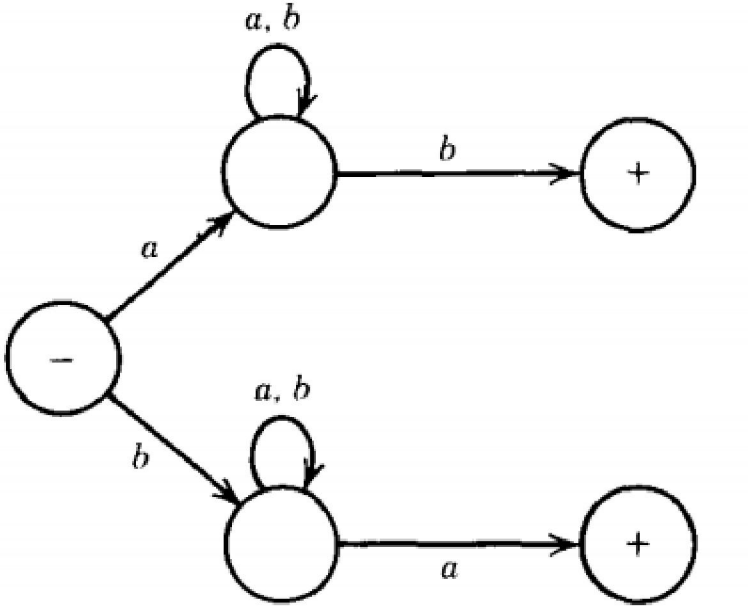
Transition Graph (T.G)

- The language accepted by this TG is all words ending in b. One regular expression for this language is (a + b)*b and an FA that accepts the same language is



Transition Graph (T.G)

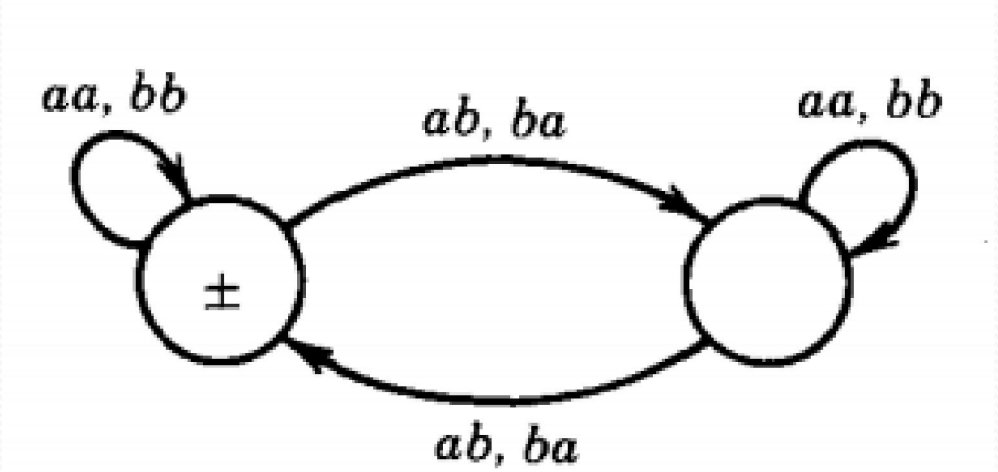
- A TG



- accepts the language of all words that begin and end with different letters

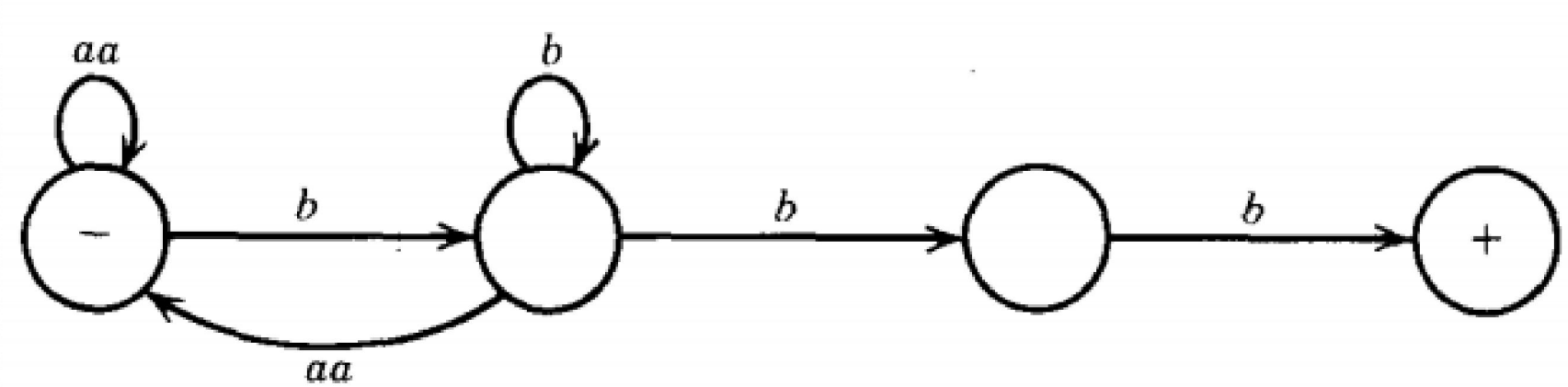
Transition Graph (T.G)

- Consider the following TG:



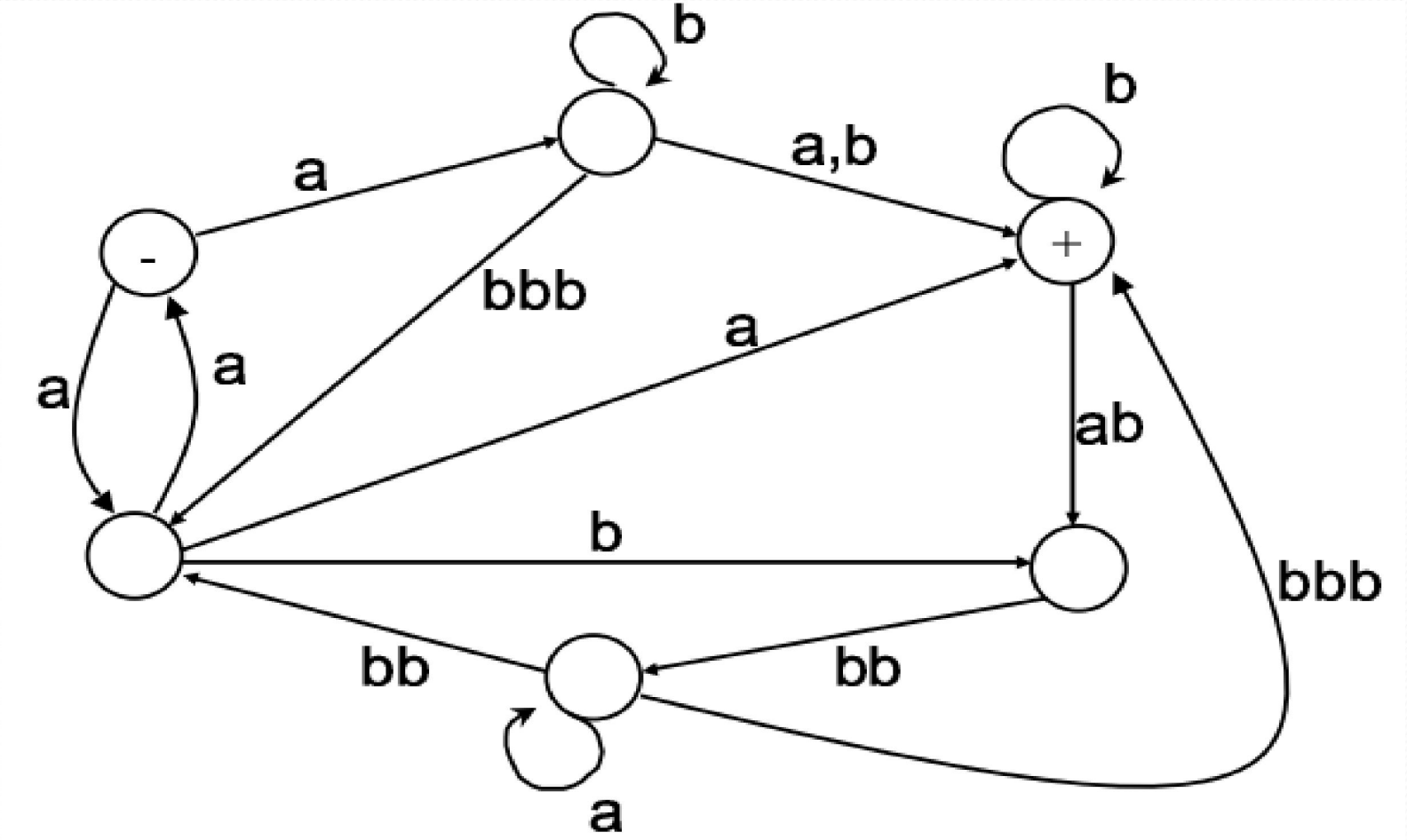
- in this TG every edge is labeled with a pair of letters. This means that for the string to be accepted it must have an **even number of letters** that are read in and processed in groups of two's.

Transition Graph (T.G)



- accepts the language of all words in which the **a's occur only in even clumps** and that end in **three or more b's**.

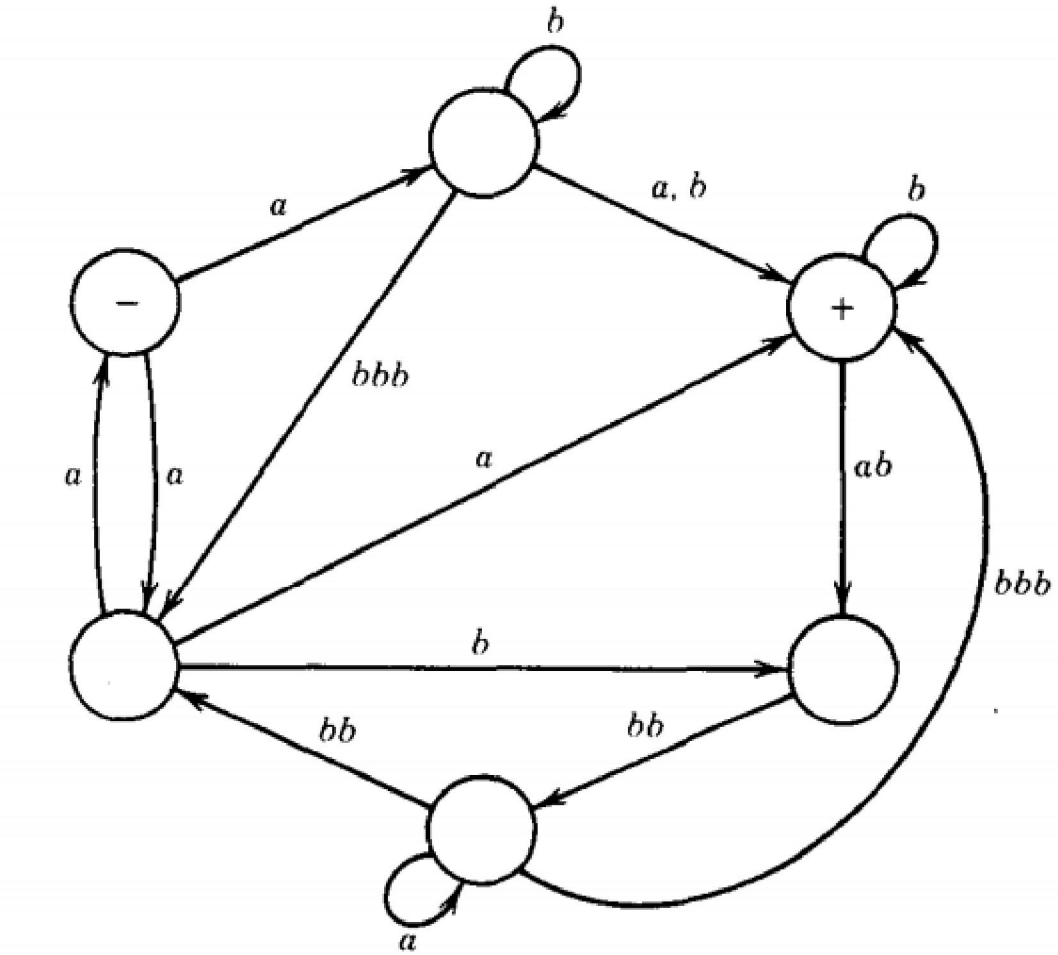
Transition Graph (T.G)-Even Even



is **abbbabbbabba** is accepted?

Transition Graph (T.G)

- Consider this TG

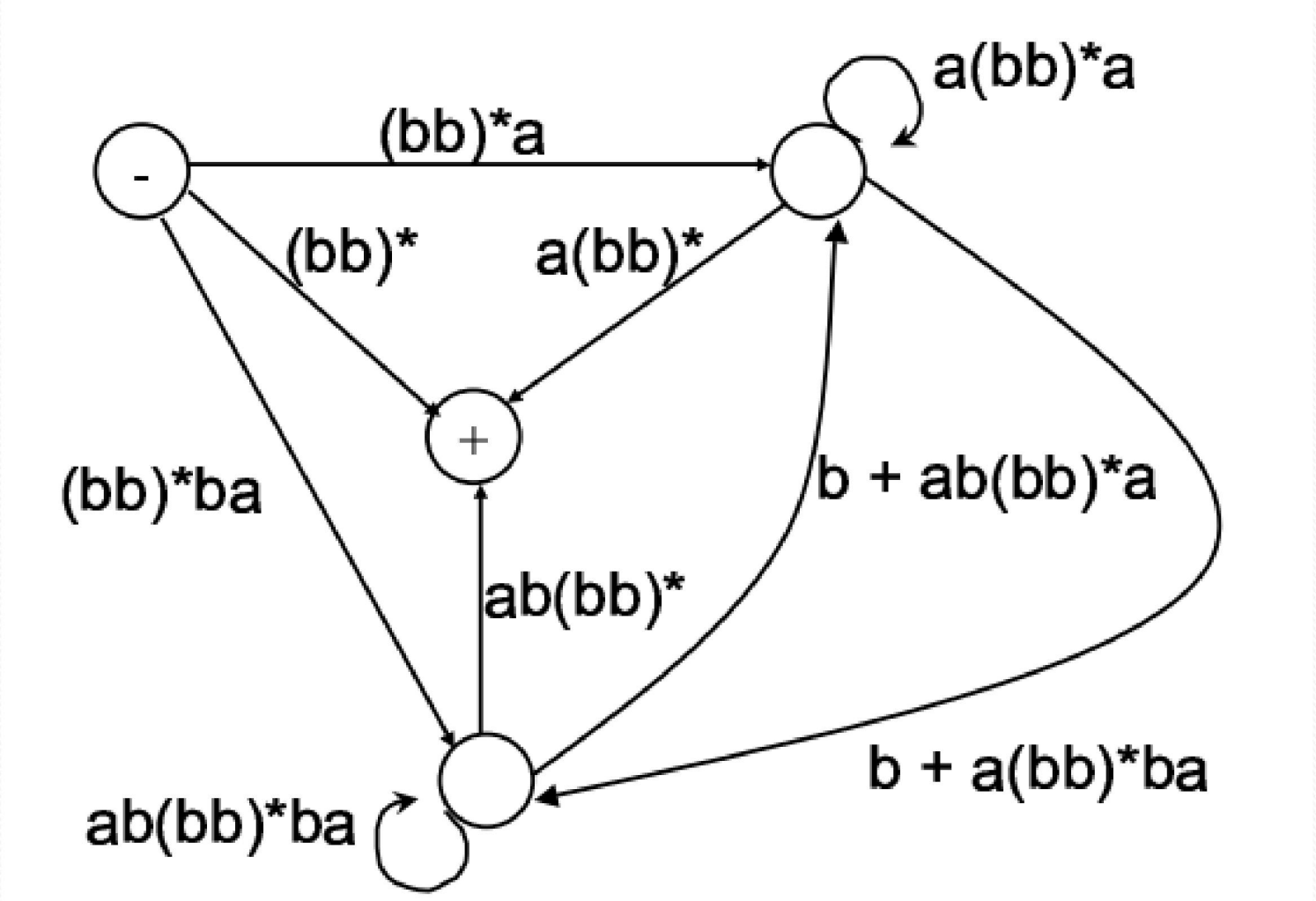


Is the word *abbbabbbabba* accepted by this machine? (Yes, in two ways.)

Generalized Transition Graph(GTG)

- Its is similar to transition graph except for transitions
- Transitions
  - Goto from some states to some other states
  - Labeled by **regular expressions** formed from letters from the alphabets

Generalized Transition Graph(GTG)



Types of Automata

- **FA**
  - For each state and letter there is a **unique transition.**
- **NFA**
  - For **some** states and letters there is a transition.
- **NFA-Λ**
  - For **some** states and letters and **Λ** is a transition.

Types of Transition Graphs

- **TG**
  - For some states and **strings** there is a transition.
  - Maybe **more than one start state.**
- **GTG**
  - For some states and **regular expressions** there is a transition.
  - Maybe **more than one start state.**

Assignment

- Solve question # 4 of book on page 103

Kleene Theorem-PART-I

- Part I: Every FA is a TG

FA	TG
Single Start State and multiple end states	Multiple Start States and multiple end states
Finite set of input symbols	Same
Finite set of transitions	Same
Deterministic	Non-Deterministic
Distinguishing Rule	No such rule

Kleene Theorem-PART-II

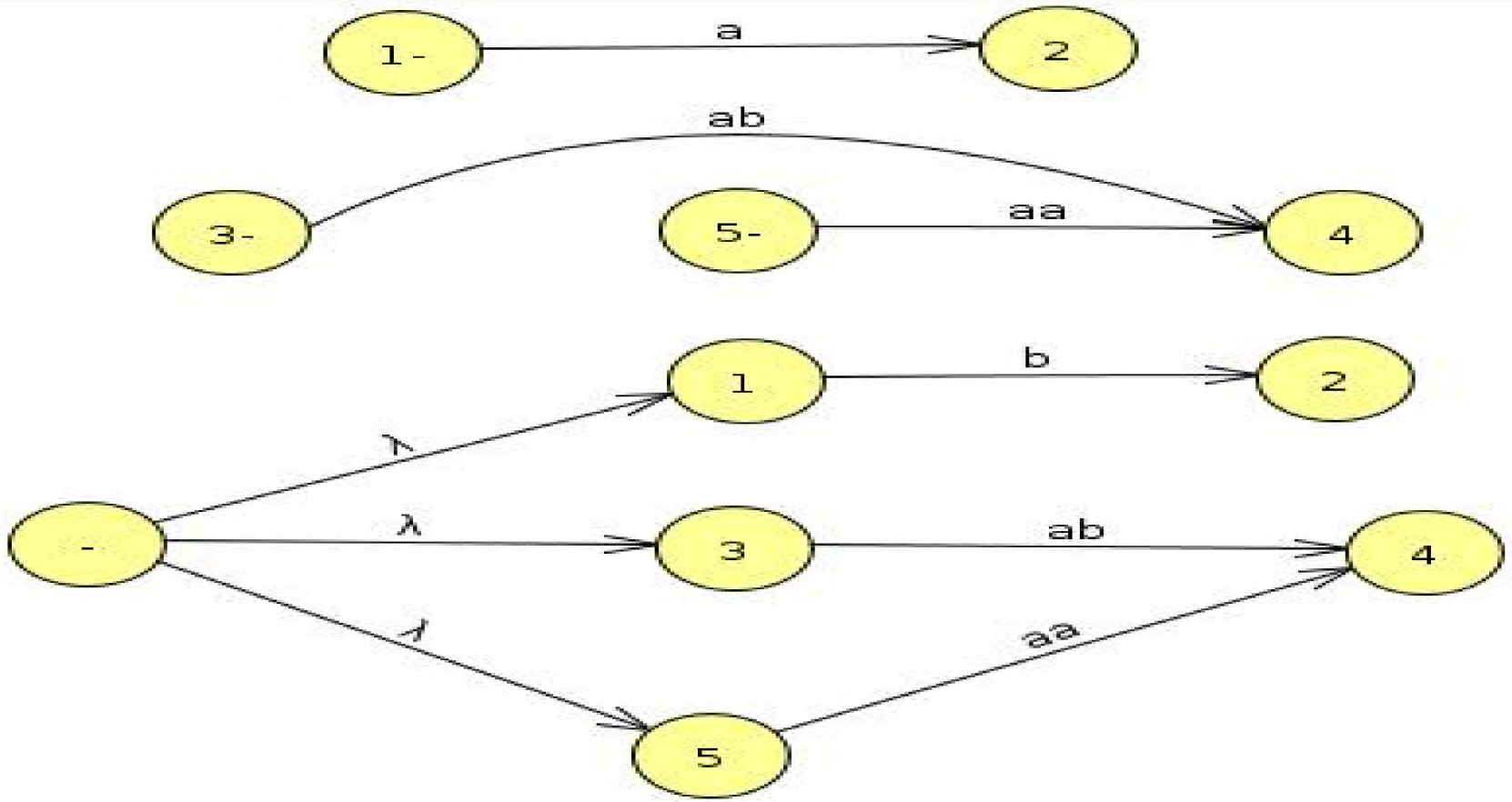
- Part I: Every language that can be defined by a finite automaton can also be defined by a transition graph. (Every FA is a TG)
- This is the easiest part.
- Every finite automaton is itself a transition graph.
- Therefore, any language that has been defined by a finite automaton has already been defined by a transition graph.
- Done

Kleene Theorem-PART-II

- Part II : Every language that can be defined by a transition graph can also be defined by a regular expression
- The prove of this Part requires a systematic algorithm through which a **TG can be converted to a GTG, in which all transitions are actually regular expressions**. **Thus, we need to transform a TG to a GTG and eliminate its various states and convert it to a single state or single transition GTG**, so that we can get the regular expression associated with the TG.
- Steps involved in TG to GTG conversion Examples taken from Daniel I Cohen Book

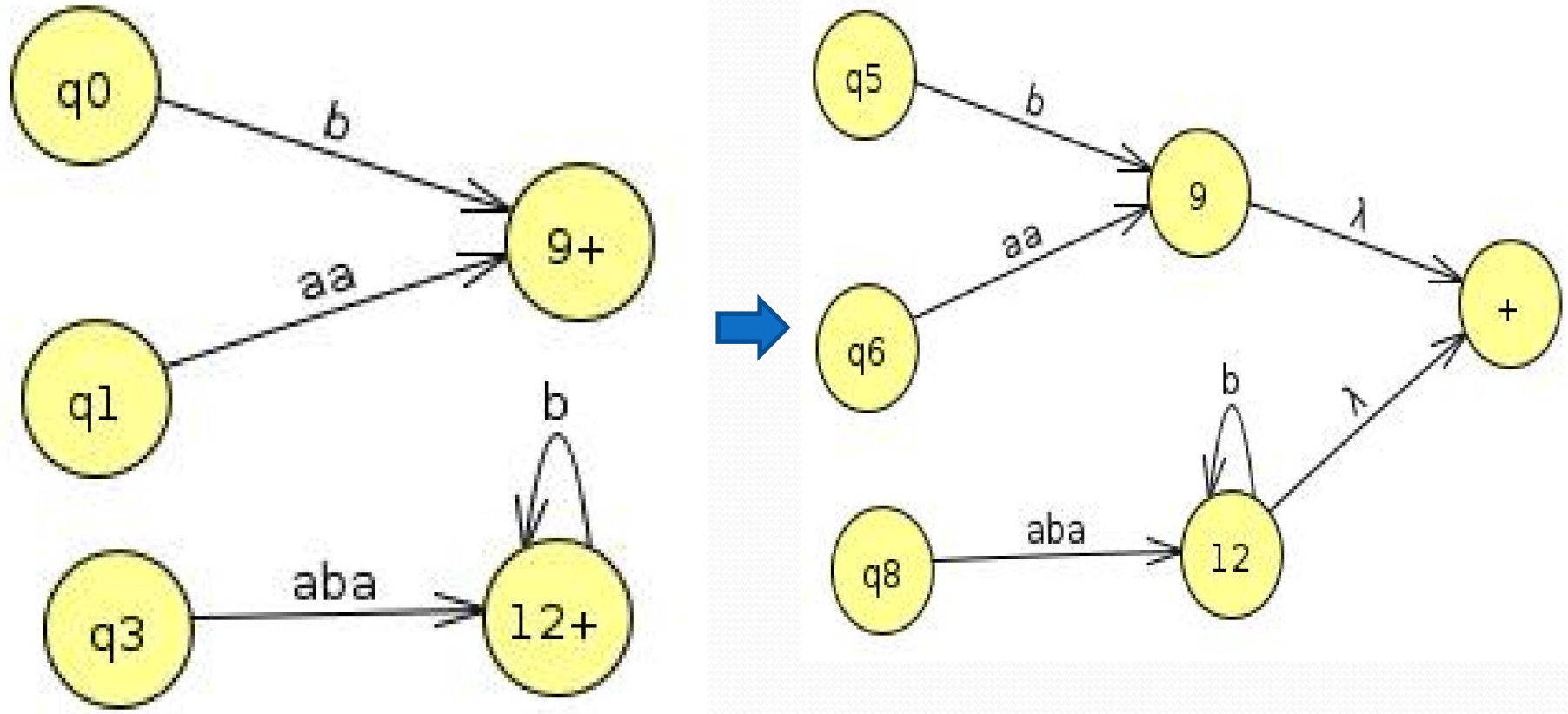
Kleene Theorem-PART-II

- **Step I: More than one Initial states conversion:**



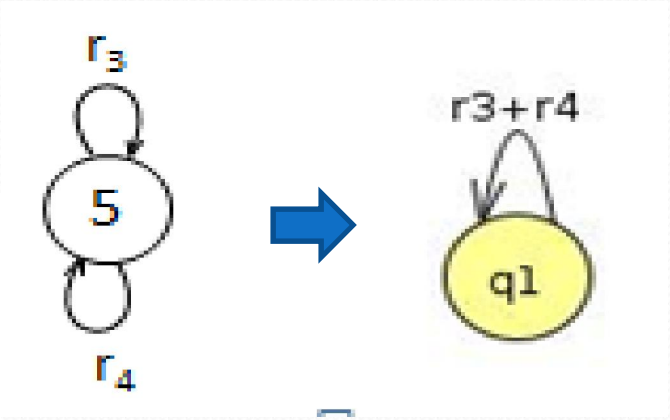
Kleene Theorem-PART-II

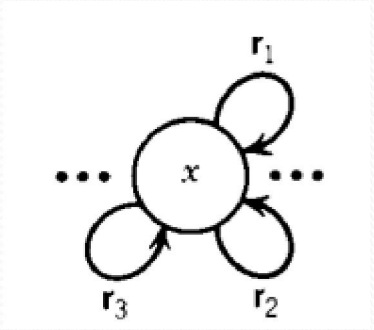
- **Step II: More than one Final states conversion:**

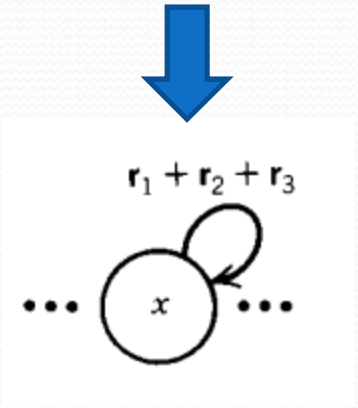


Kleene Theorem-PART-II

- Step III: Multiple Loops conversion:



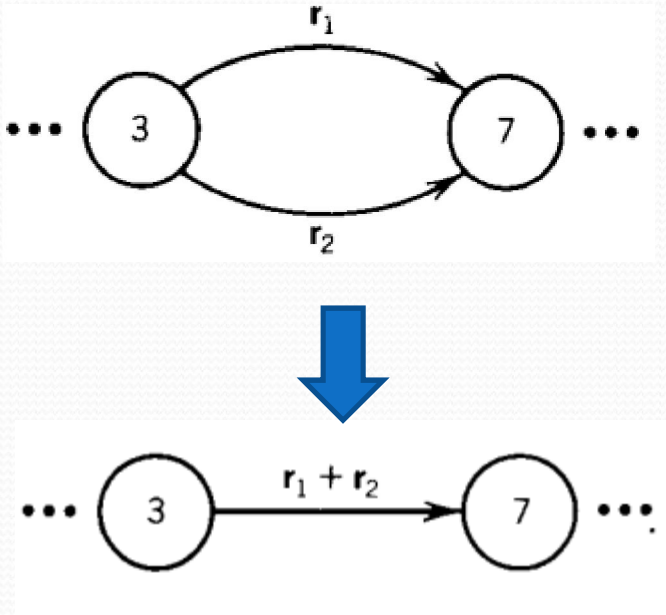




Kleene Theorem-PART-II

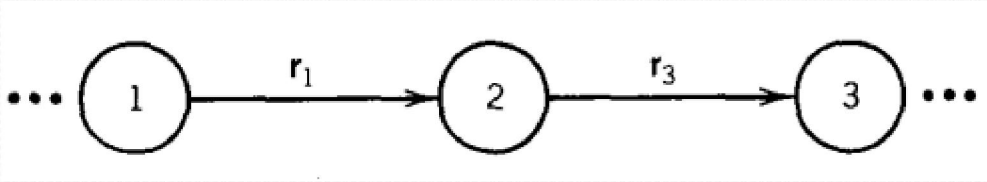
- Step 4: State Merging

(Alternative)

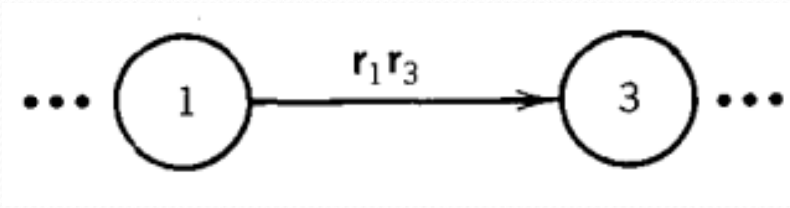


Kleene Theorem-PART-II

- State Merging

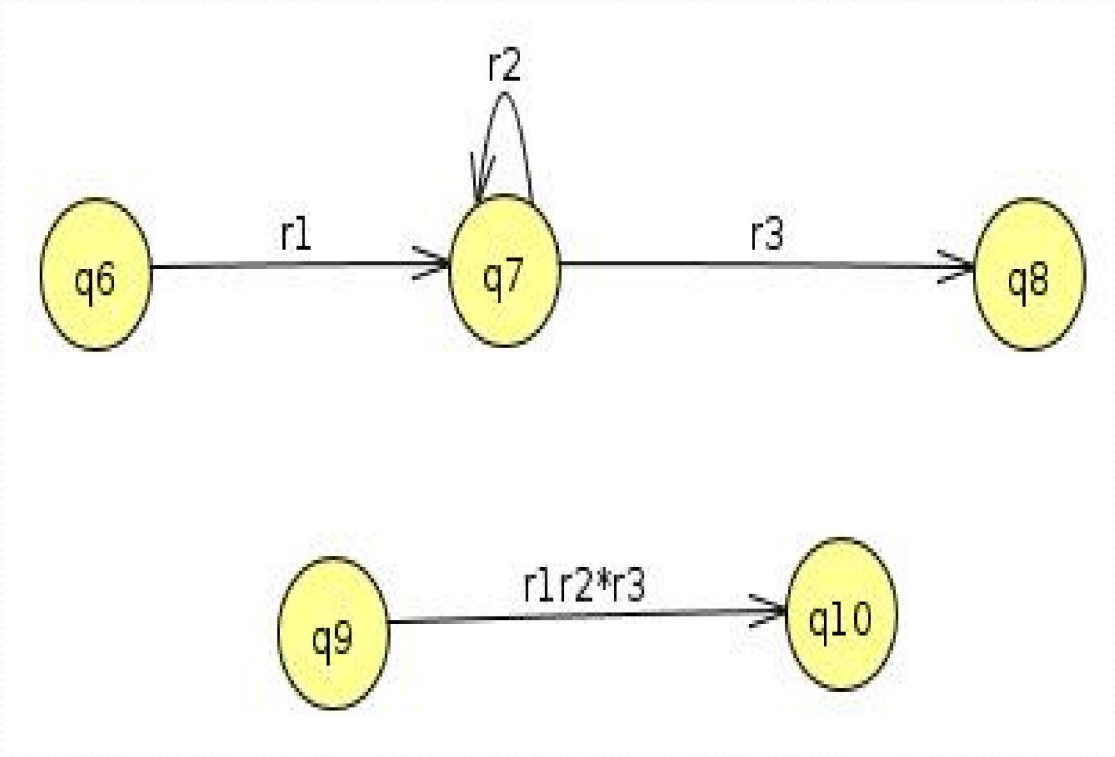


(Concatenation)



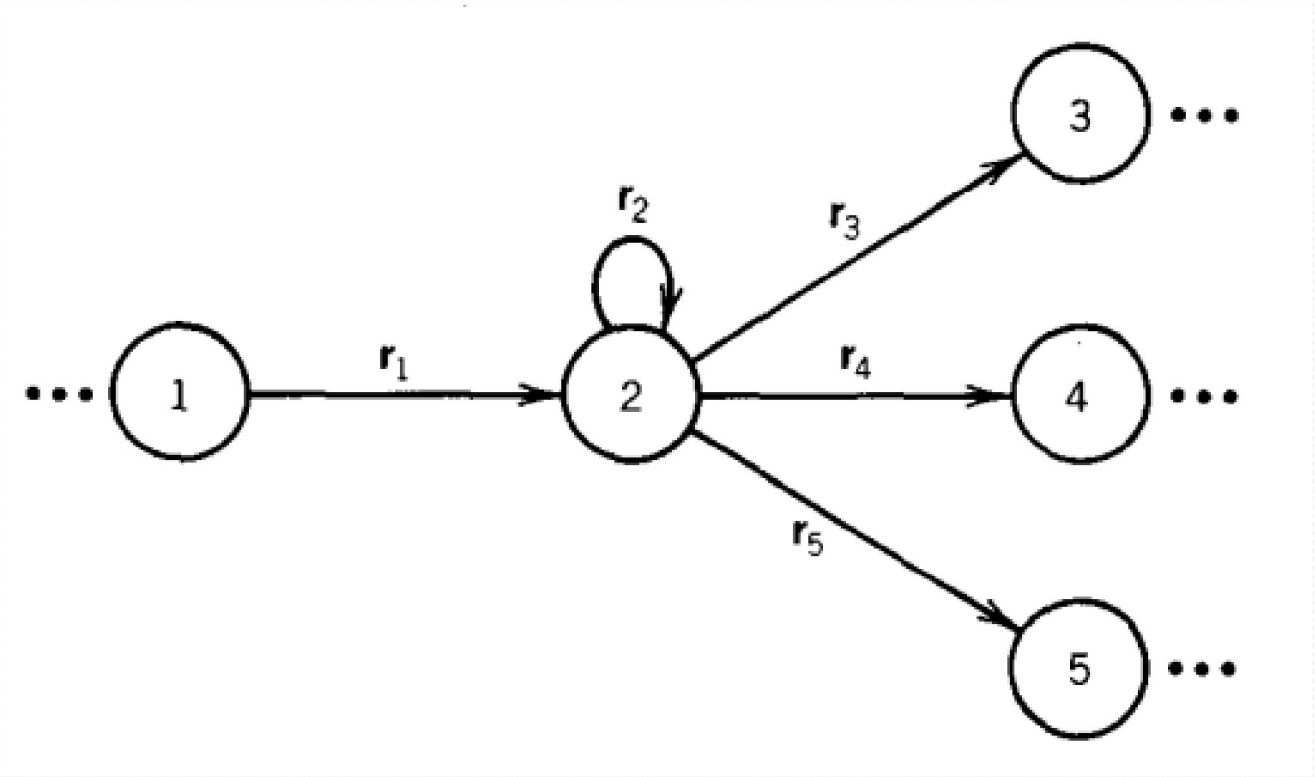
Kleene Theorem-PART-II

- State Merging (Remove self state Loops)



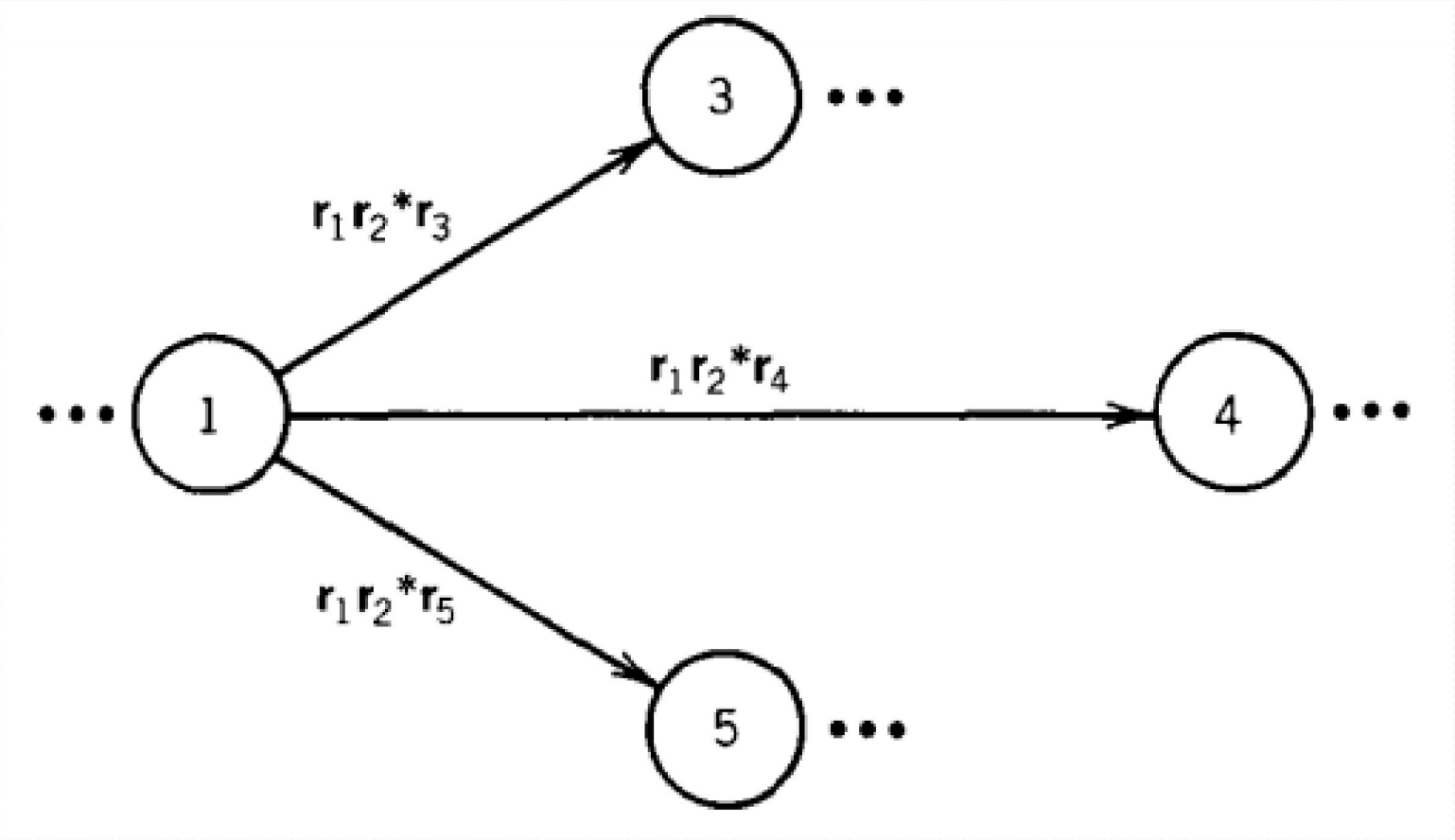
Kleene Theorem-PART-II

- State Merging

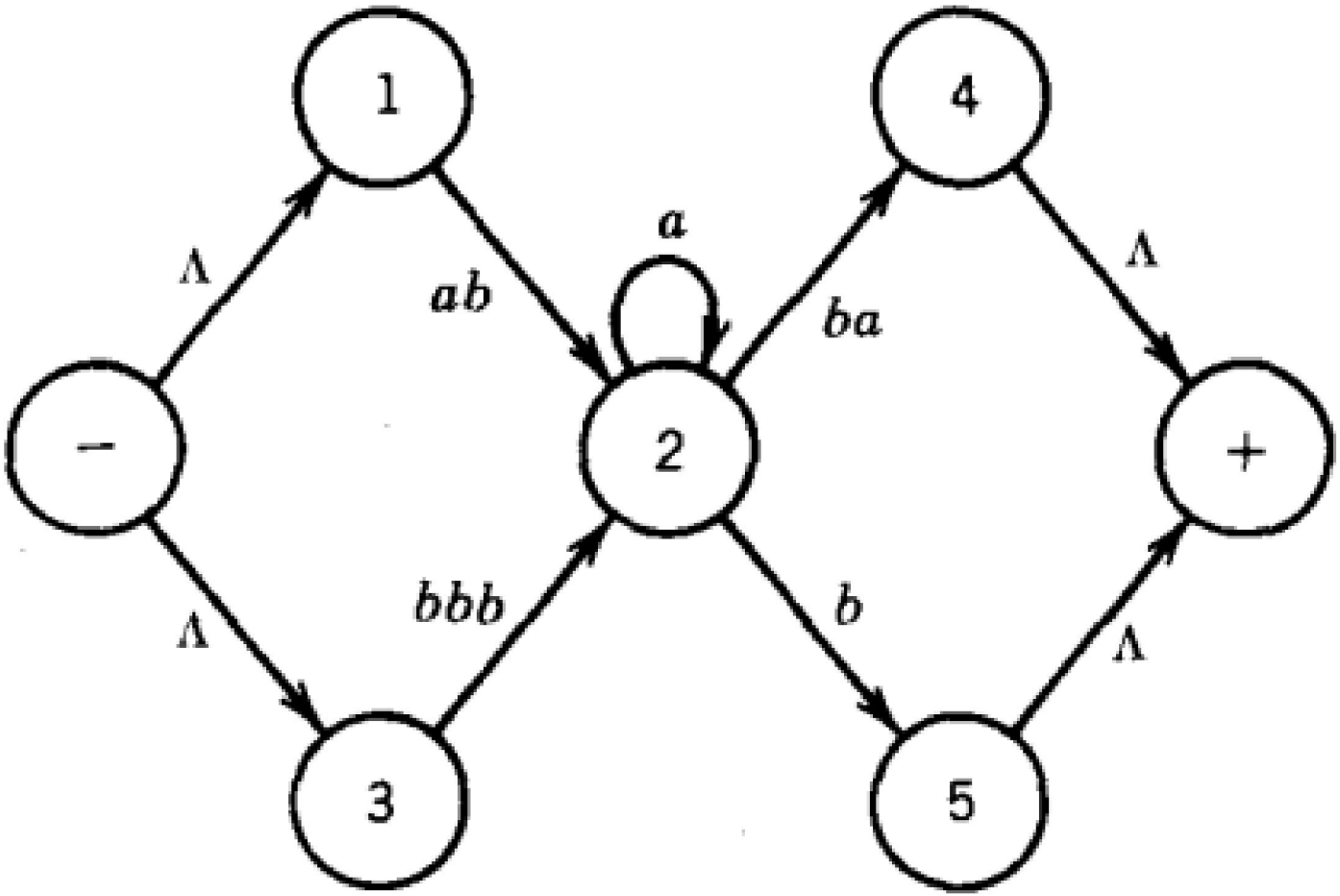


Kleene Theorem-PART-II

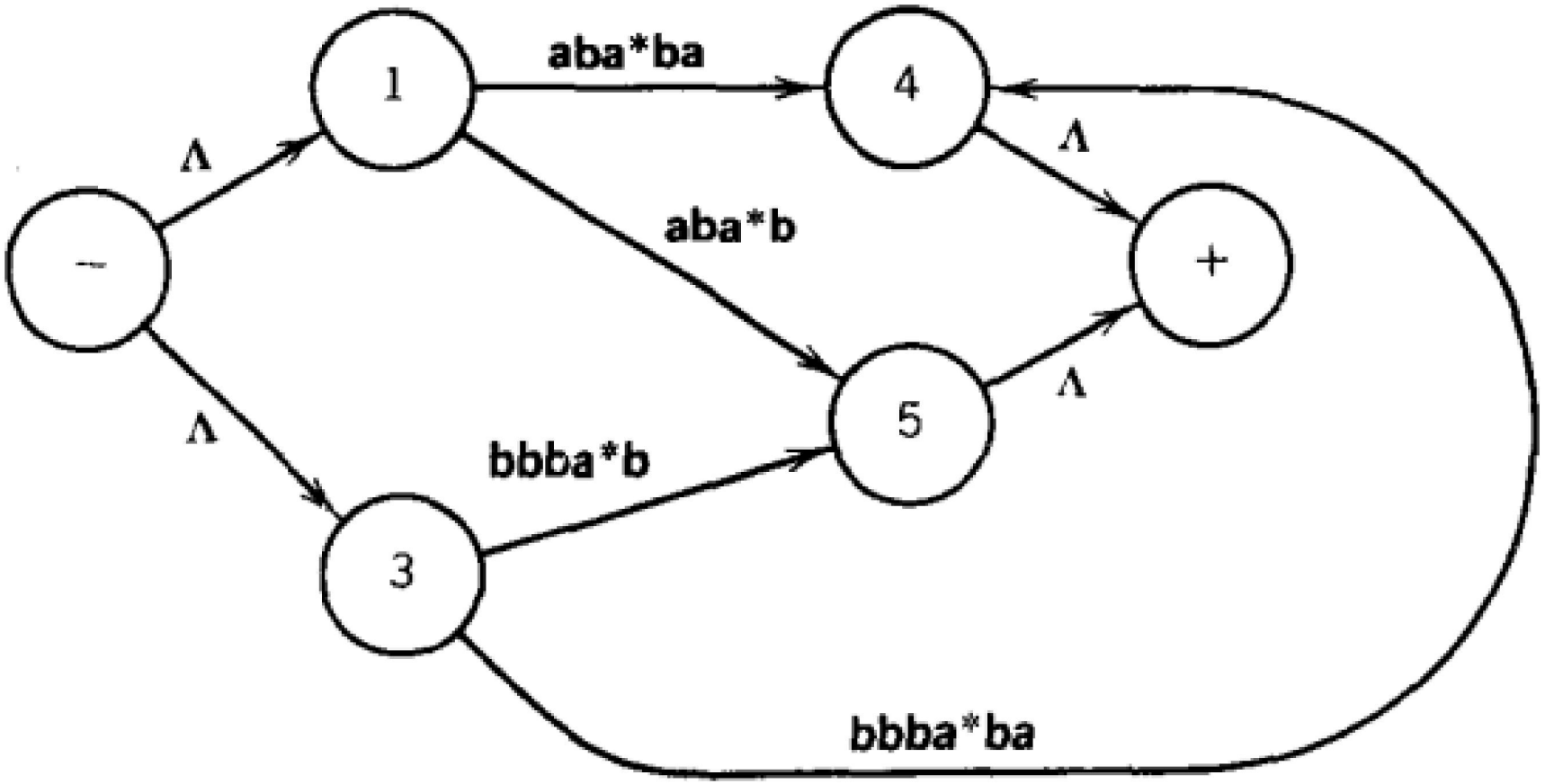
- State Merging



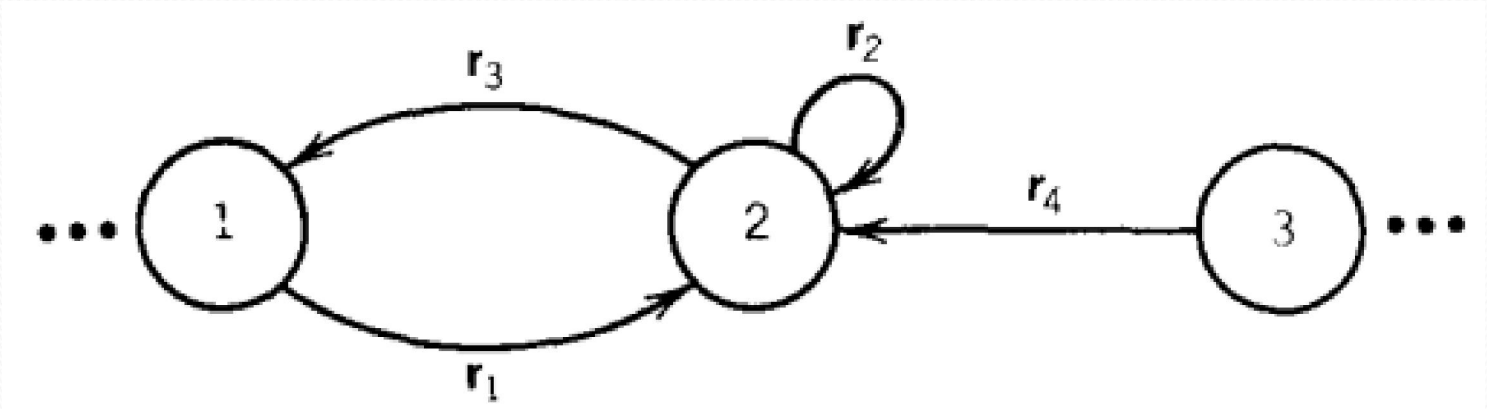
State elimination in Kleene Theorem

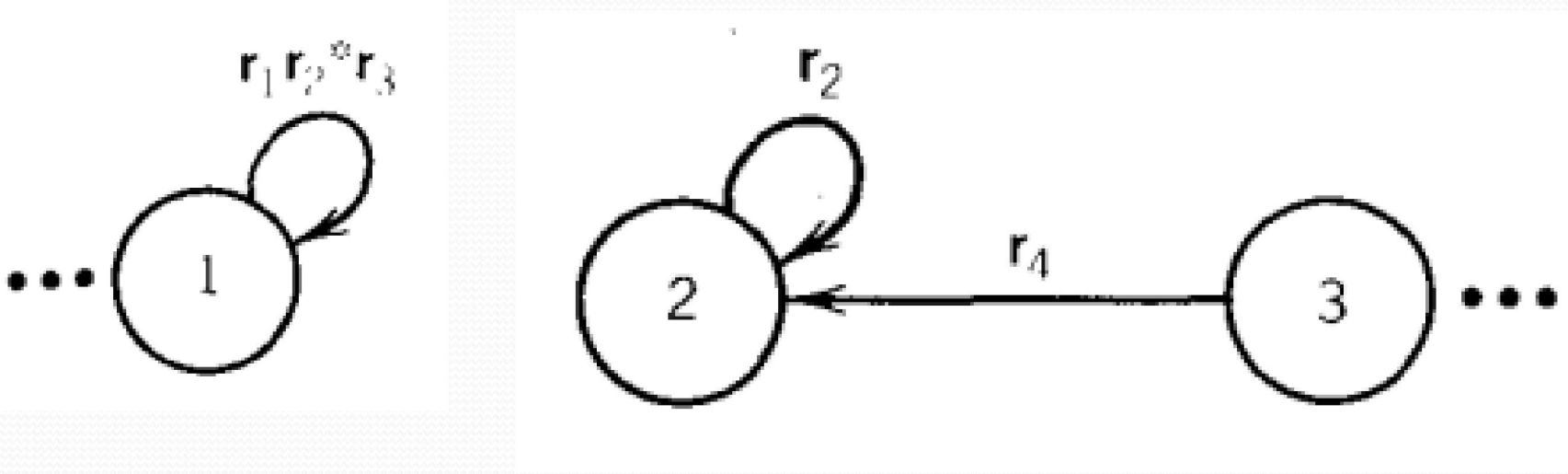


State elimination in Kleene Theorem

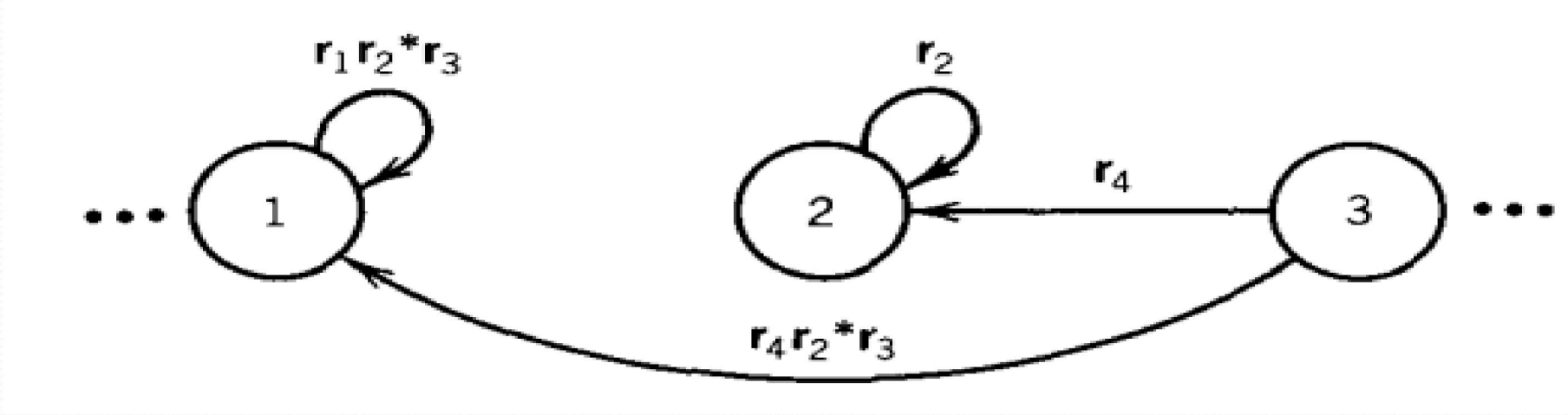


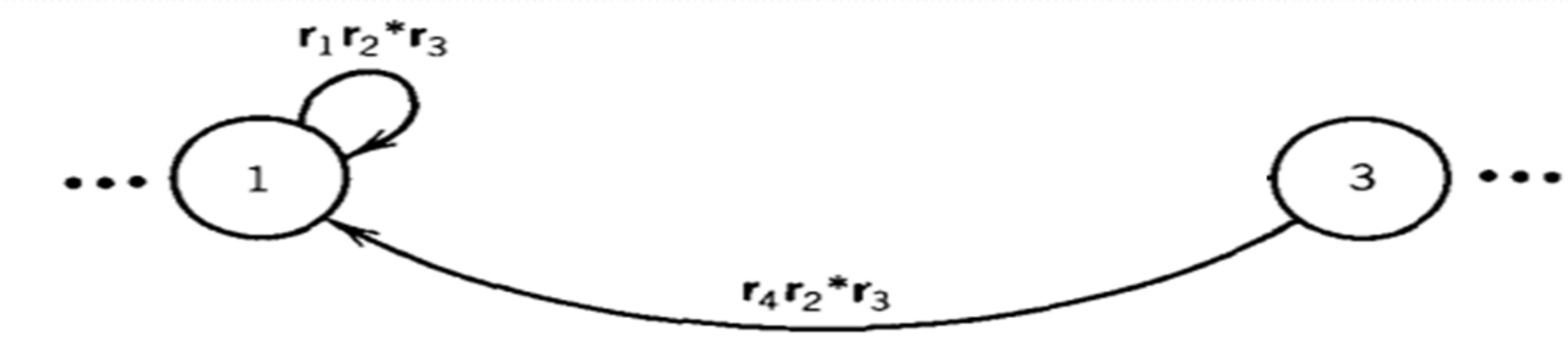
State elimination in Kleene Theorem



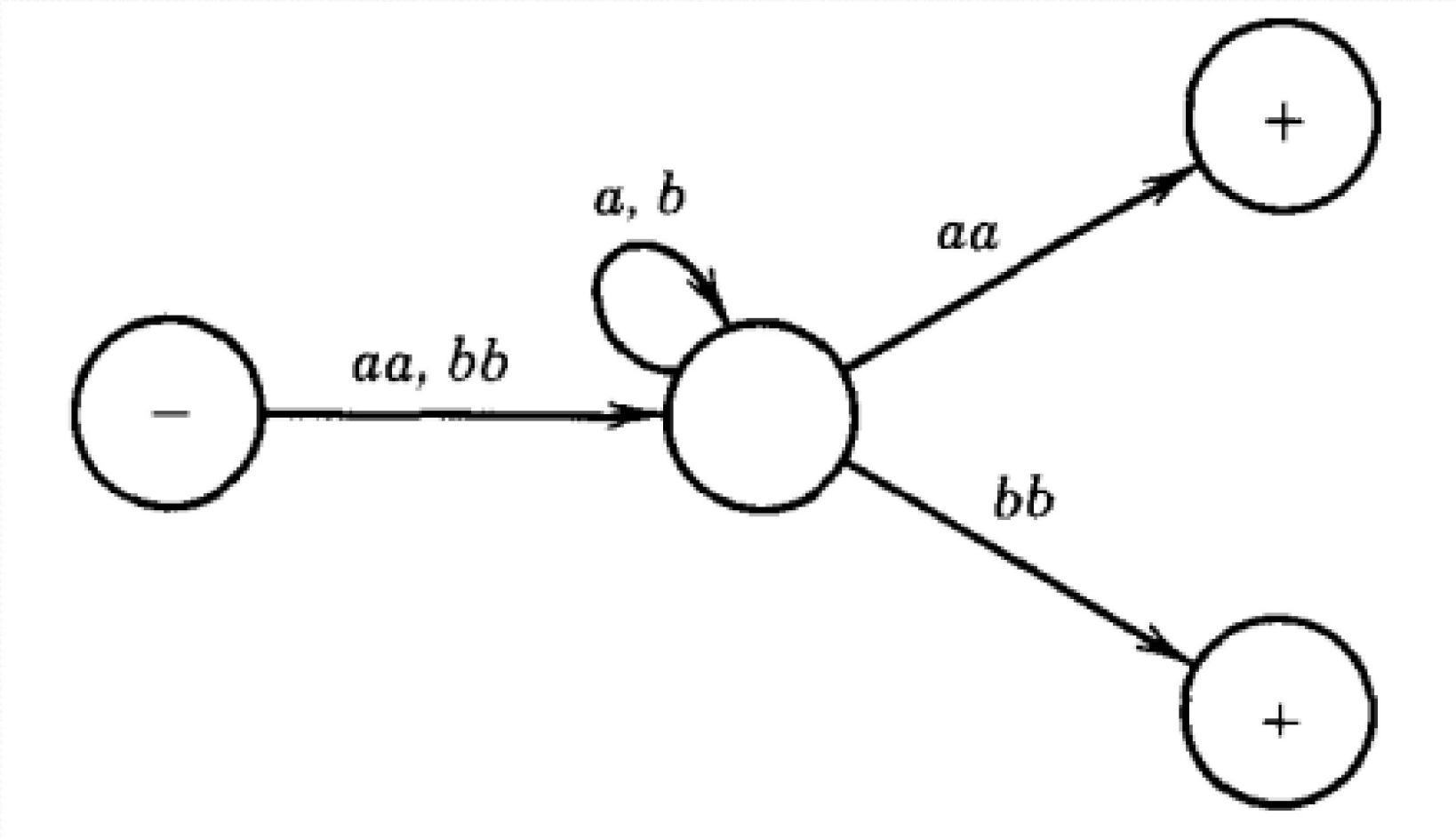


State elimination in Kleene Theorem

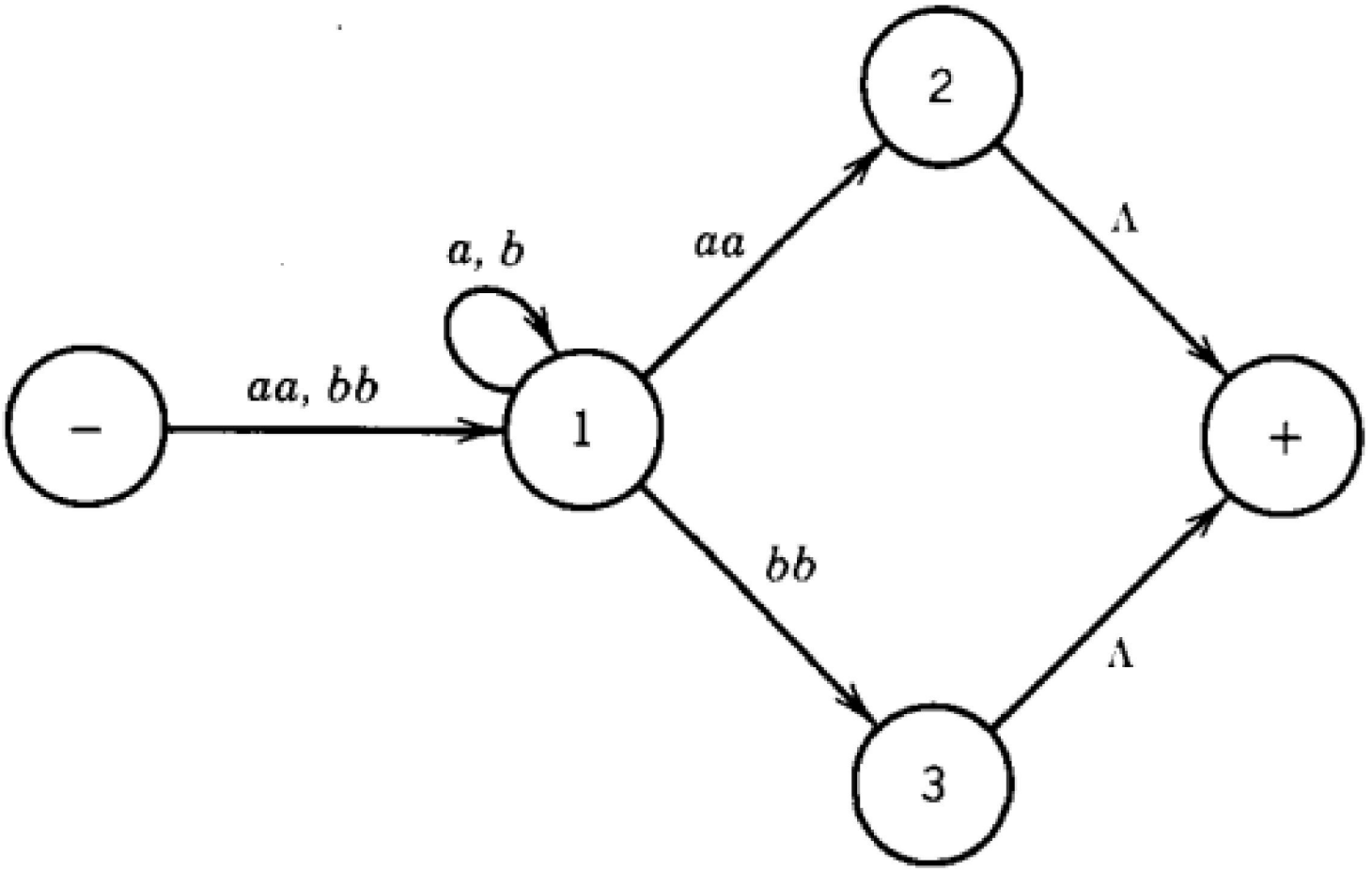




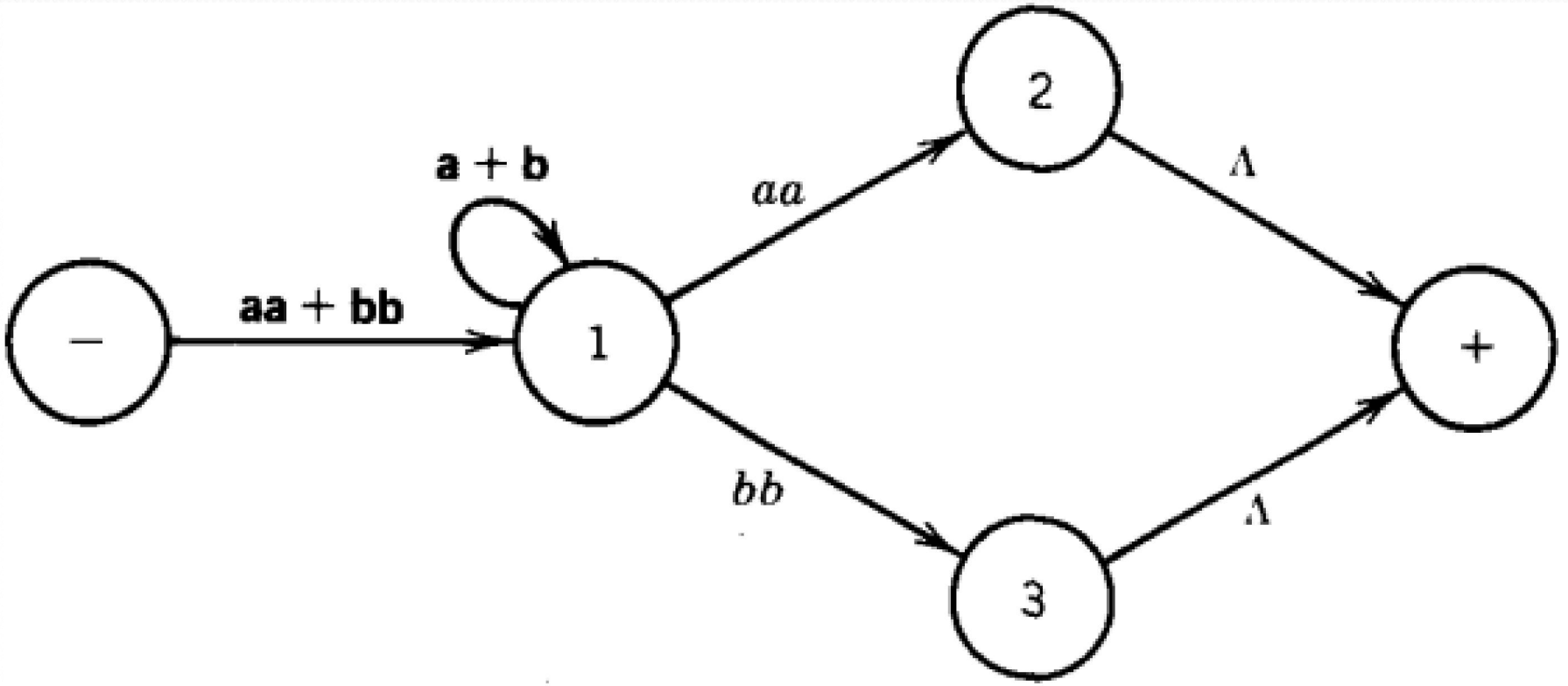
State elimination in Kleene Theorem



State elimination in Kleene Theorem



State elimination in Kleene Theorem



State elimination in Kleene Theorem