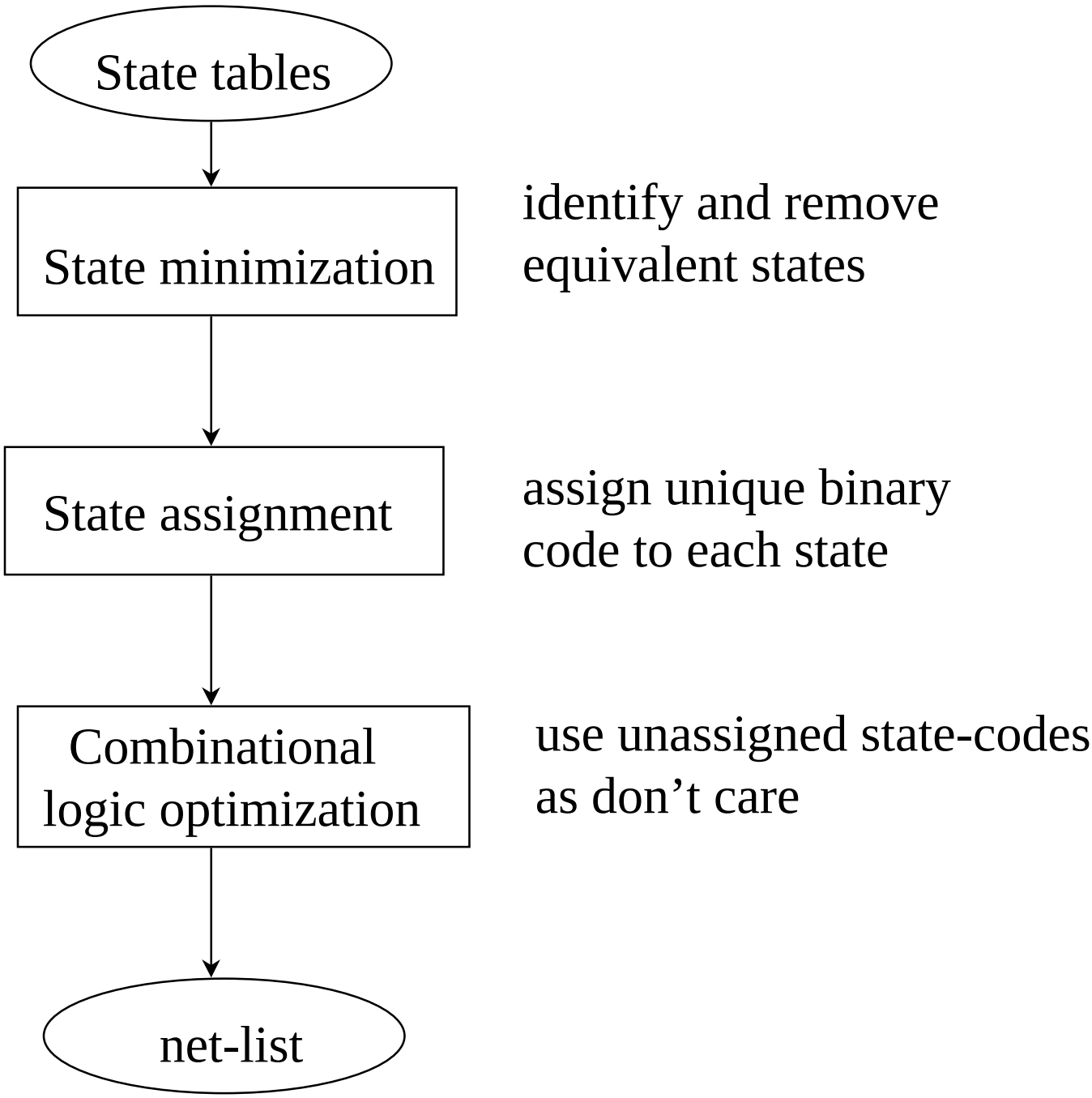
FSM (Finite State Machine) Optimization



State Minimization

Goal : identify and remove redundant states (states which can not be observed from the FSM I/O behavior)

Why : 1. **Reduce number of latches**

- assign minimum-length encoding
- only as the logarithm of the number of states

2. **Increase the number of unassigned state codes**

- heuristic to improve state-assignment and logic-optimization

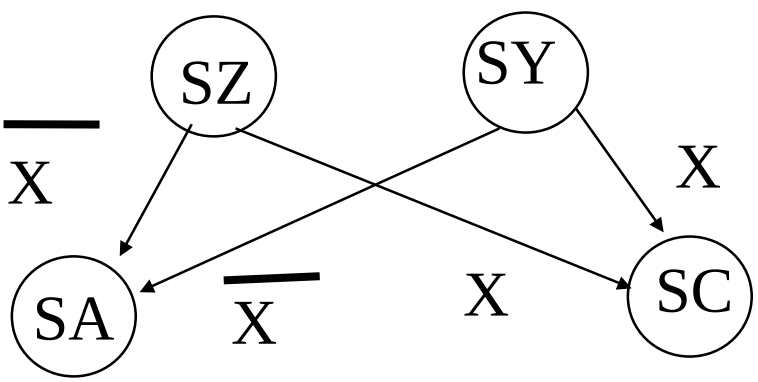
State Minimization Definition

- **Completely-specified state machine**
  - two states are equivalent if outputs are identical for all input combinations
    Next states are equivalent for all input combinations
  - equivalence of states is an equivalence relation which partitions the states into disjoint equivalence classes
- **Incompletely specified state machines**
  - **this problem is more difficult**

Basic Principle of State Minimization for Completely Specified Machines

Procedure (fast) for lazy students

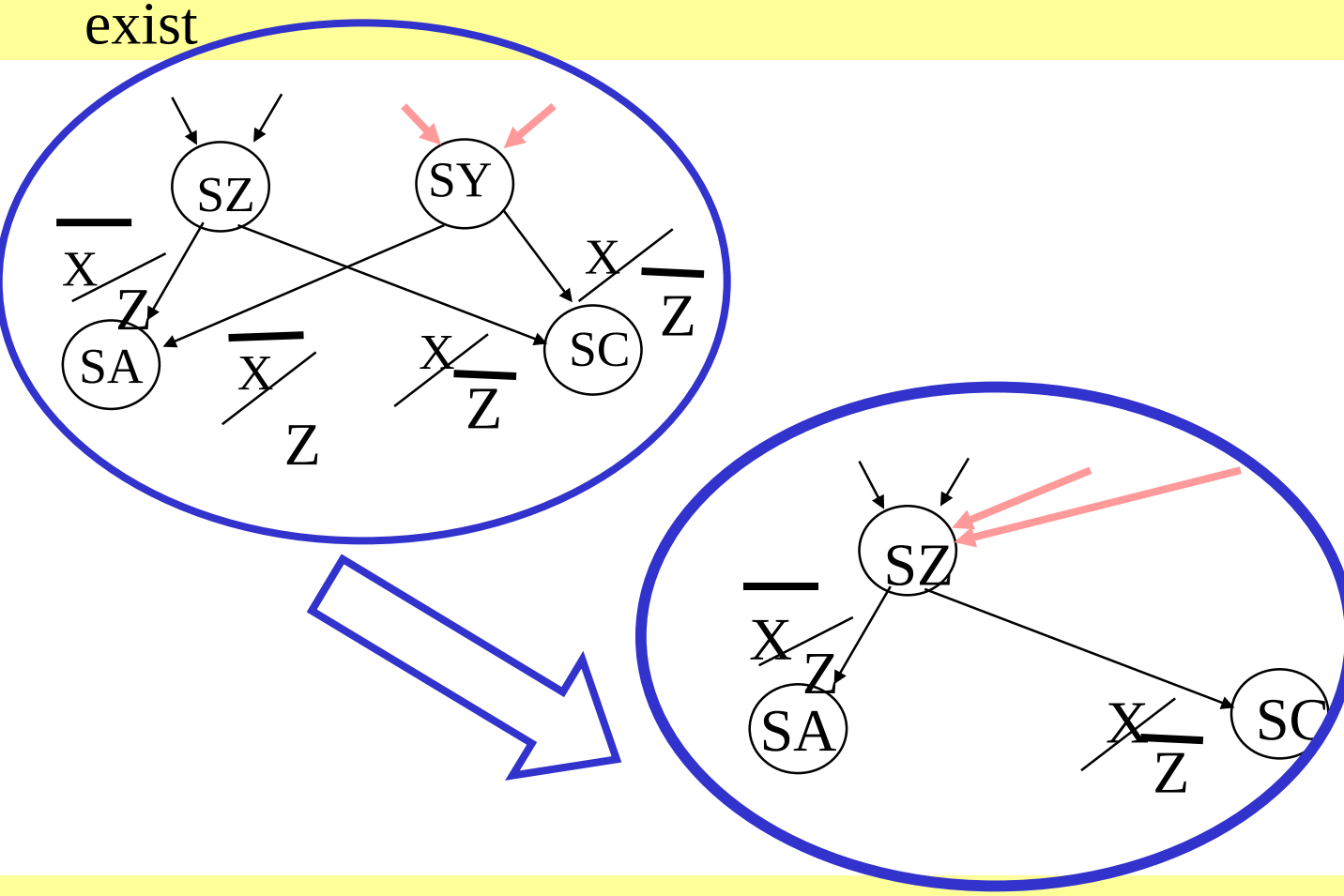
- Any two states of Moore Machine that have the same output and transit to the same states under the same input symbols are equivalent and can be combined
- This step is repeated until no more equivalent states exist



States SZ and SZ are equivalent and are combined to one state by pointing all arows that go to SY to state SZ and removing SY with its all arrows

Procedure (fast) for lazy students (for Mealy machines)

- Any two states of Mealy Machine that have the same output for the same input symbol and transit to the same states under the same input symbols are equivalent and can be combined
- This step is repeated until no more equivalent states exist



States SZ and SZ are equivalent and are combined to one state by pointing all arows that go to SY to state SZ and removing SY with its all arrows

Classical State Minimization Algorithm

Only for Completely specified Machines

1. **Partition the set of internal states based on input output values asserted in the state**
2. **Define the partitions so that all states in a partition transition into the same next-state partition (under corresponding inputs)**

Example (FSM in Kiss format)

Ex :

0 A B 0
1 A C 0
0 B D 0
1 B E 0
0 C F 0
1 C A 0
0 D H 0
1 D G 0
0 E B 0
1 E C 0
0 F D 0
1 F E 0
0 G F 1
1 G A 0
0 H H 0
1 H A 0

G has other input-output response than other states

(A,B,C,D,E,F,H) (G)

D has other input-output response than other states because it goes to G which is known to be non-equivalent state- goes to red and blue groups

(A,B,C,E,F,H)(G)(D)

(A,C,E)(G)(D)(B,F)

B and F go to D

States A, C and E can be combined to one state

States B and F can be combined to one state

Please check this using triangular table

You can also marke each new group with a new symbol and check transitions to thus marked groups