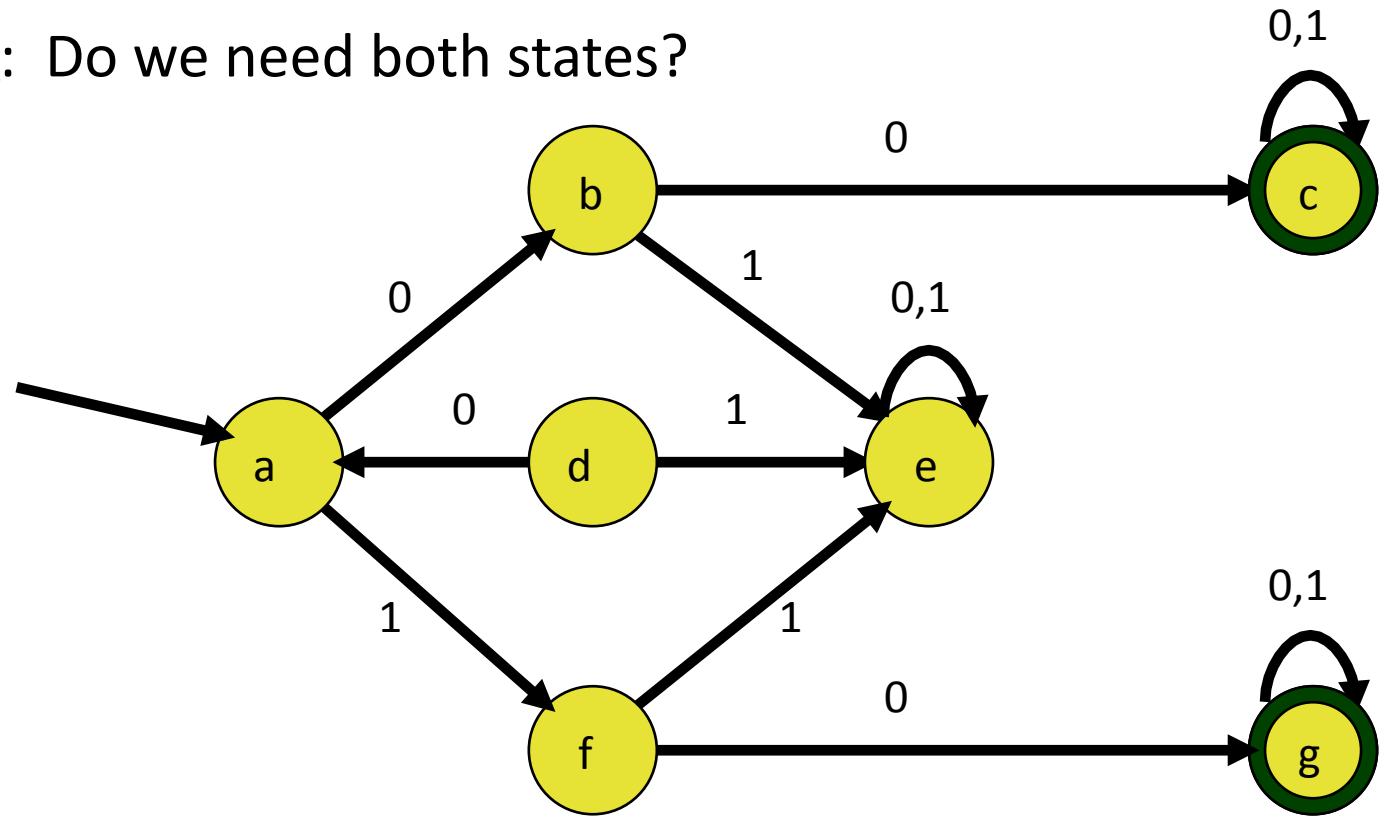
Minimizing Automata
Myhill-Nerode Theorem

Taken from:
COMS 3261: Computability, Fall 2004
Columbia University
Zeph Grunschlag

Equivalent States. Example

Consider the accept states c and g. They are both sinks meaning that any string which ever reaches them is guaranteed to be accepted later.

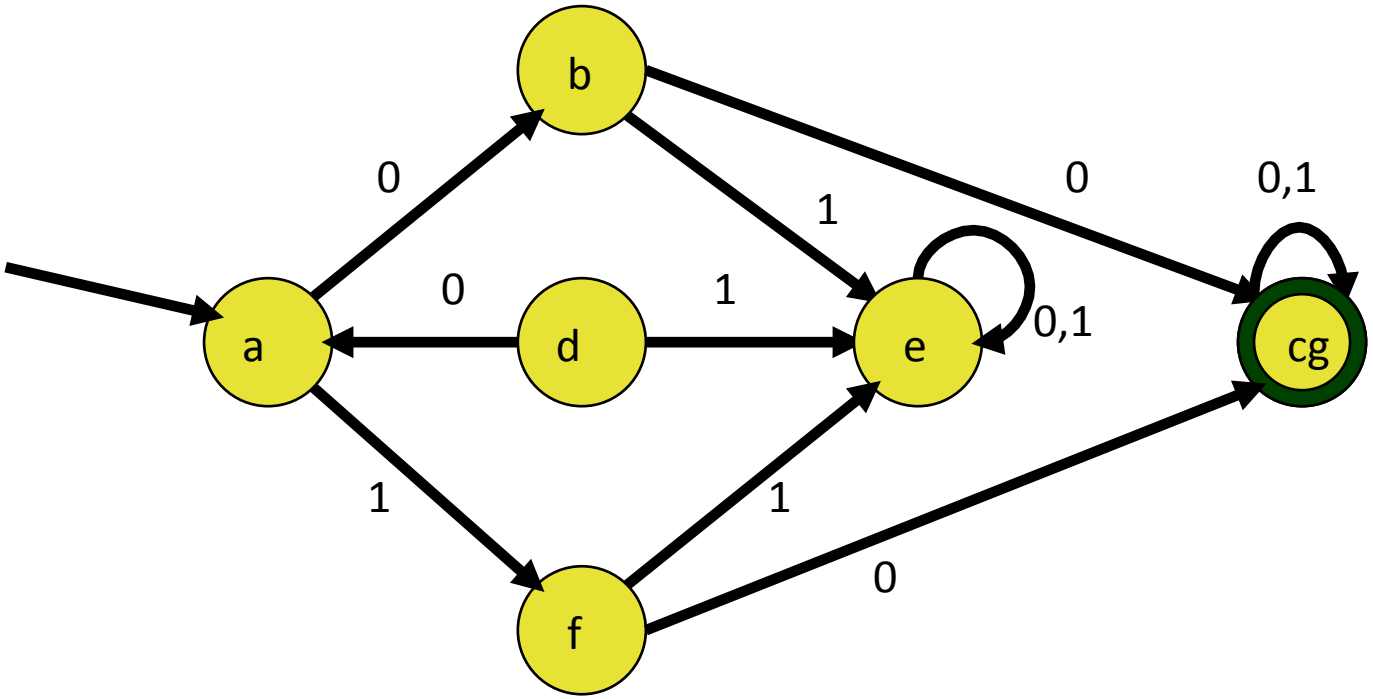
Q: Do we need both states?



Equivalent States. Example

A: No, they can be unified as illustrated below.

Q: Can any other states be unified because any subsequent string suffixes produce identical results?

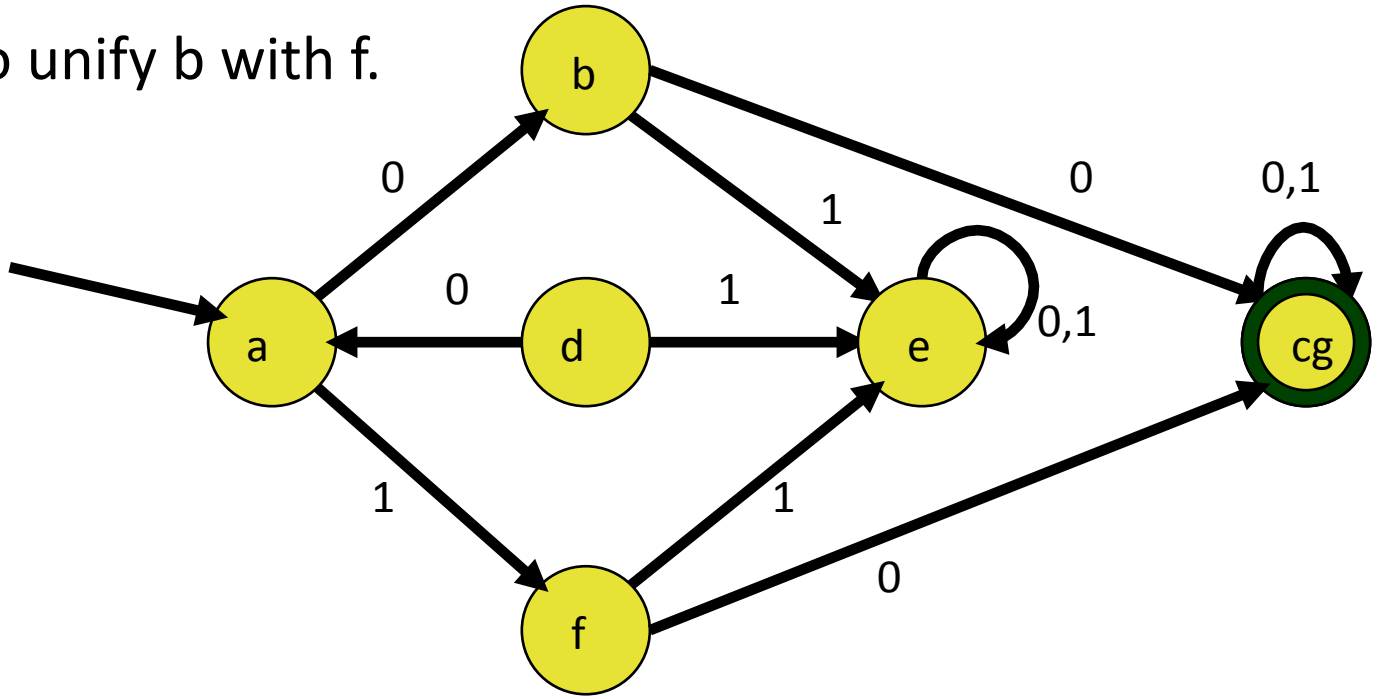


Equivalent States. Example

A: Yes, b and f. Notice that if you're in b or f then:

1. if string ends, reject in both cases
2. if next character is 0, forever accept in both cases
3. if next character is 1, forever reject in both cases

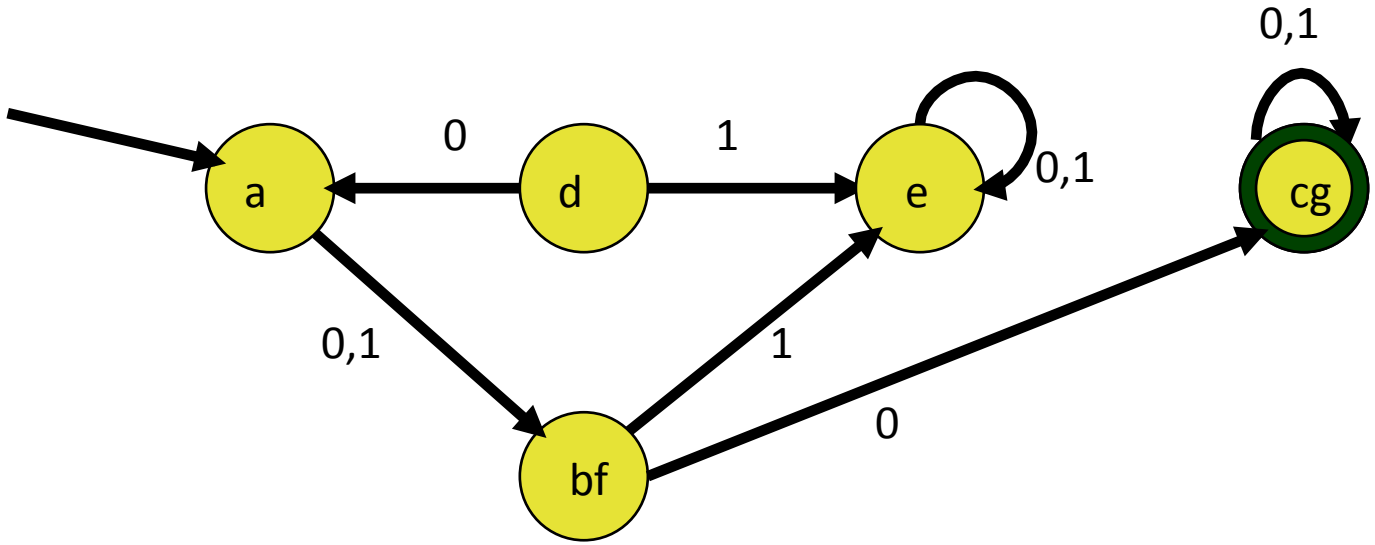
So unify b with f.



Equivalent States. Example

Intuitively two states are equivalent if all subsequent behavior from those states is the same.

Q: Come up with a formal characterization of state equivalence.



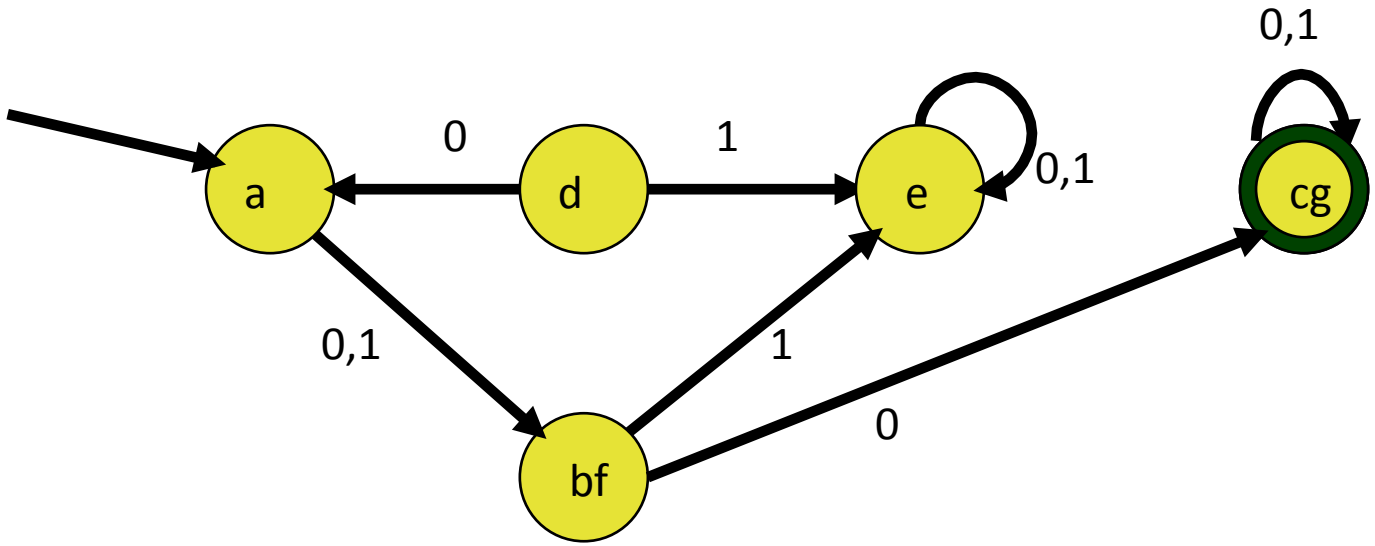
Equivalent States. Definition

DEF: Two states q and q' in a DFA $M = (Q, \Sigma, \delta, q_0, F)$ are said to be ***equivalent*** (or ***indistinguishable***) if for all strings $u \in \Sigma^*$, the states on which u ends on when read from q and q' are both accept, or both non-accept.

Equivalent states may be glued together without affecting M's behavior.

Finishing the Example

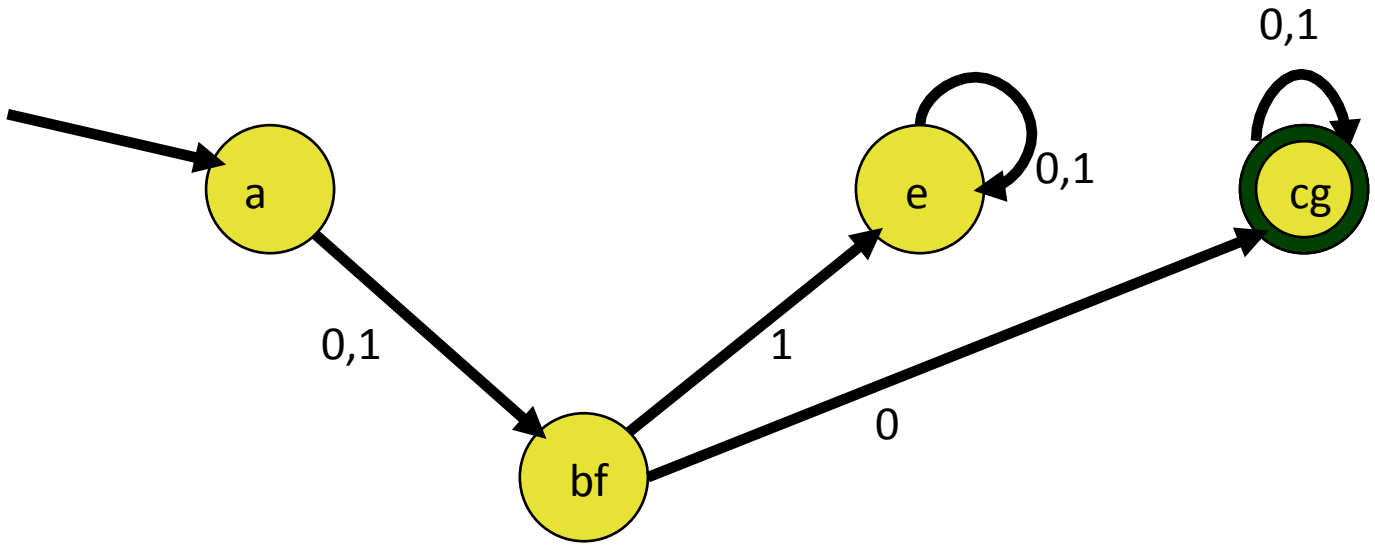
Q: Any other ways to simplify the automaton?



Useless States

A: Get rid of d.

Getting rid of unreachable ***useless states*** doesn't affect the accepted language.



Minimization Algorithm. Goals

DEF: An automaton is ***irreducible*** if

- it contains no useless states, and
- no two distinct states are equivalent.

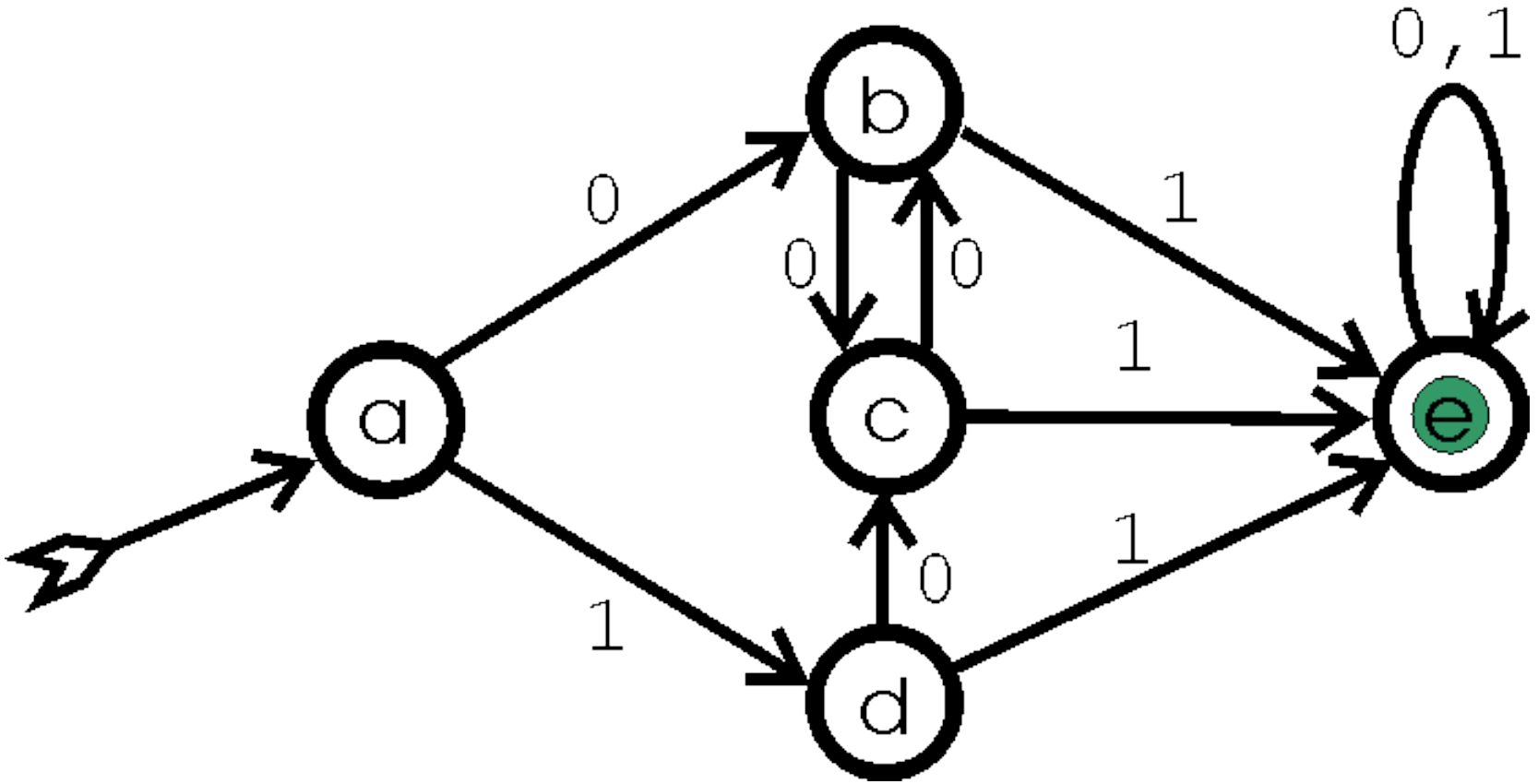
The goal of minimization algorithm is to create irreducible automata from arbitrary ones.

Remarkably, the algorithm actually produces smallest possible DFA for the given language, hence the name "minimization".

The minimization algorithm *reverses* previous example. Start with least possible number of states, and create new states when forced to.

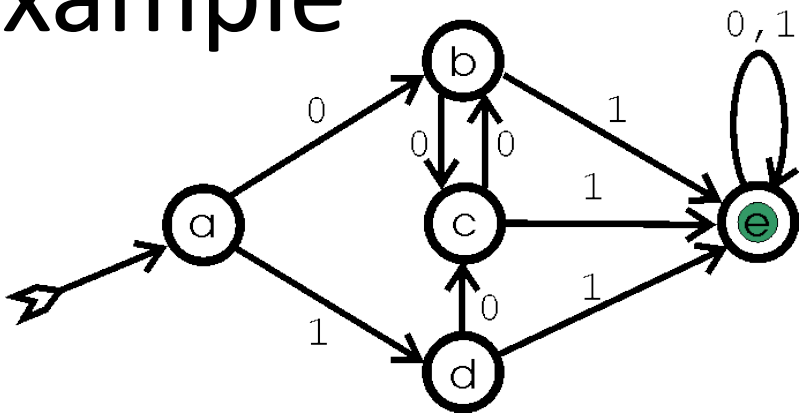
Minimization Example

Start with a DFA



Minimization Example

Overview version →



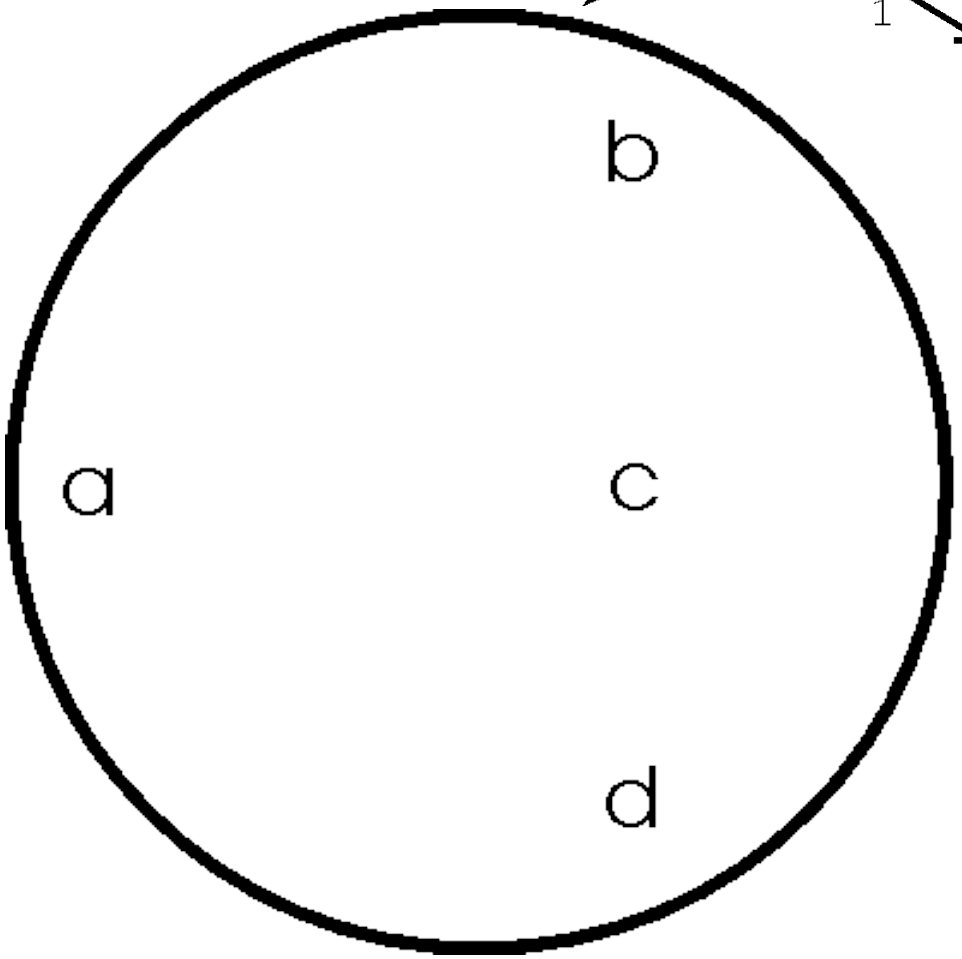
Minimization Example

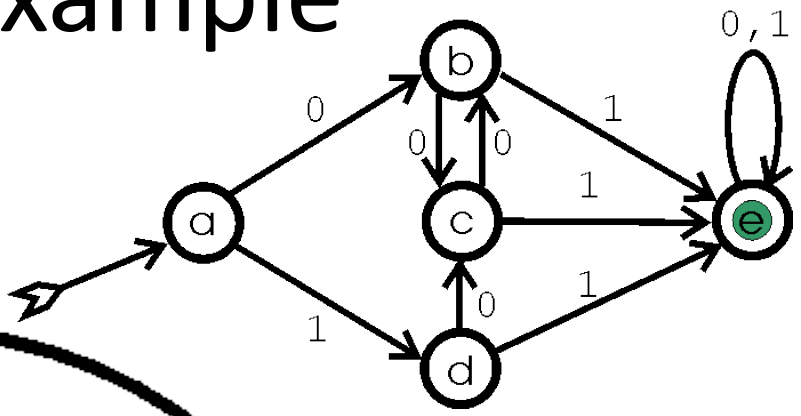
Split into two teams.

REJECT

vs.

ACCEPT

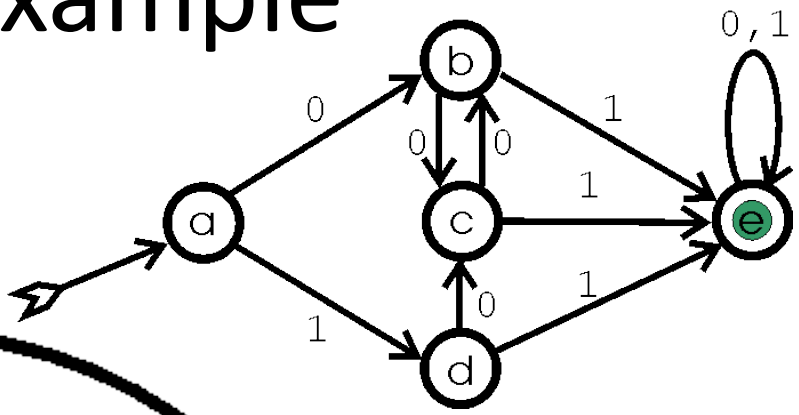


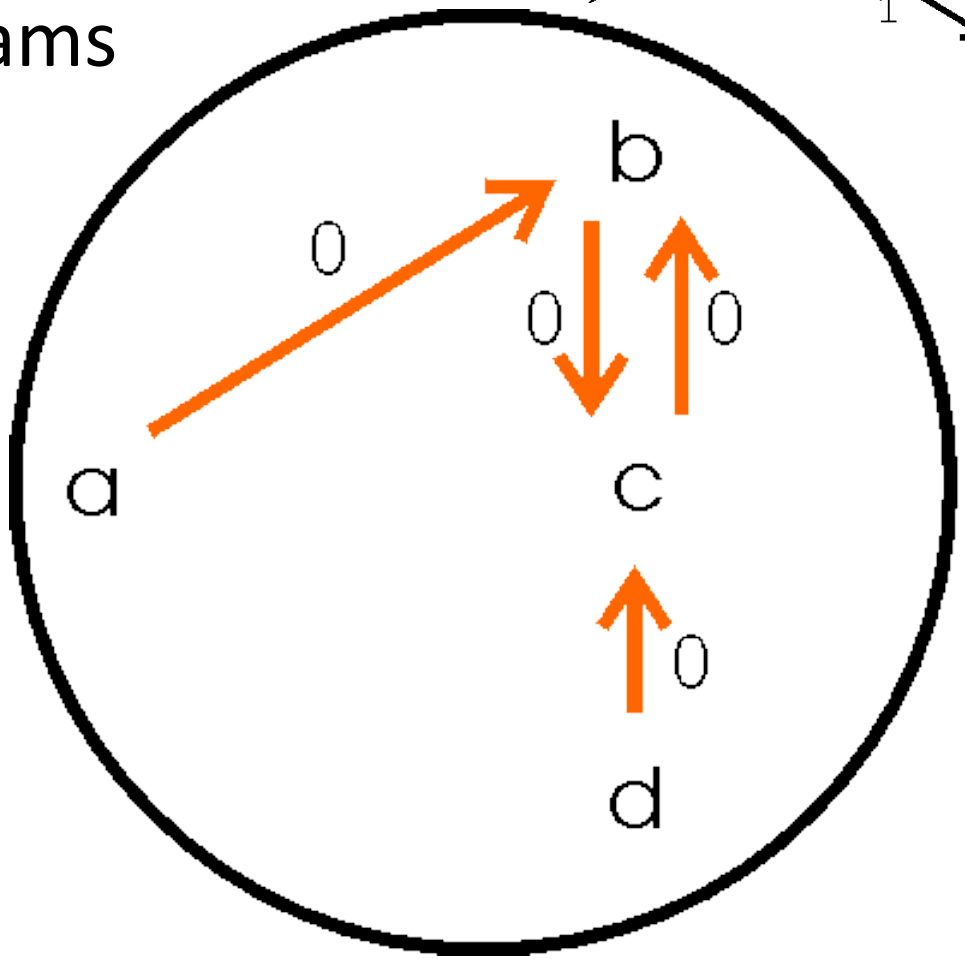


Minimization Example

0-label doesn't split

up any teams

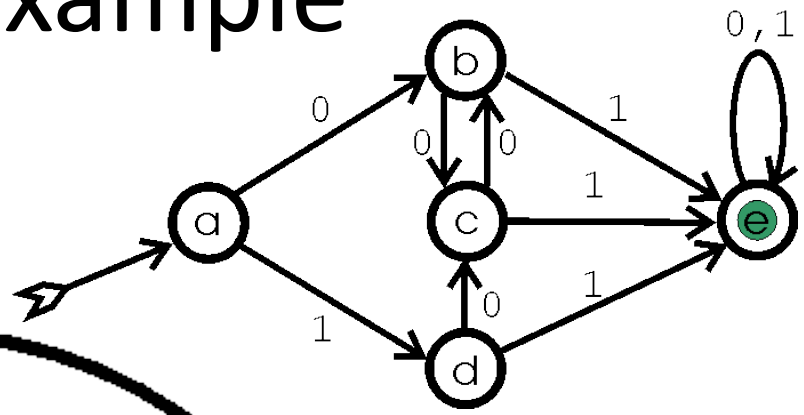


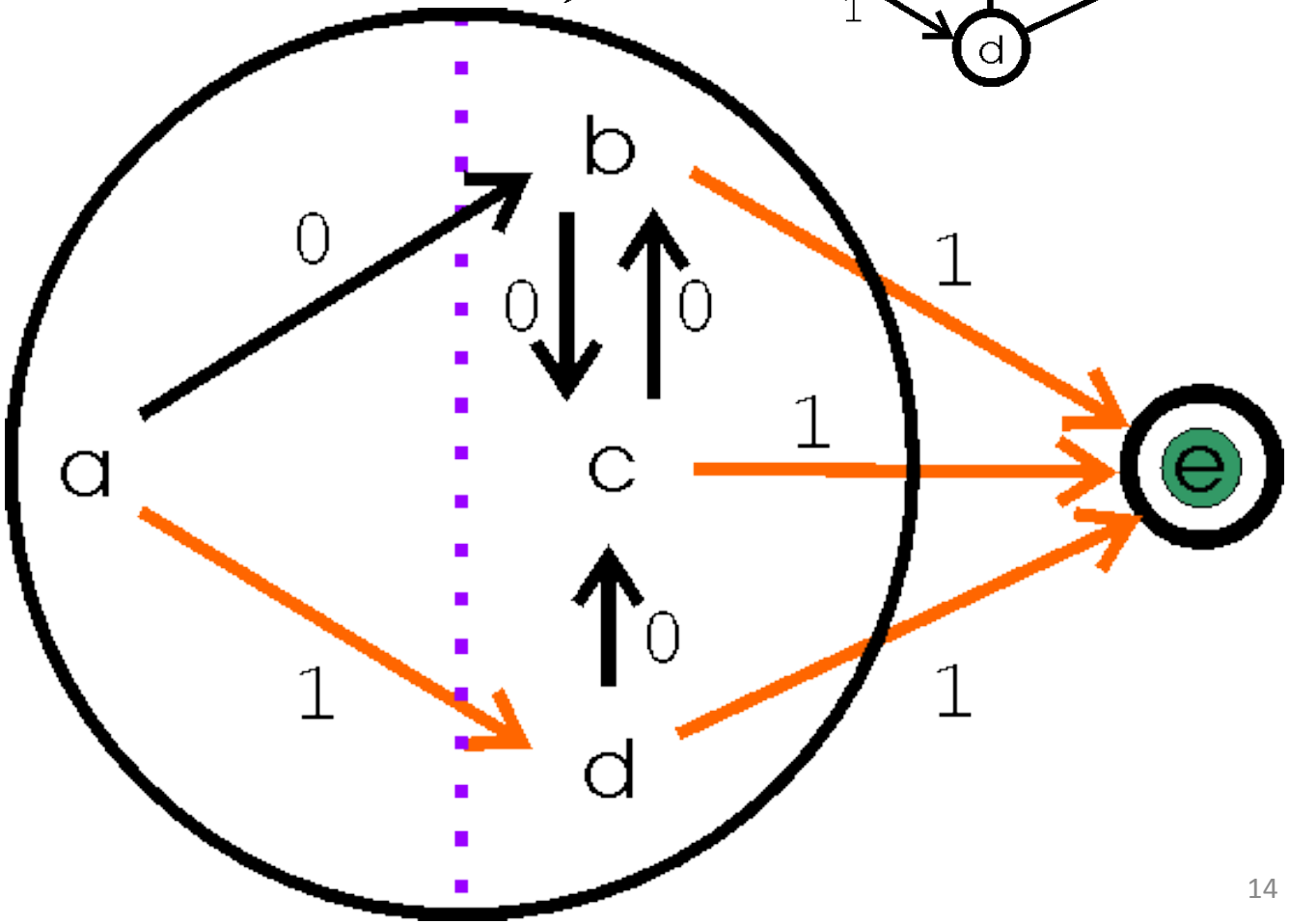


Minimization Example

1-label splits up

REJECT's

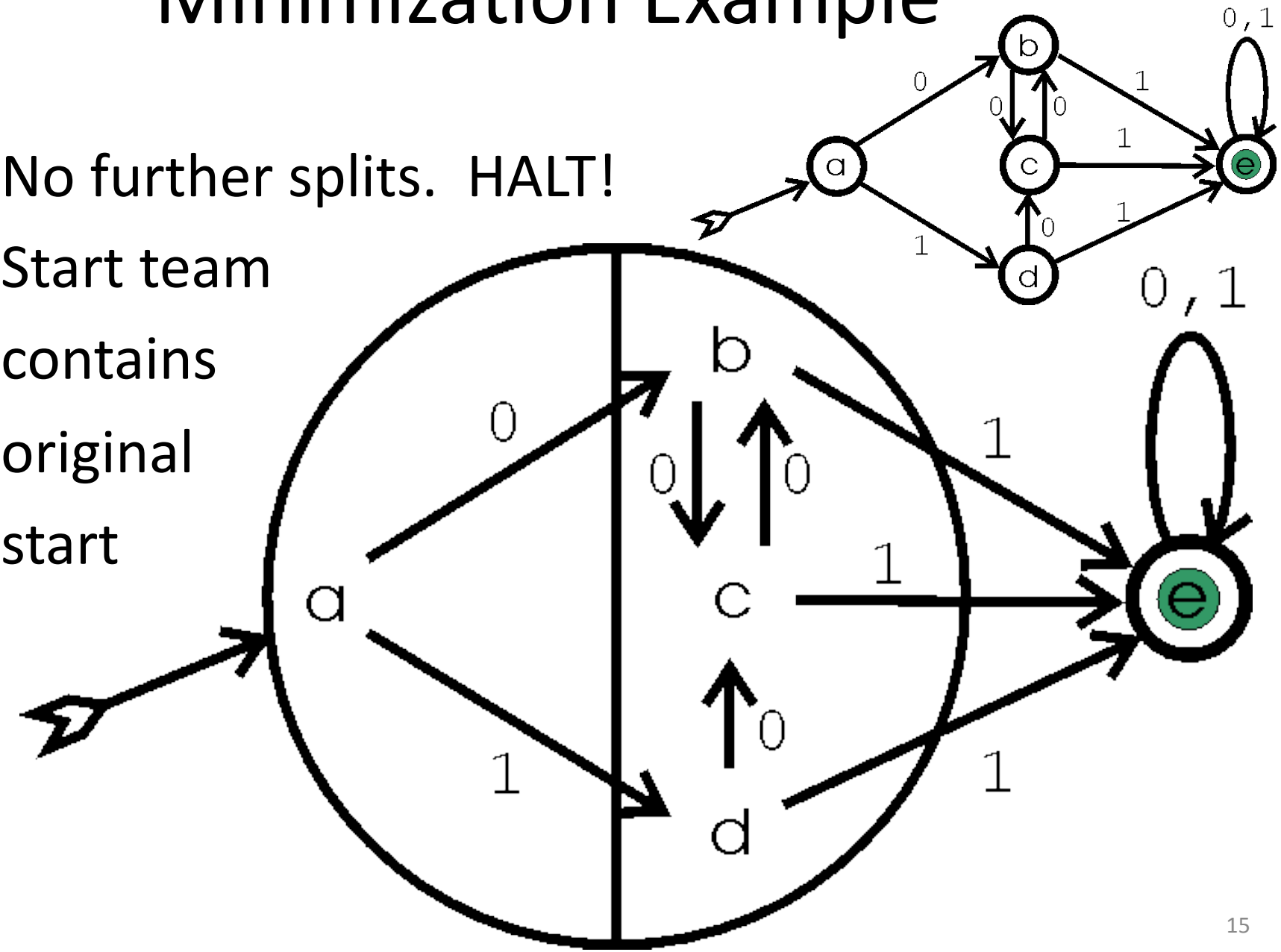




Minimization Example

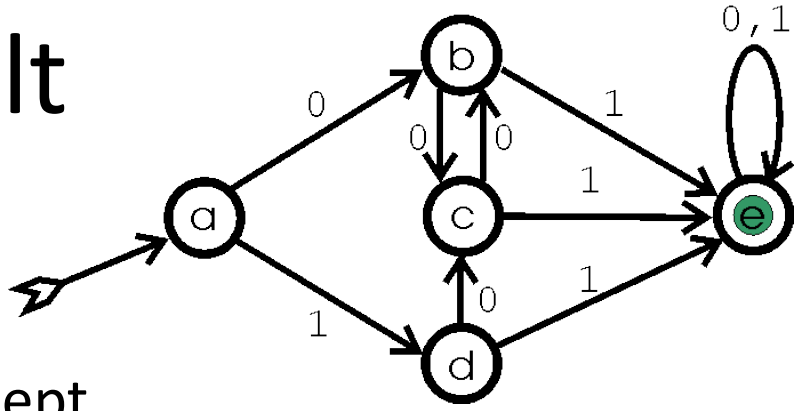
No further splits. HALT!

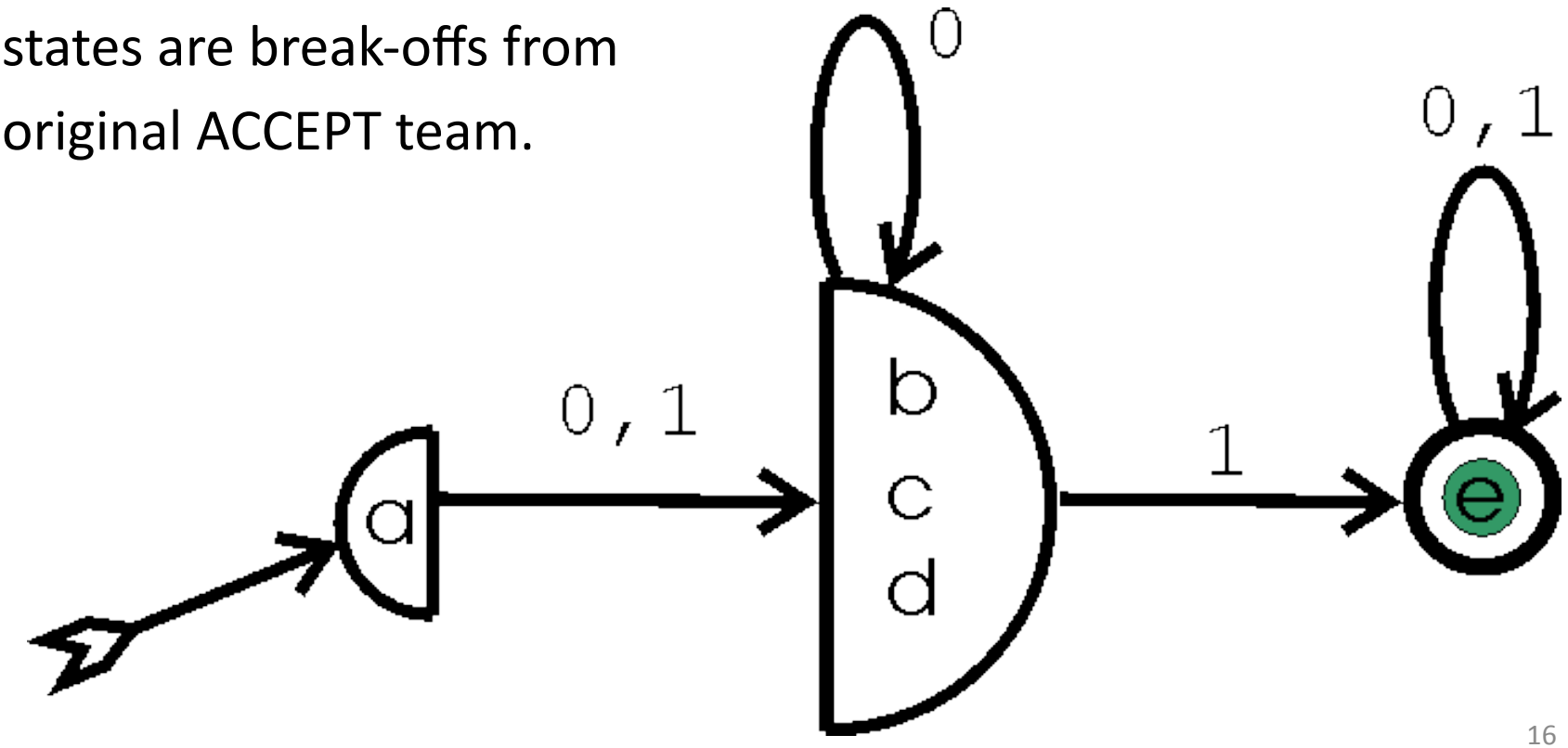
Start team contains original start



Minimization Example. End Result

States of the minimal automata are remaining teams. Edges are consolidated across each team. Accept states are break-offs from original ACCEPT team.





Minimization Algorithm. (Partition Refinement) Code

DFA minimize(DFA (Q, Σ, δ, q_0, F))

remove any state q unreachable from q_0

Partition P = {F, Q - F }

boolean Consistent = false

while(Consistent == false)

Consistent = true

forevery (Set $S \in P$, char $a \in \Sigma$, Set $T \in P$)

Set temp = {$q \in T$ | $\delta(q,a) \in S$ }

if (temp != $\emptyset$ && temp != T)

Consistent = false

P = (P -T)$\cup${temp,T-temp}

return defineMinimizor((Q, Σ, δ, q_0, F), P)

Minimization Algorithm. (Partition Refinement) Code

DFA defineMinimizor

(DFA (Q, Σ, δ, q_0, F), Partition P)

Set $Q' = P$

State q'_0 = the set in P which contains q_0

$F' = \{ S \in P \mid S \subseteq F \}$

for (each $S \in P$, $a \in \Sigma$)

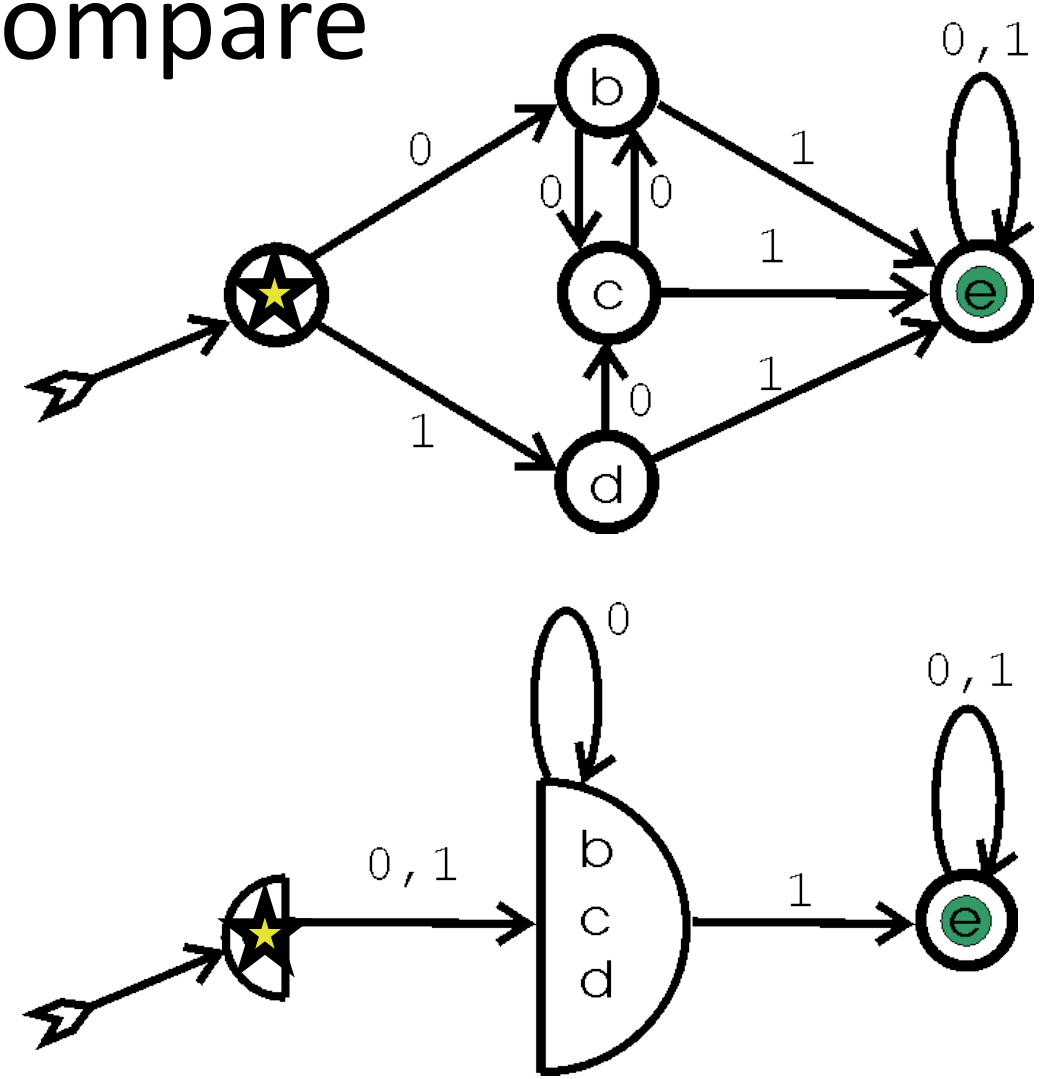
define $\delta'(S,a)$ = the set $T \in P$ which contains the states $\delta'(S,a)$

return (Q', Σ, δ', q'_0, F')

Minimization Example.

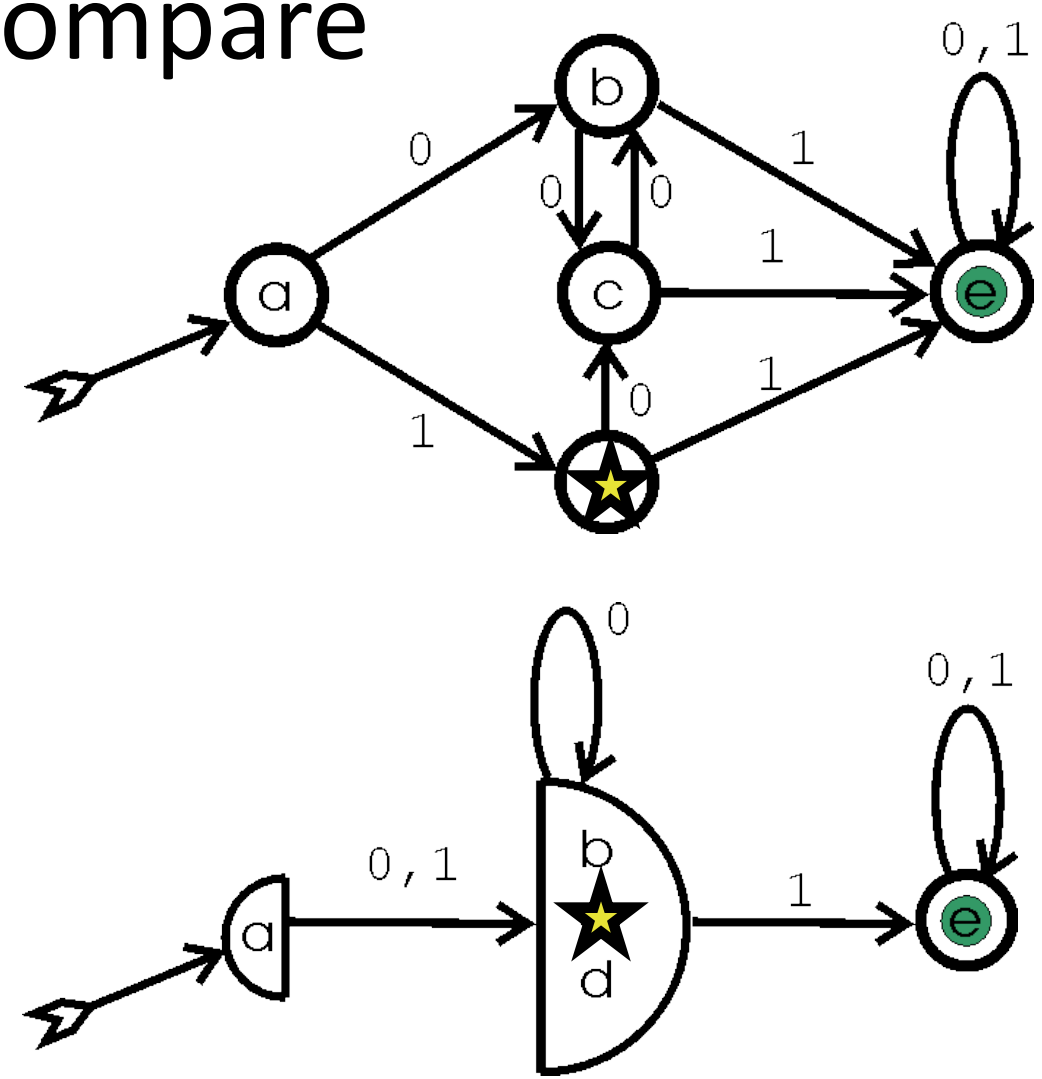
Compare

100100101



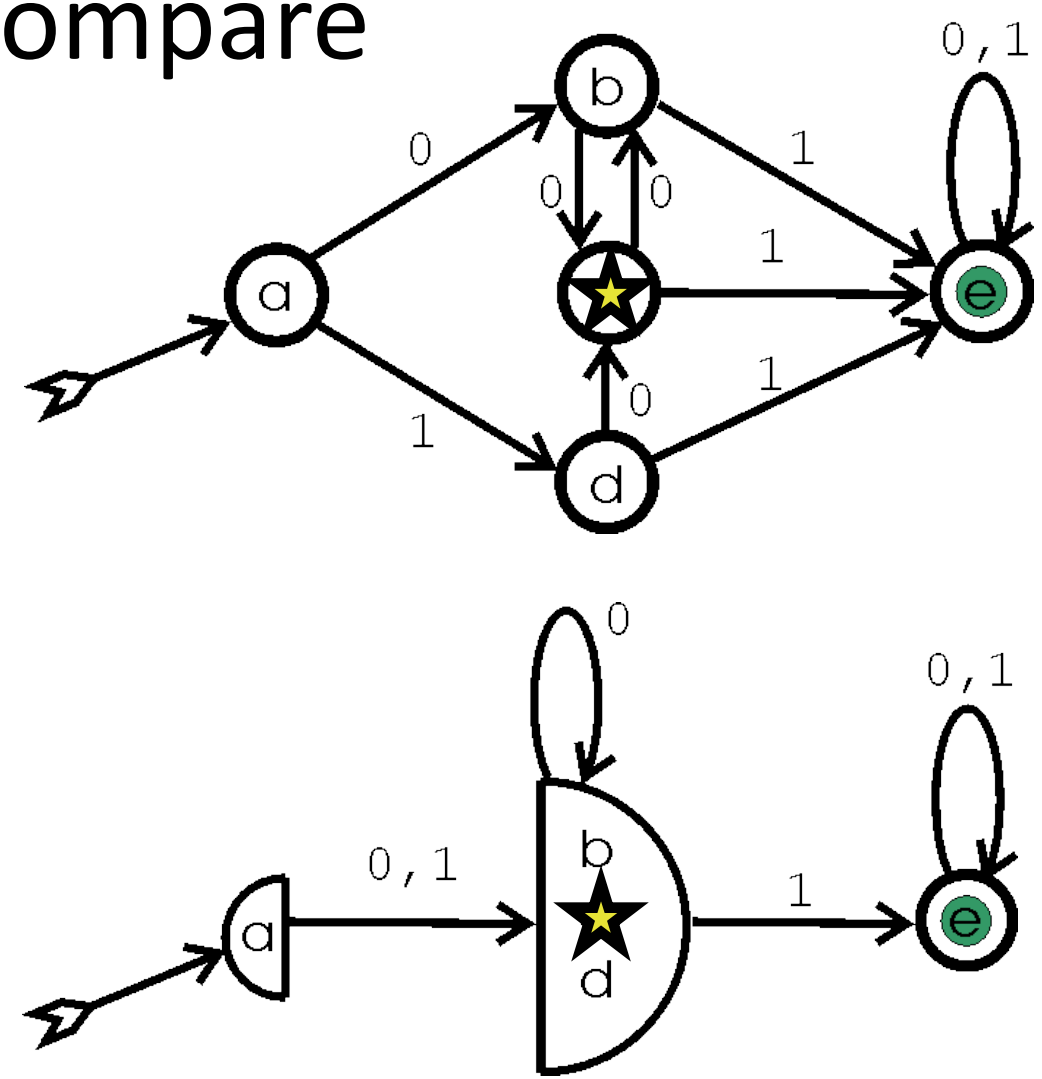
Minimization Example. Compare

100100101



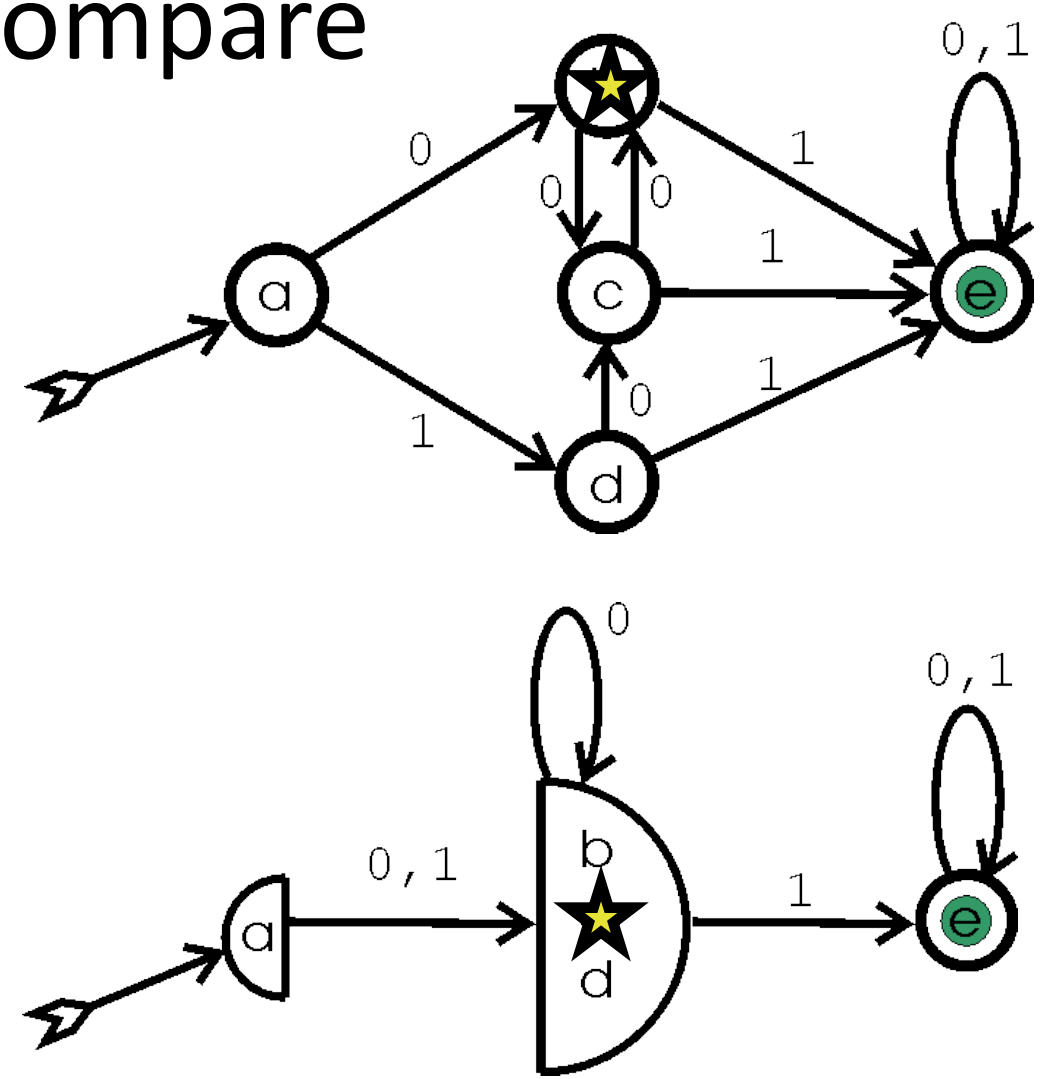
Minimization Example. Compare

100100101



Minimization Example. Compare

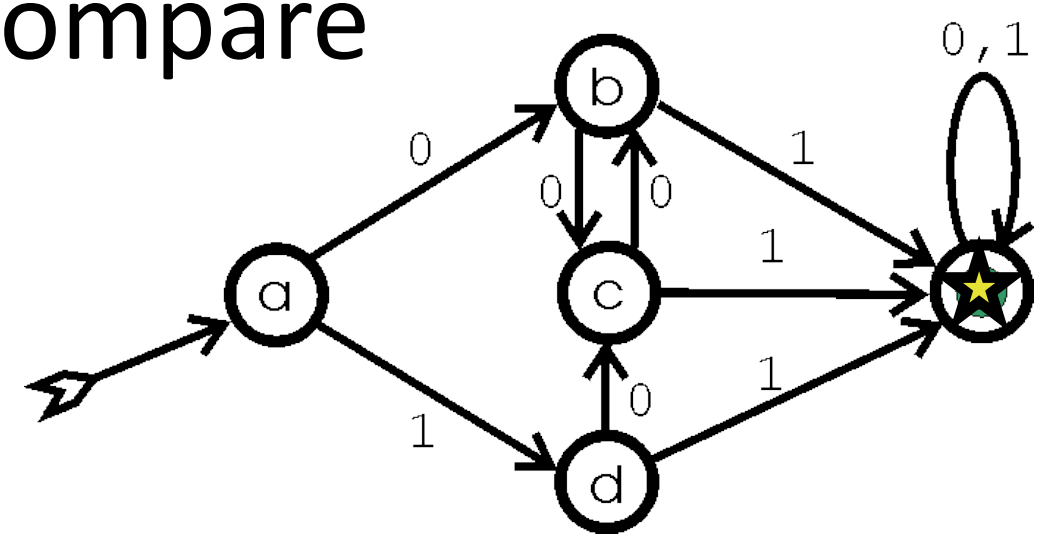
100100101

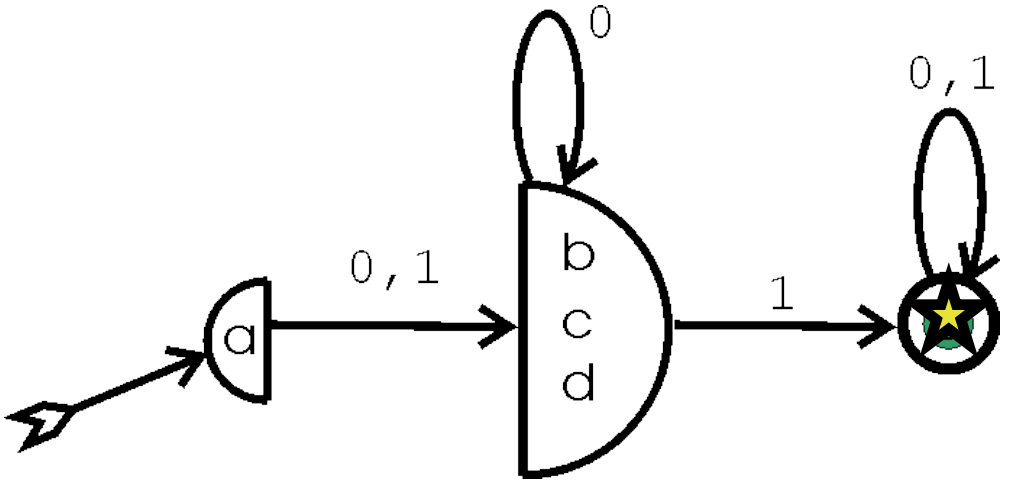


Minimization Example.

Compare

100100101

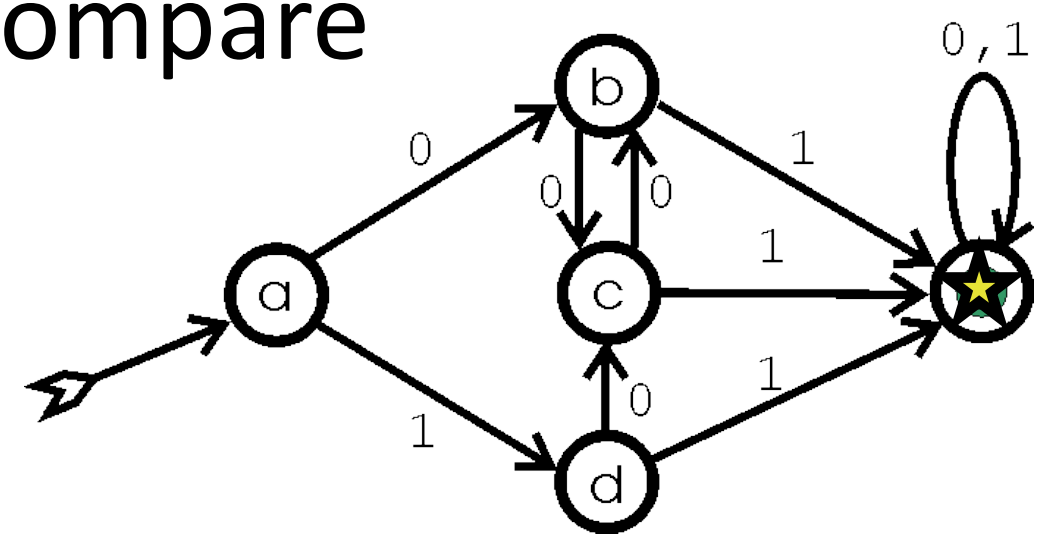


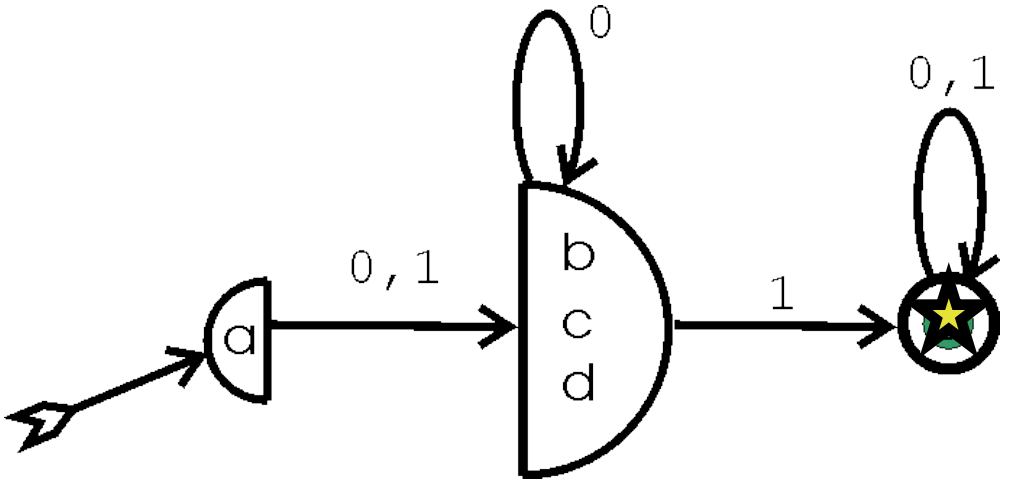


Minimization Example.

Compare

100100101

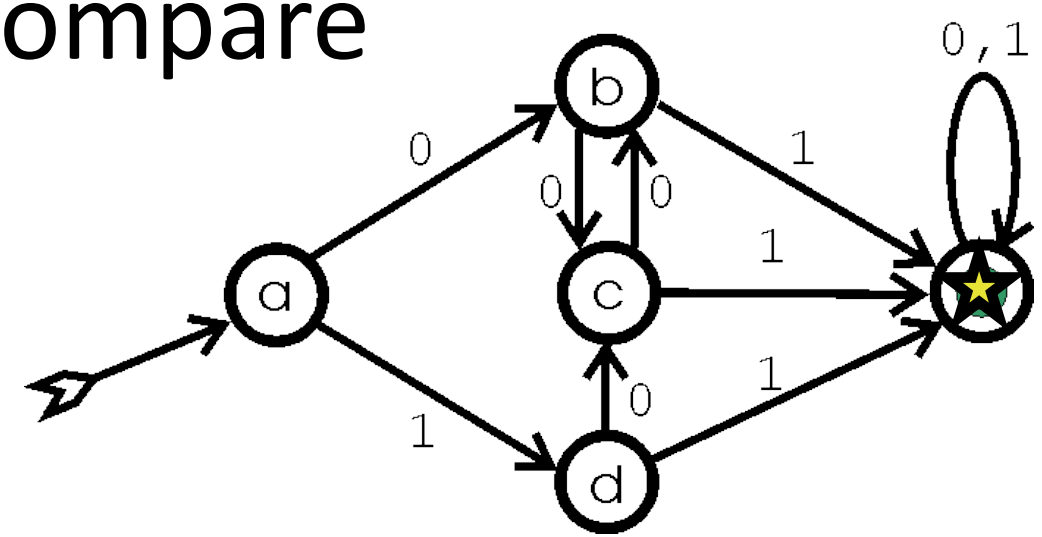


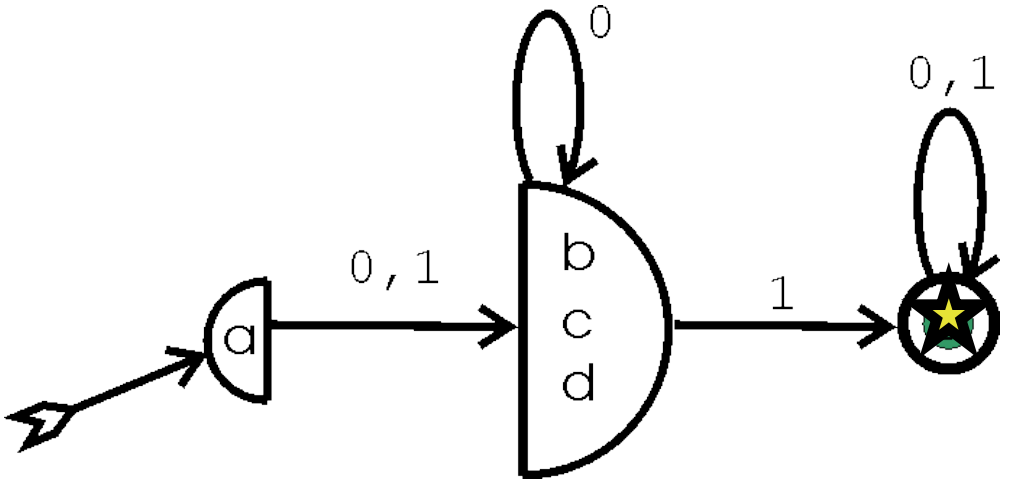


Minimization Example.

Compare

100100101

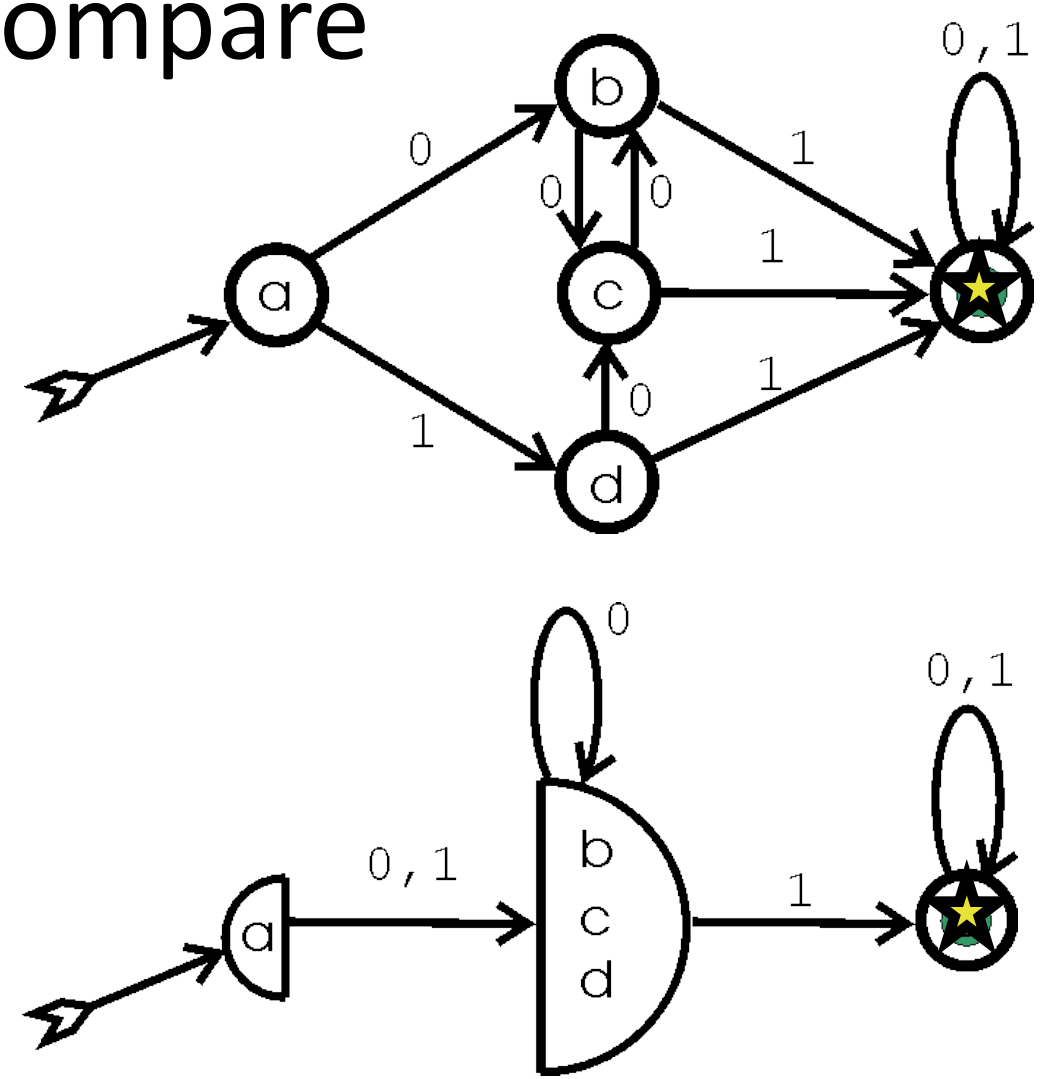




Minimization Example.

Compare

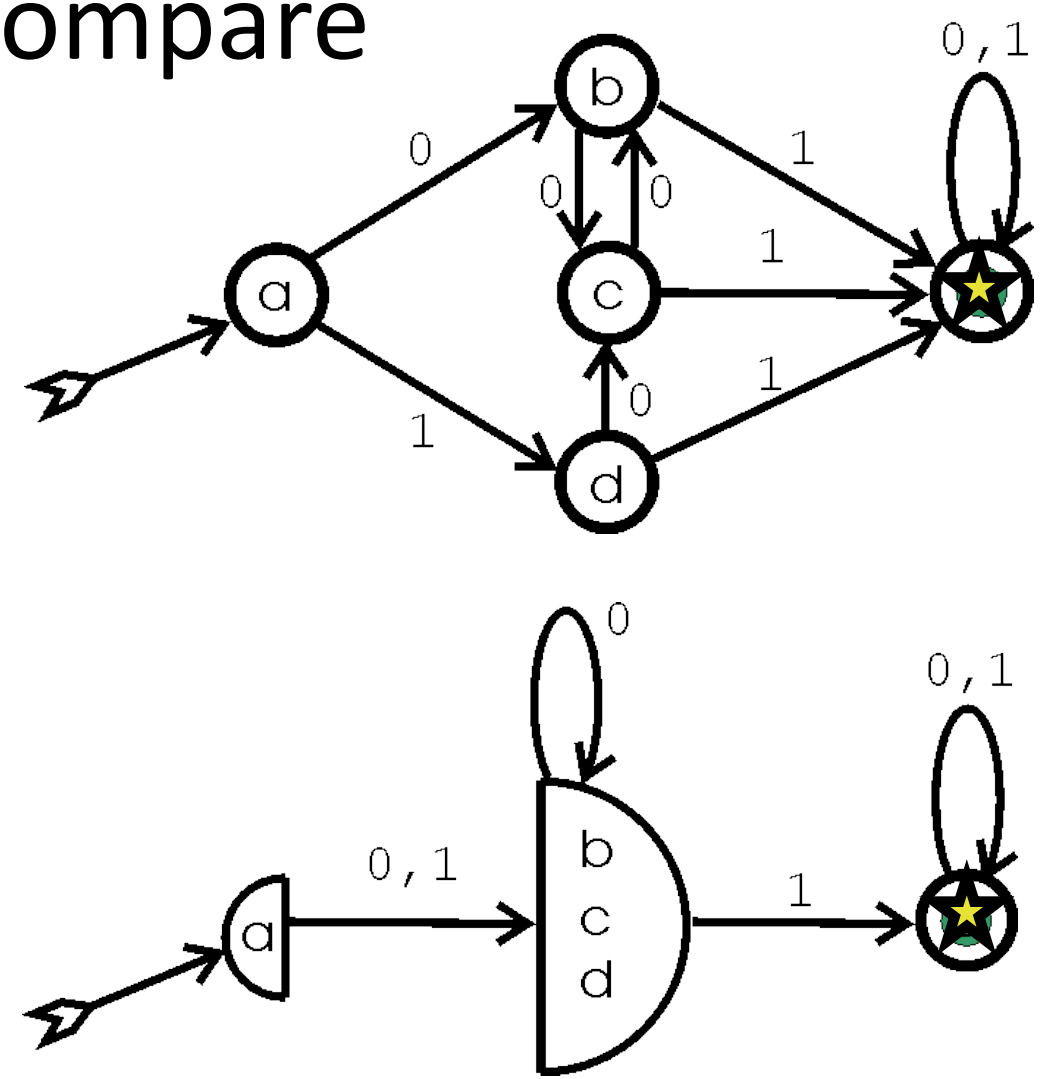
100100101



Minimization Example.

Compare

100100101

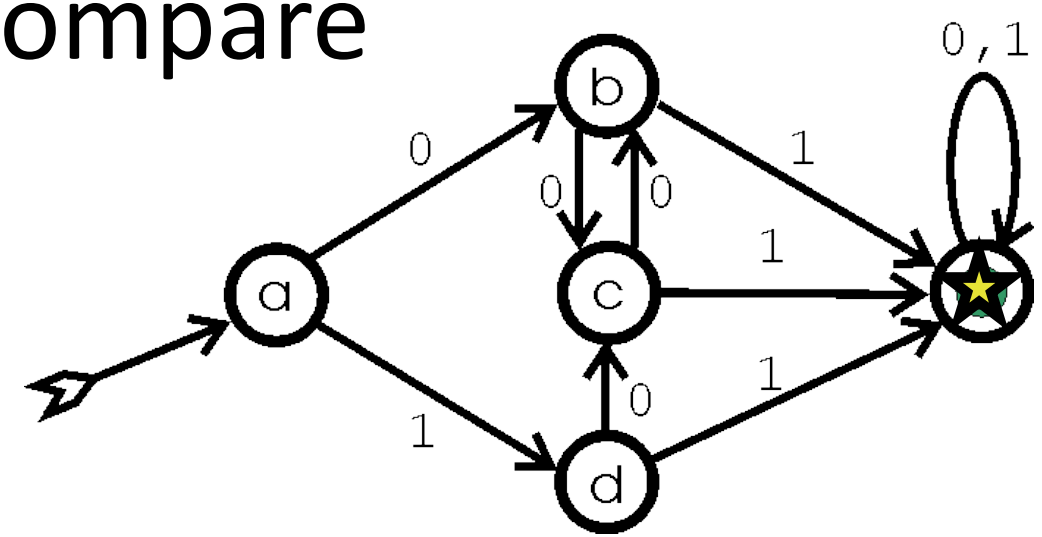


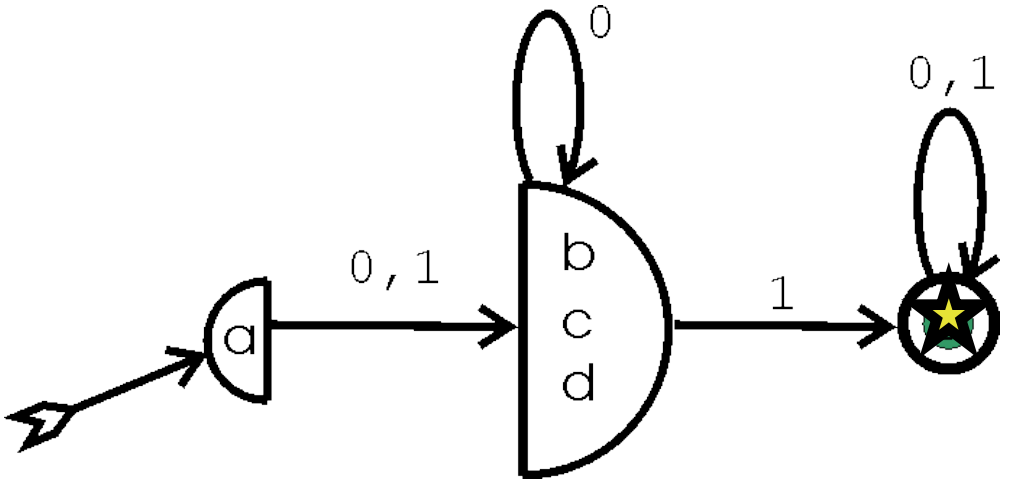
Minimization Example.

Compare

100100101

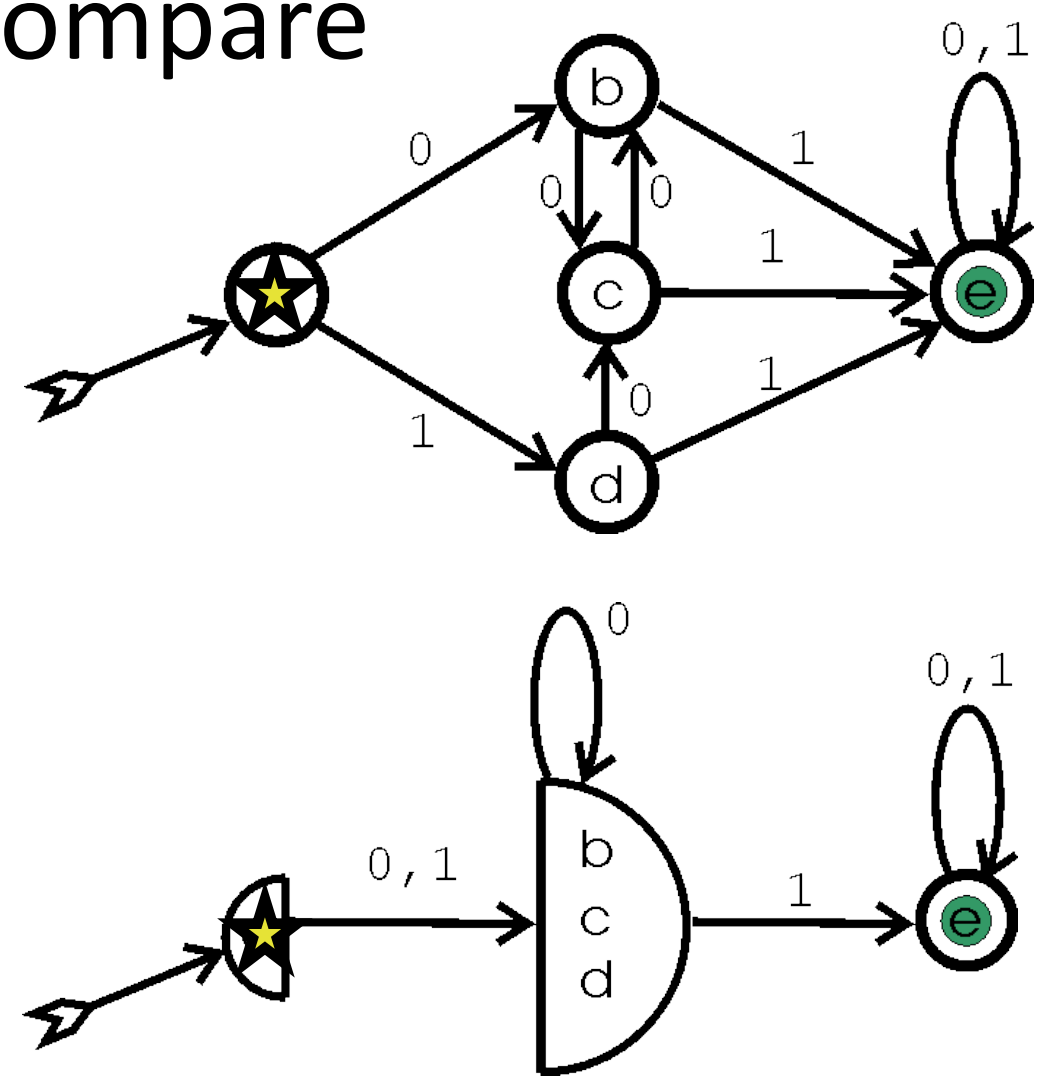
ACCEPTED.





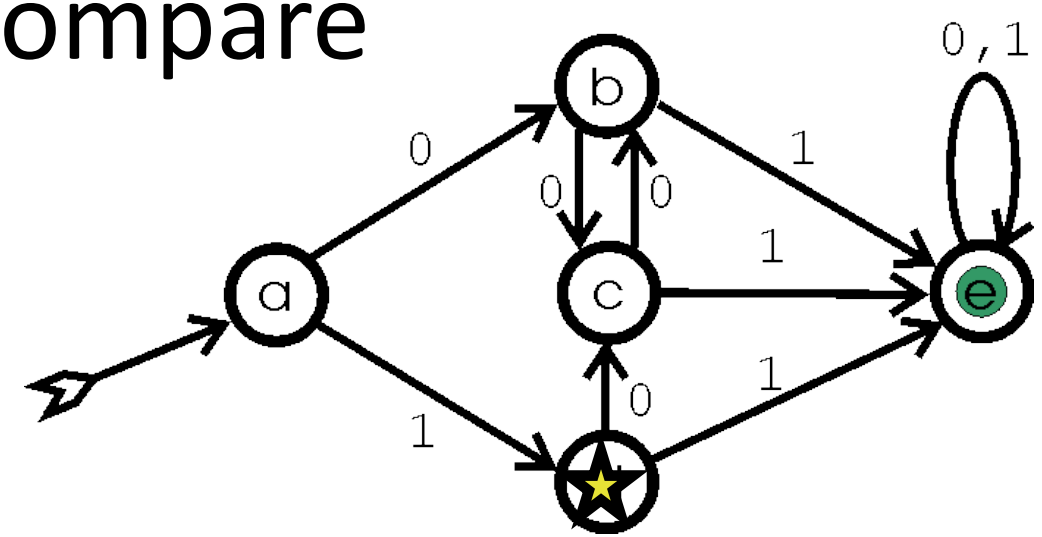
Minimization Example. Compare

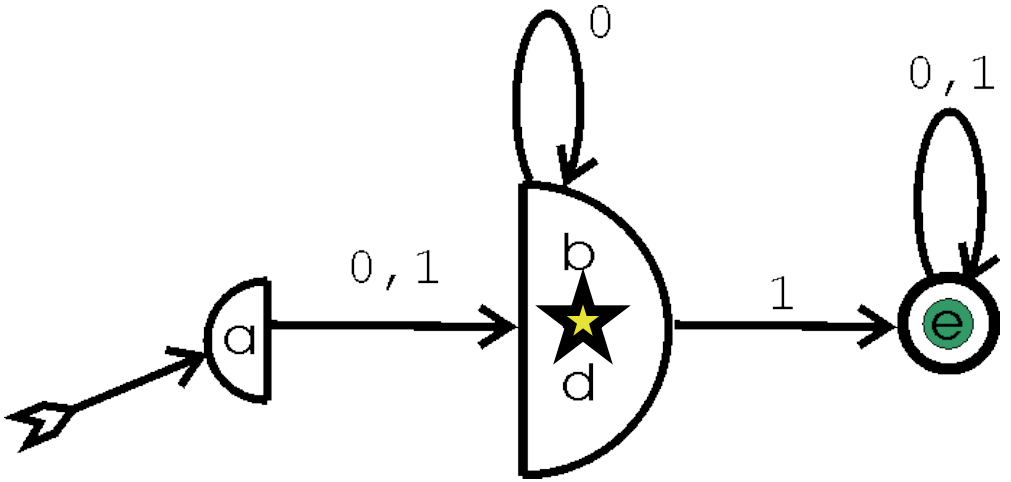
10000



Minimization Example. Compare

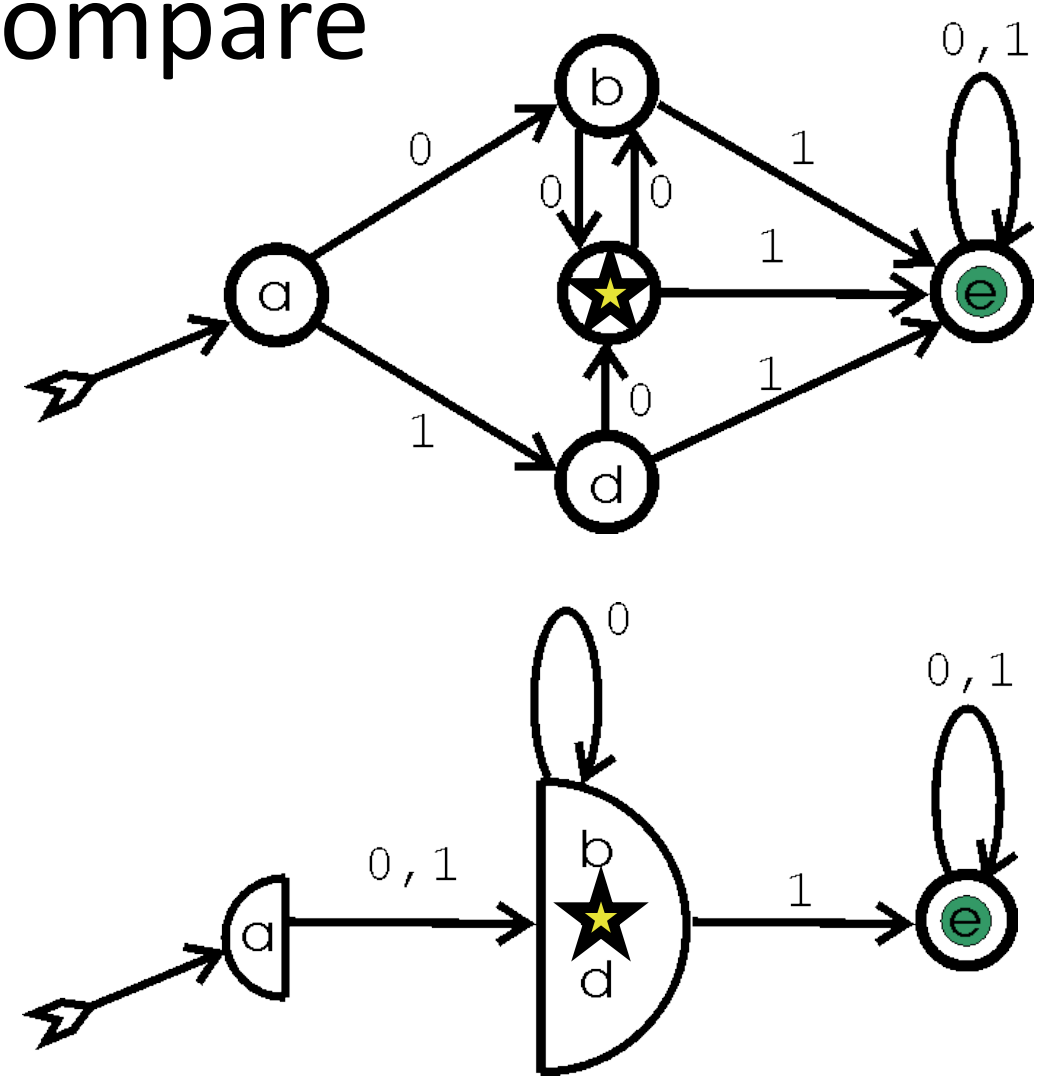
10000





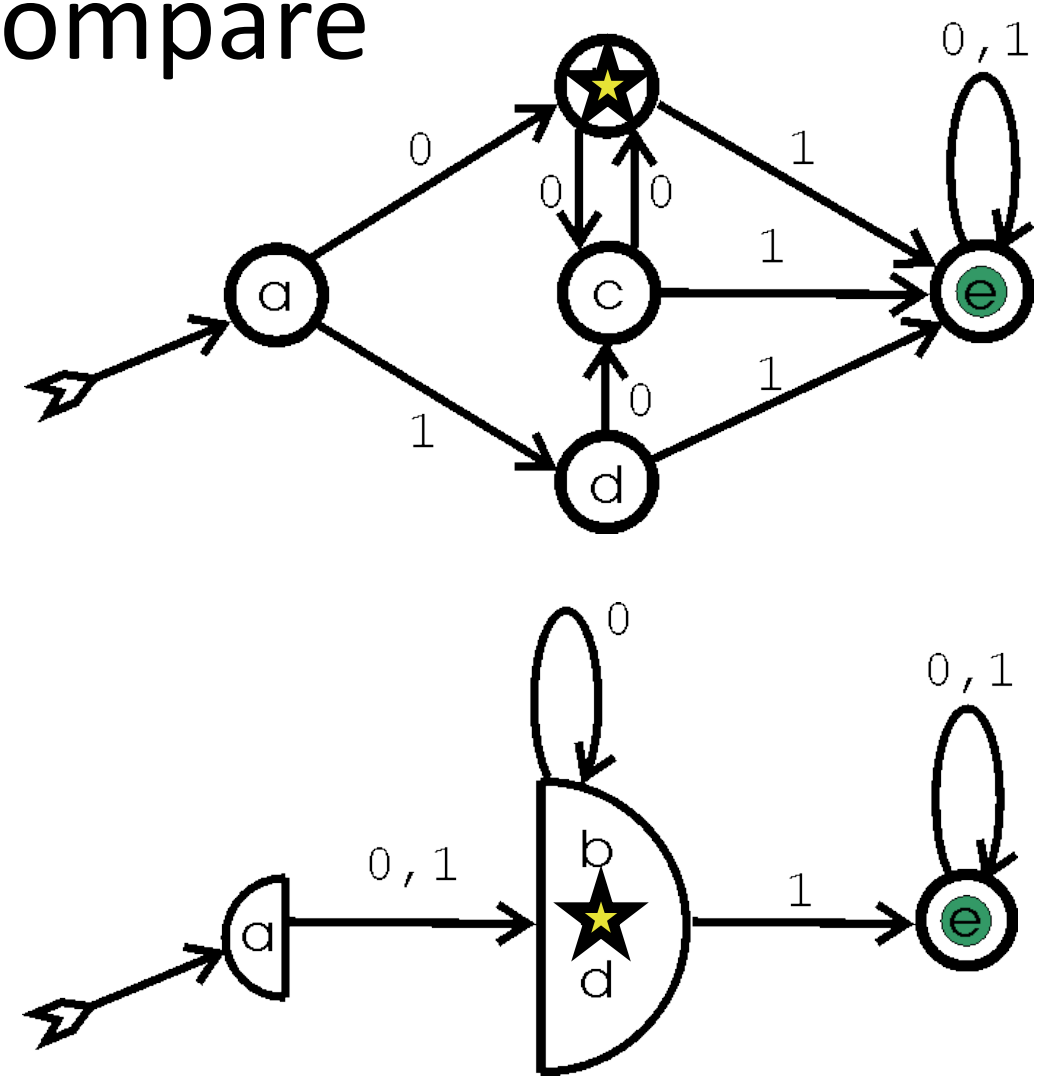
Minimization Example. Compare

10000



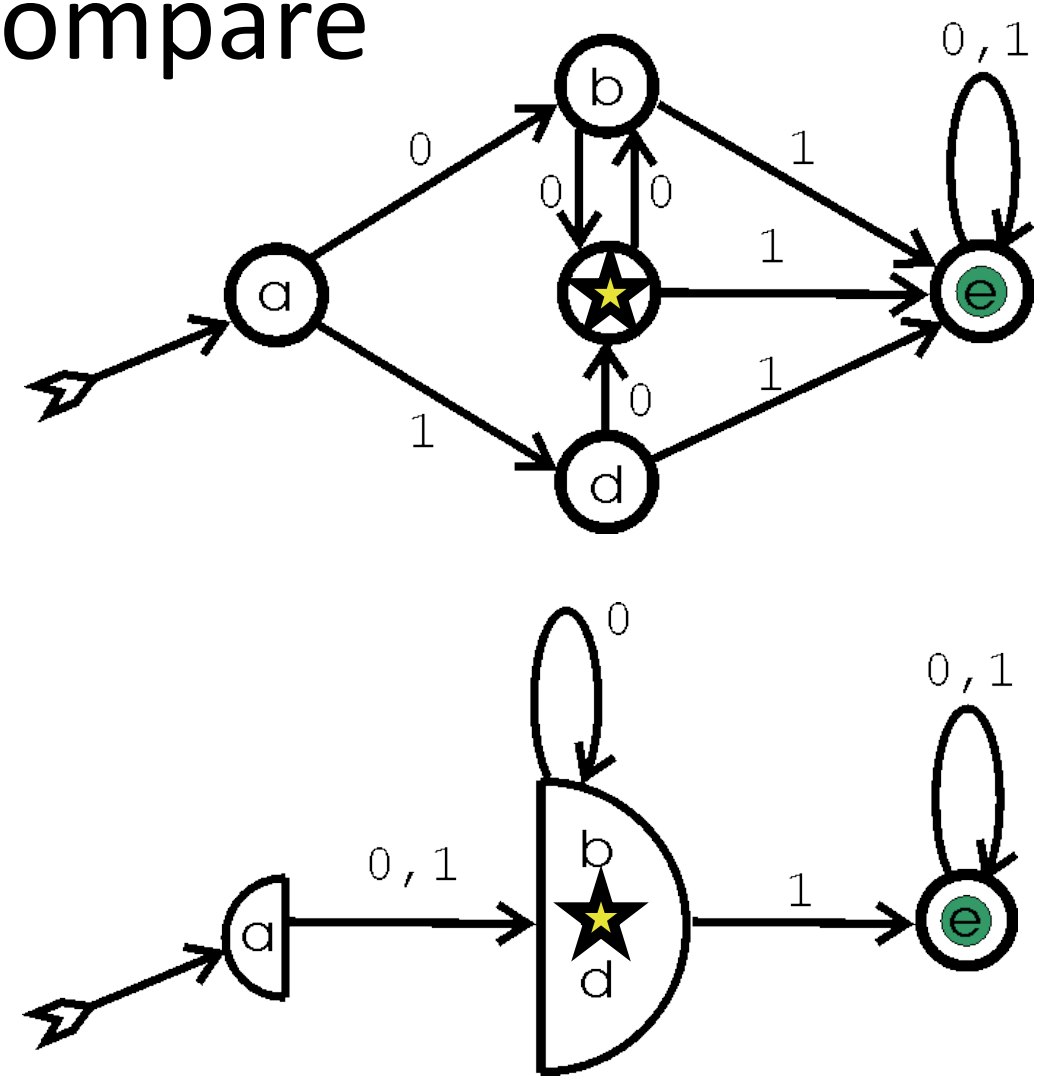
Minimization Example. Compare

10000



Minimization Example. Compare

10000

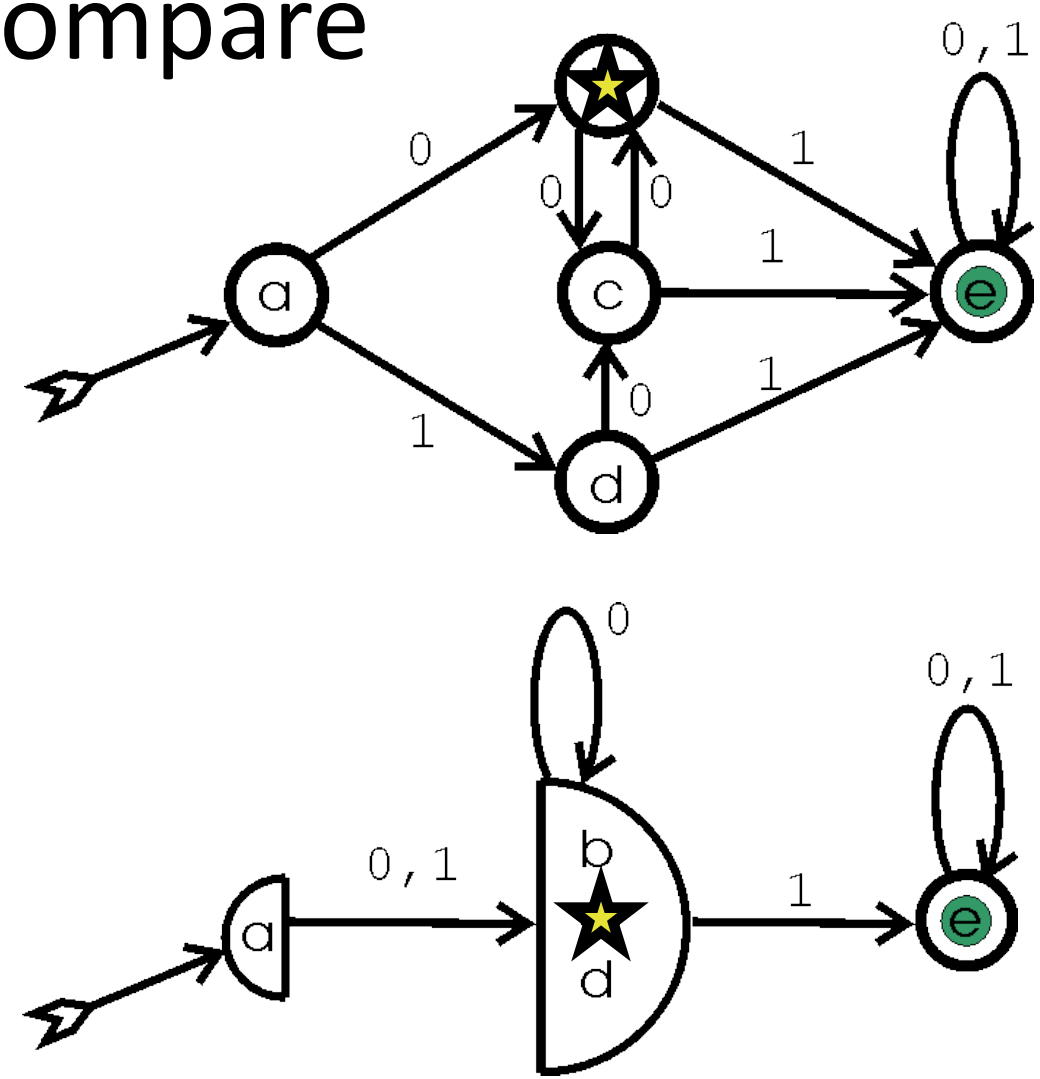


Minimization Example.

Compare

10000

REJECT.



Proof of Minimal Automaton

Previous algorithm guaranteed to produce an irreducible FA. Why should that FA be the smallest possible FA for its accepted language?

Analogous question in calculus: Why should a local minimum be a global minimum? *Usually* not the case!

Proof of Minimality

THM (Myhill-Nerode): The minimization algorithm produces the smallest possible automaton for its accepted language.

Proof. Show that any irreducible automaton is the smallest for its accepted language L:

We say that two strings $u,v \in \Sigma^*$ are ***indistinguishable*** if for all suffixes x, ux is in L exactly when vx is.

Proof of Minimality

Consequently, the number of states in any DFA for *L* must be as great as the number of mutually distinguishable strings for *L*.

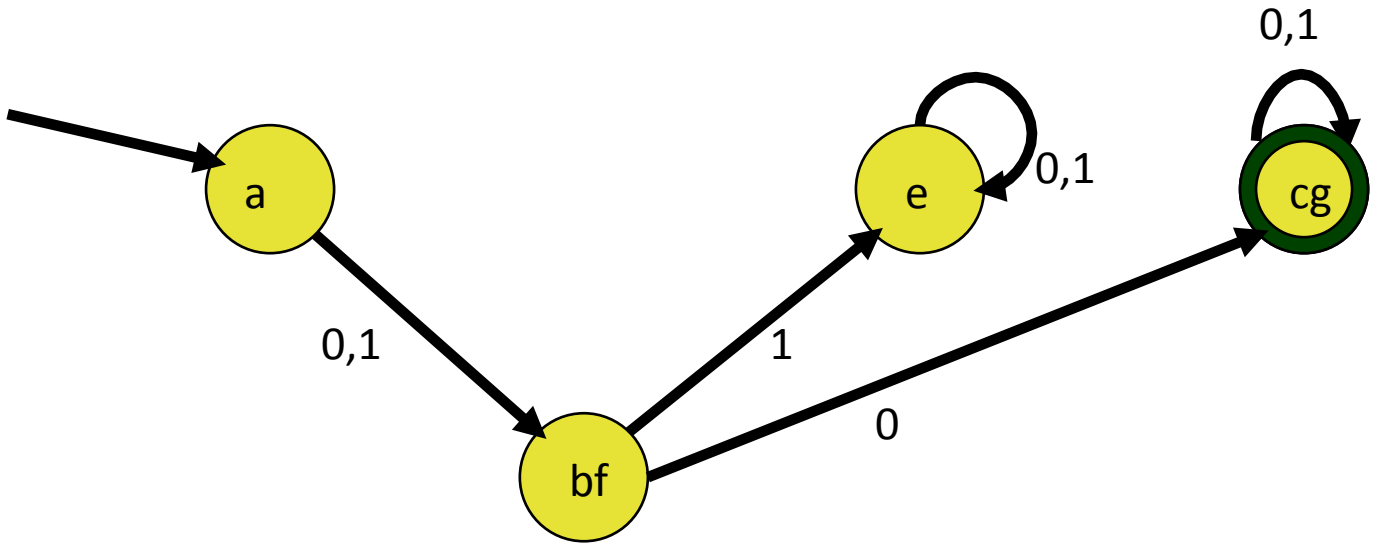
But an irreducible DFA has the property that every state gives rise to another mutually distinguishable string!

Therefore, any other DFA must have at least as many states as the irreducible DFA

Let's see how the proof works on a previous example:

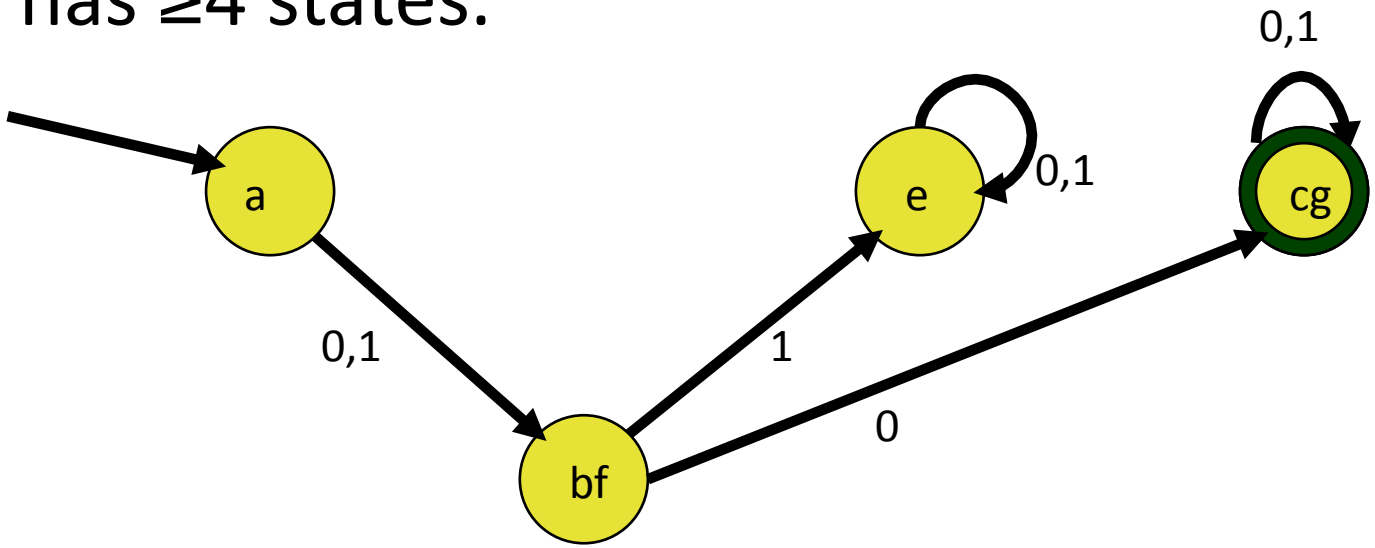
Proof of Minimal Automaton. Example

- 00 and 000 are indistinguishable
- 10 and 1010 are indistinguishable
- 11 and 01 are indistinguishable



Proof of Minimal Automaton. Example

The "spanning tree of strings" {ε,0,01,00} is a mutually distinguishable set (otherwise redundancy would occur and hence DFA would be reducible). Any other DFA for *L* has ≥4 states.



Equivalence Classes

- Finite automata induce finitely many equivalence classes on strings from Σ^*
  (Myhill Nerode: Finite Automata are of finite index)
  - $x \equiv_M y$ iff $\delta(s,x) = \delta(s,y) = q$ for some $q \in Q$, the set of states.
  - There are only $|Q|$ states, hence Σ^* has maximally $|Q|$ $\equiv_M$-classes.
- Our example automaton M has 4 equivalence classes (named with representatives $\{\varepsilon,0,01,00\}$)