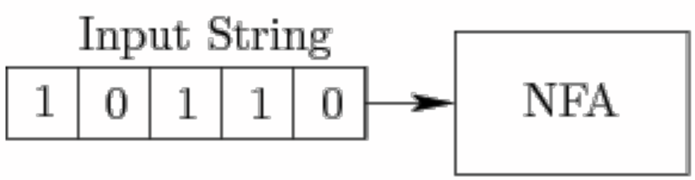
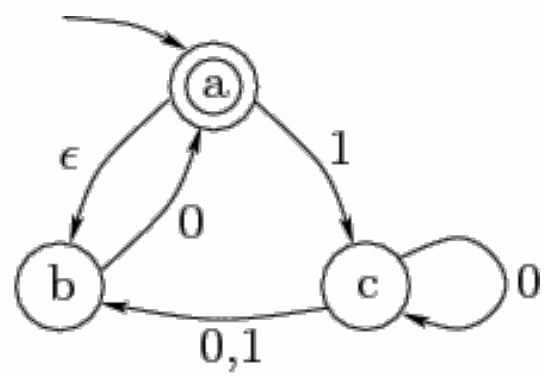
CS 321

Theory of Computation

Part I: Finite Automata and Regular Languages



(a)



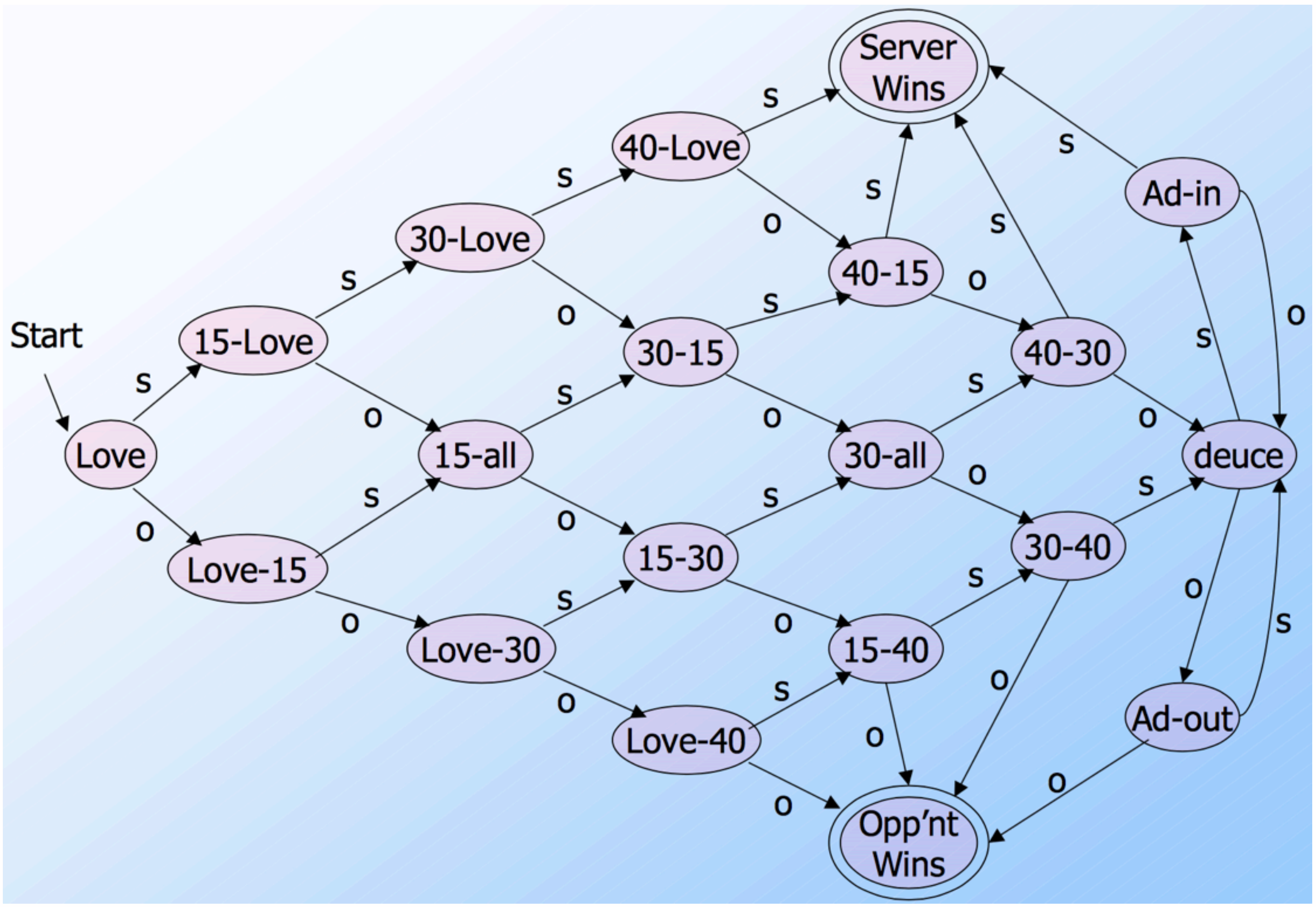
(b)

Instructor: Liang Huang

(some slides courtesy of J. Ullman and H. Potter)

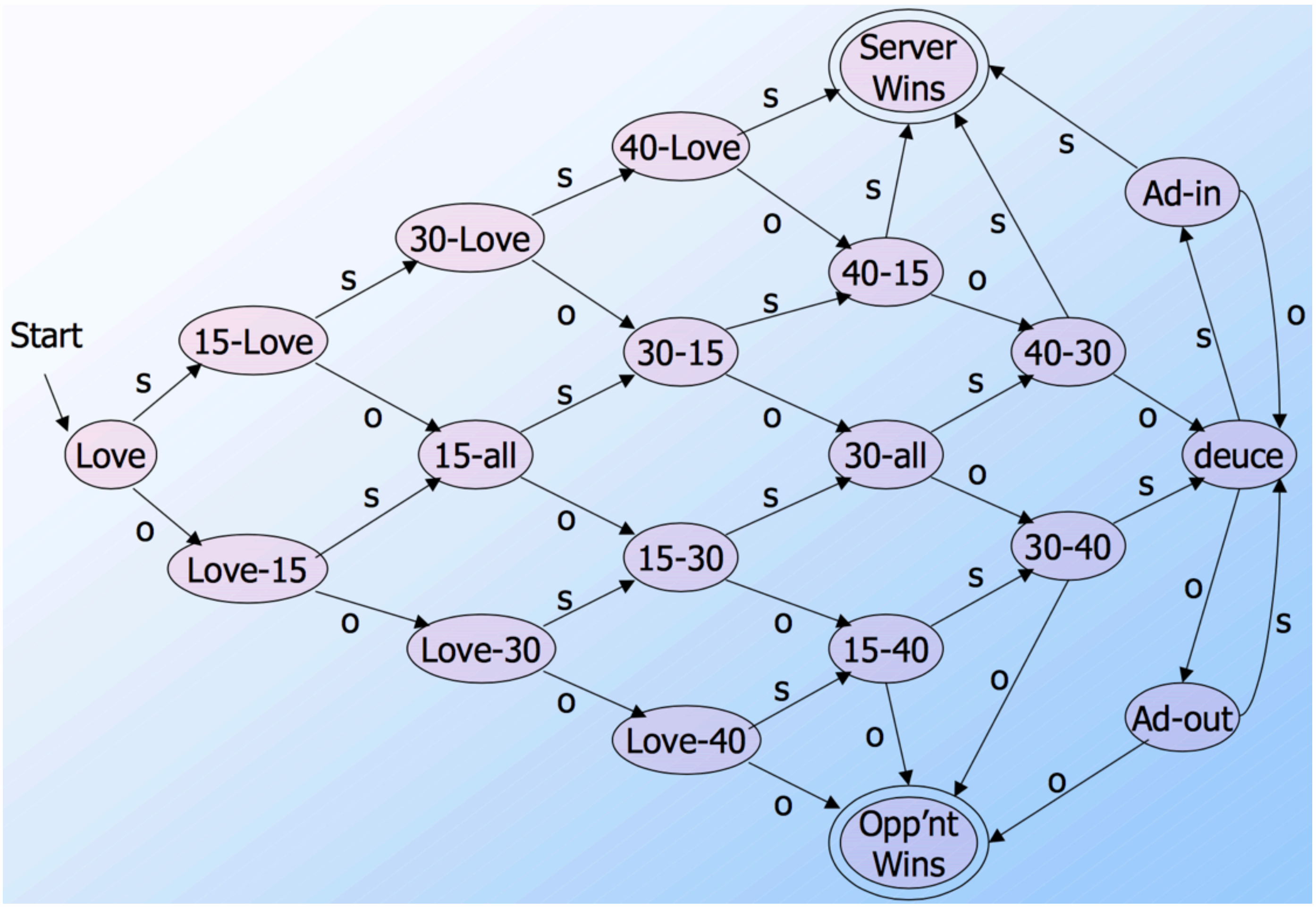
Finite Automata Example

- in Tennis, match = 3-5 sets; set = 6 or more games
- in one game, one person serves throughout
  - to win, you must win by at least 2 points



Finite Automata Example

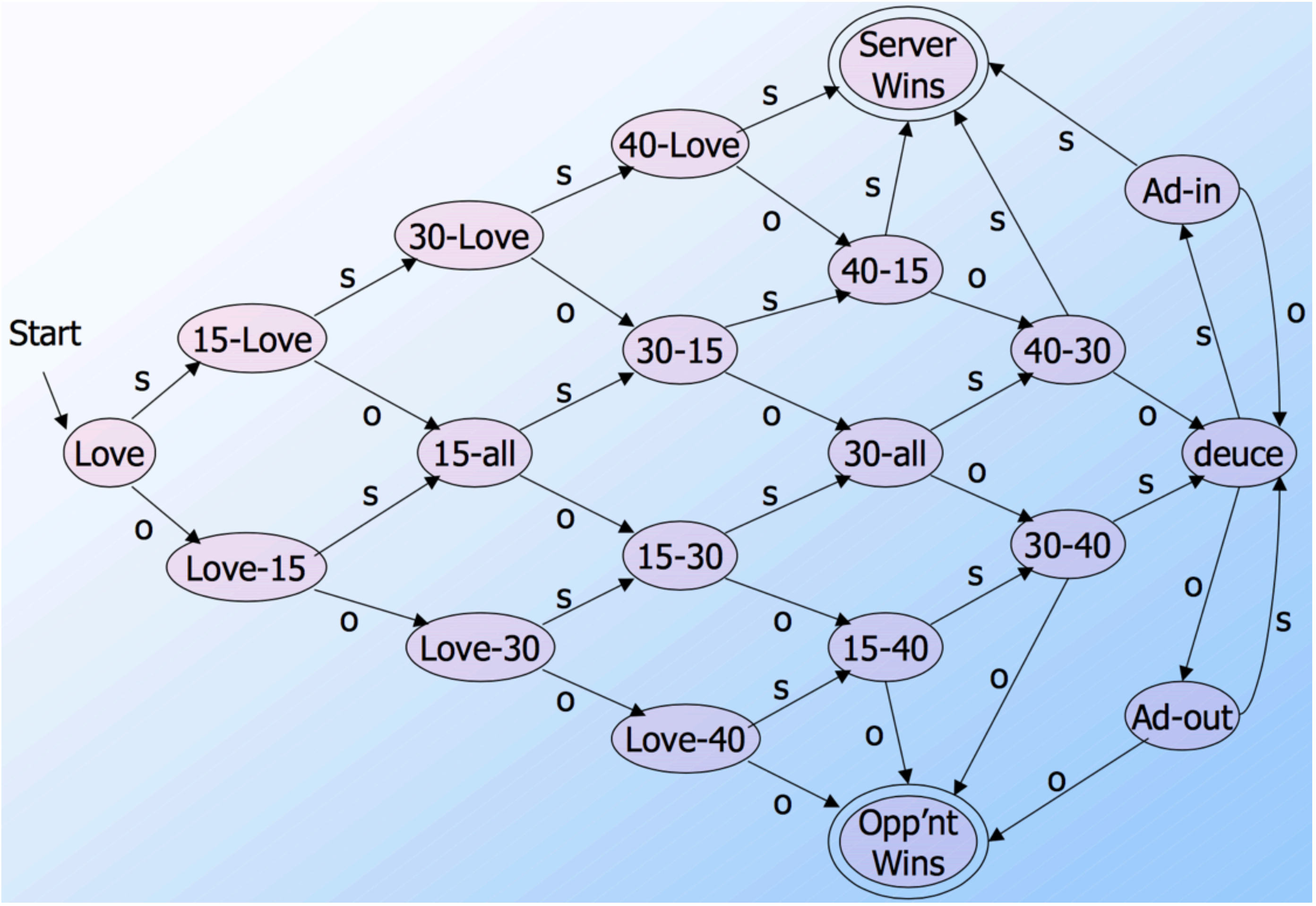
- states: encode status info. finite: only knows current state
  - state changes in response to inputs (s, o in this example)
  - transitions: rules for state change. *at state x, on input a, goto state y*



trap state not shown

Acceptance/Rejection of Inputs

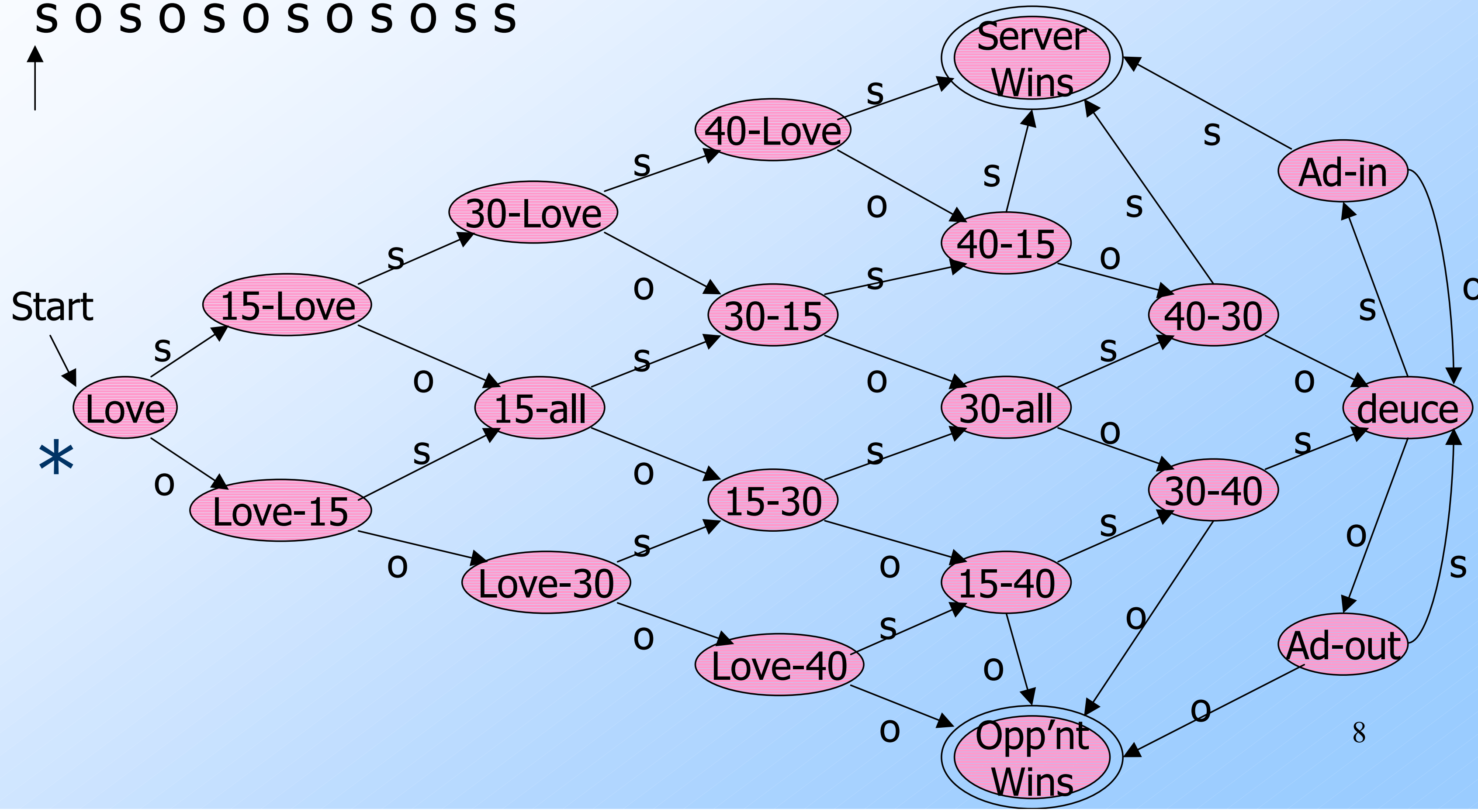
- given an input string, start at the start state and follow transitions
- input is accepted if you wind up in a final (accepting) state after all inputs have been read; otherwise rejected



trap state not shown

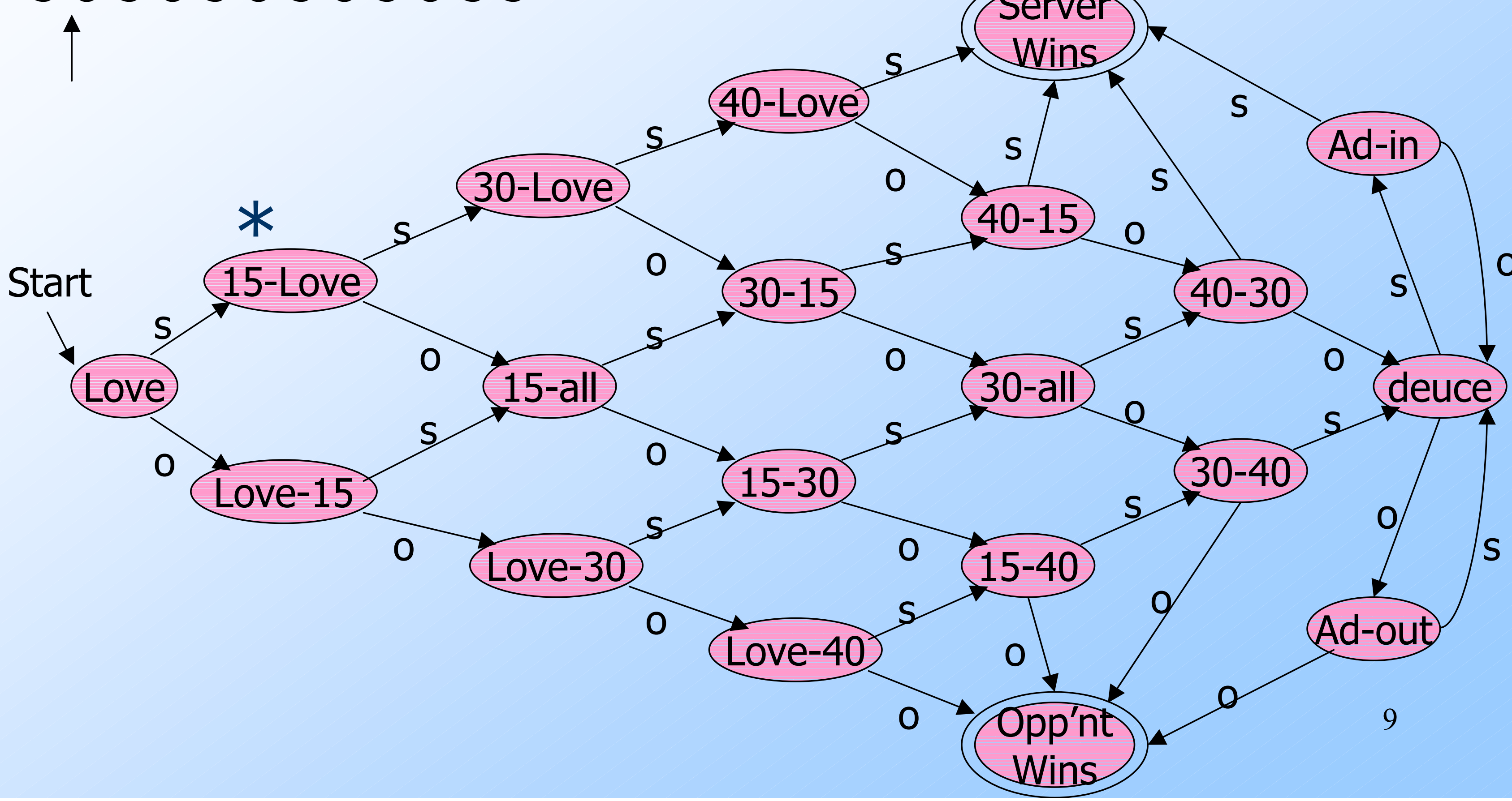
Example: Processing a String

s o s o s o s o s o s o s s



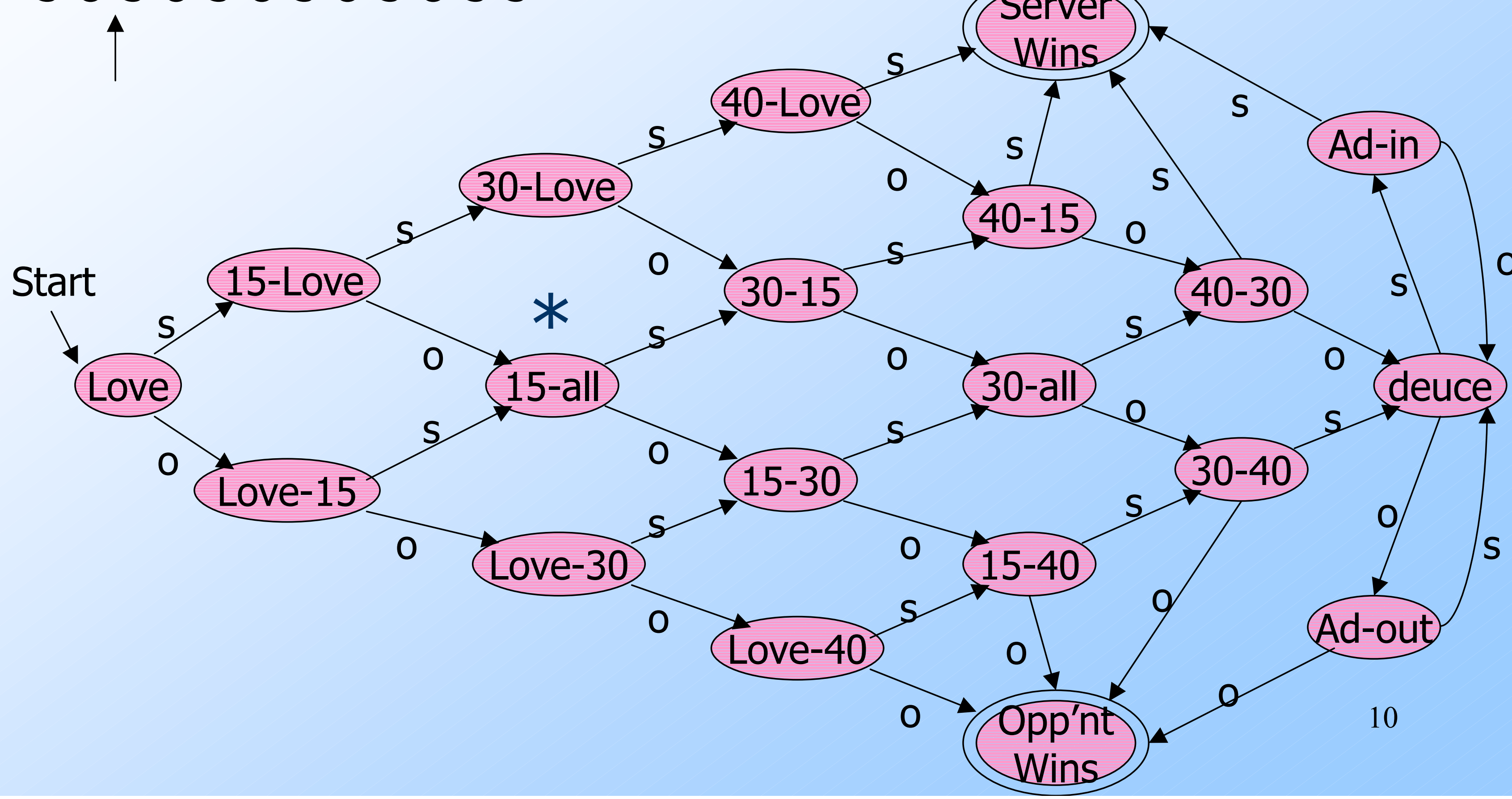
Example: Processing a String

s o s o s o s o s o s o s s



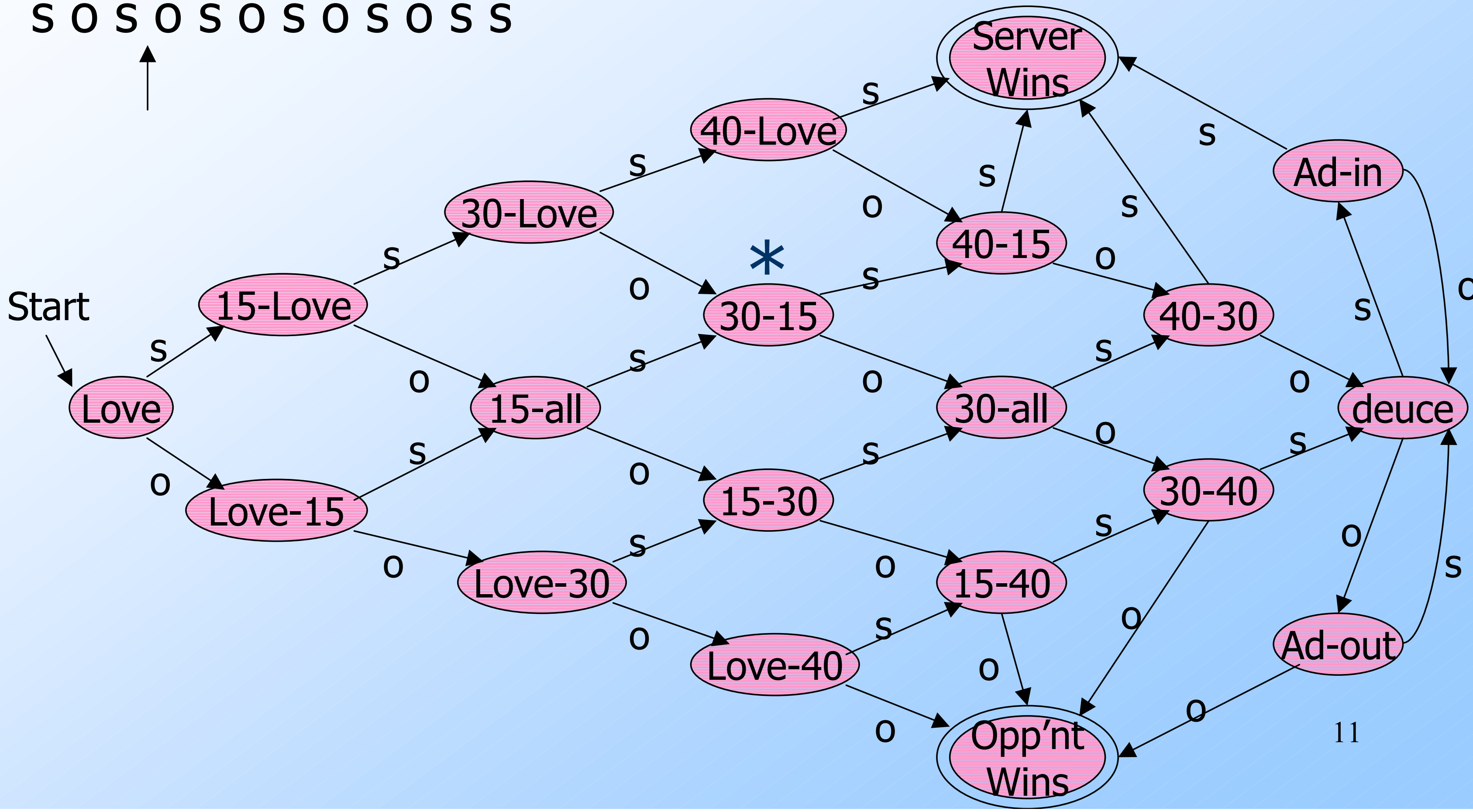
Example: Processing a String

s o s o s o s o s o s o s s

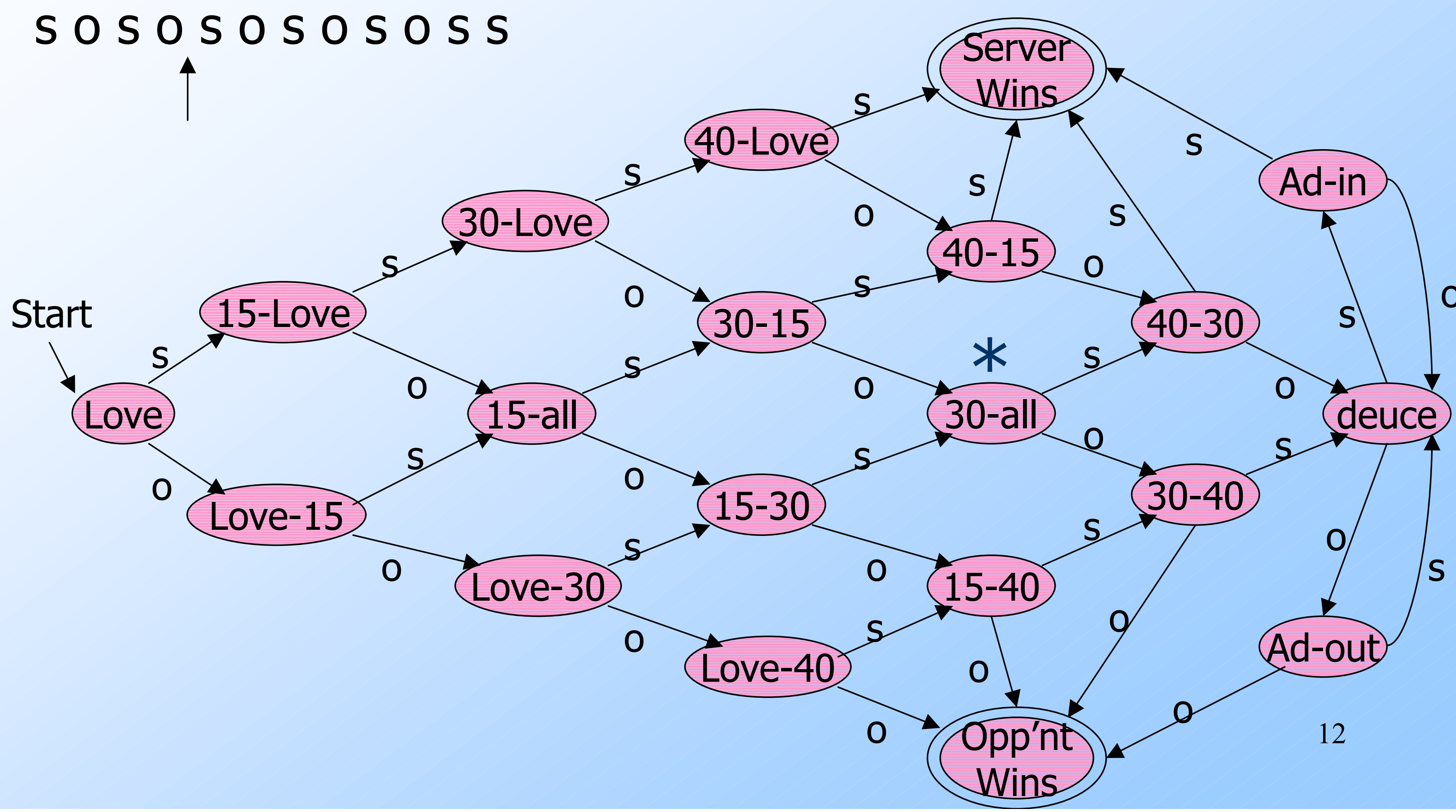


Example: Processing a String

s o s o s o s o s o s o s s

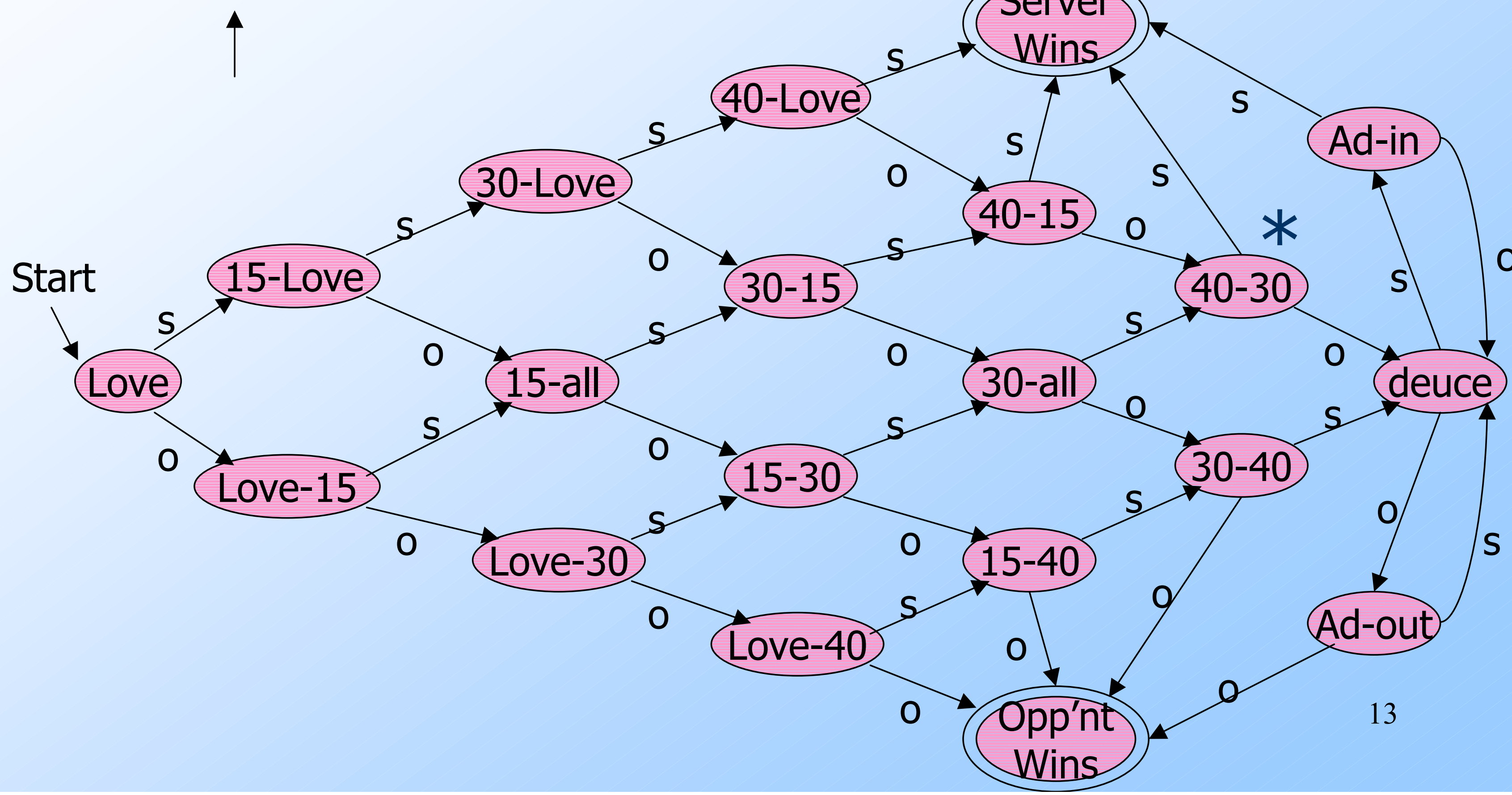


Example: Processing a String



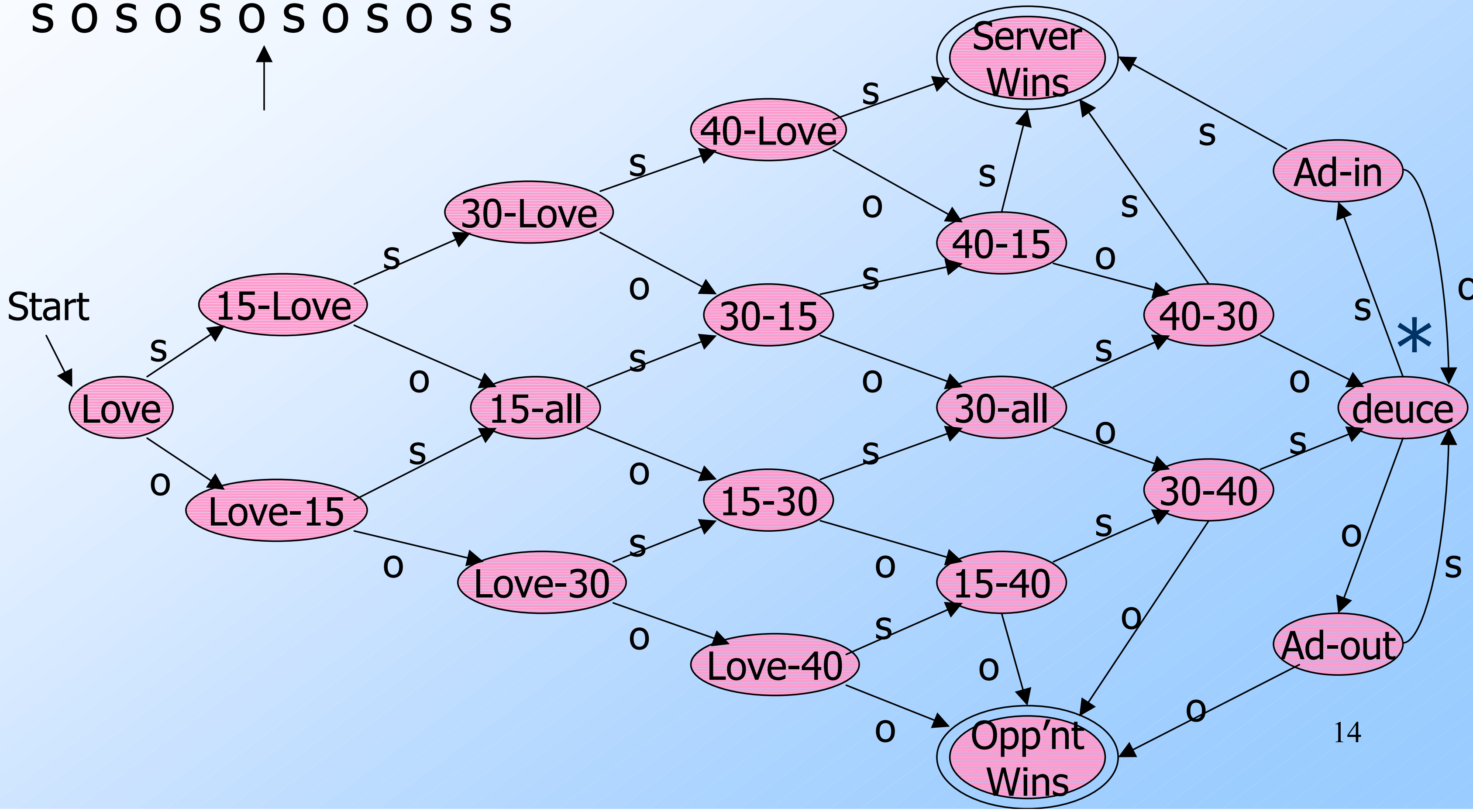
Example: Processing a String

s o s o s o s o s o s s



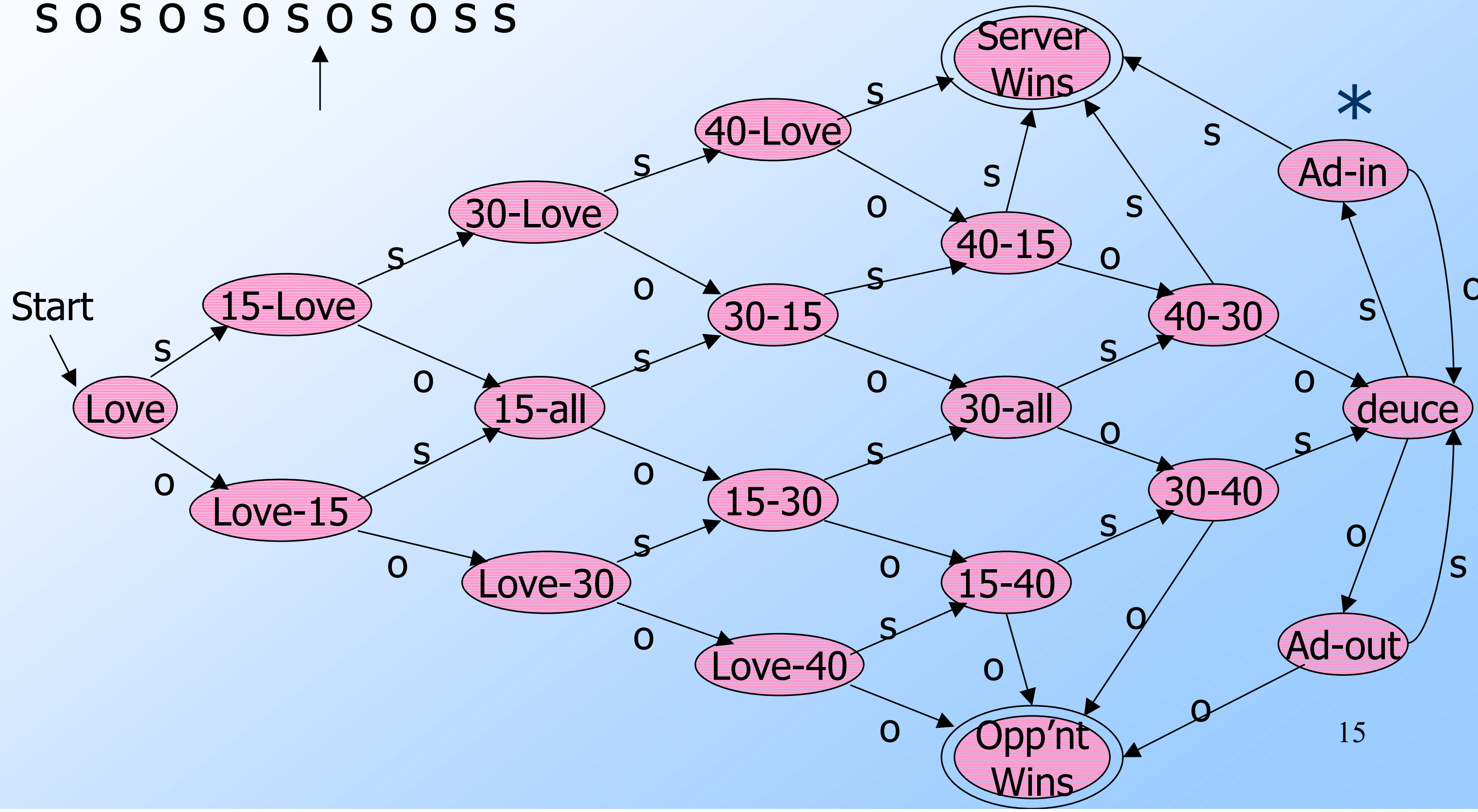
Example: Processing a String

s o s o s o s o s o s s



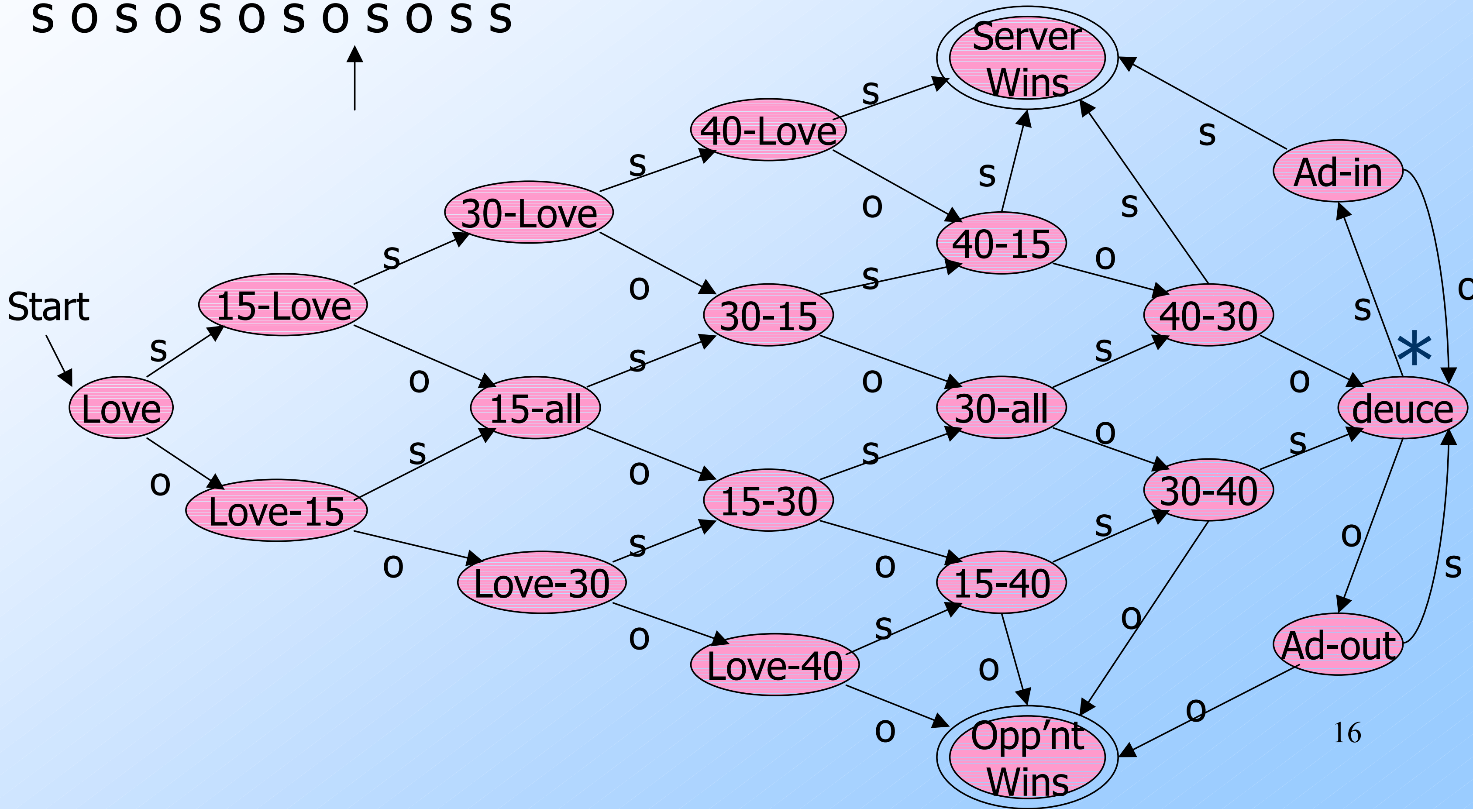
Example: Processing a String

s o s o s o s o s o s s



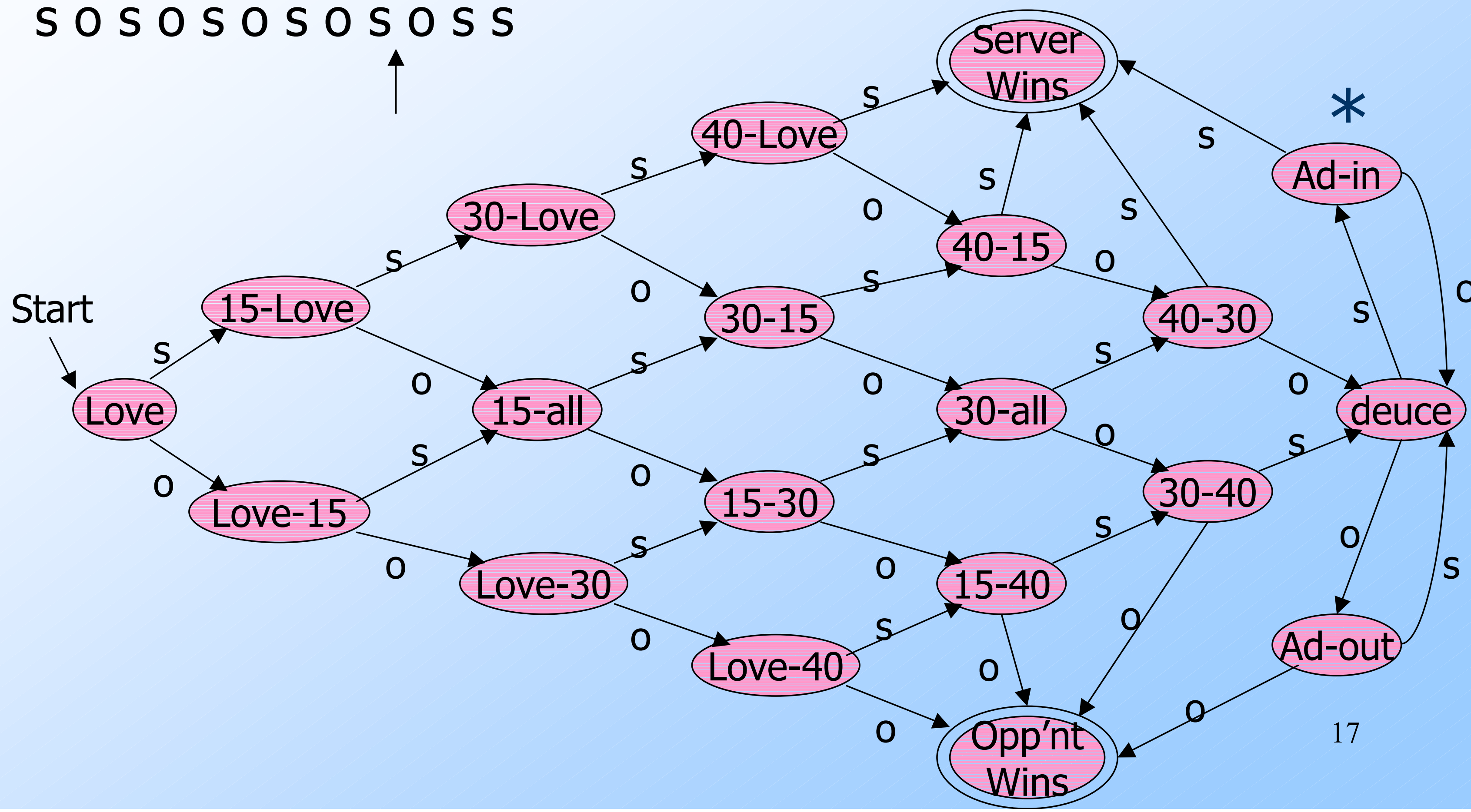
Example: Processing a String

s o s o s o s o s o s s



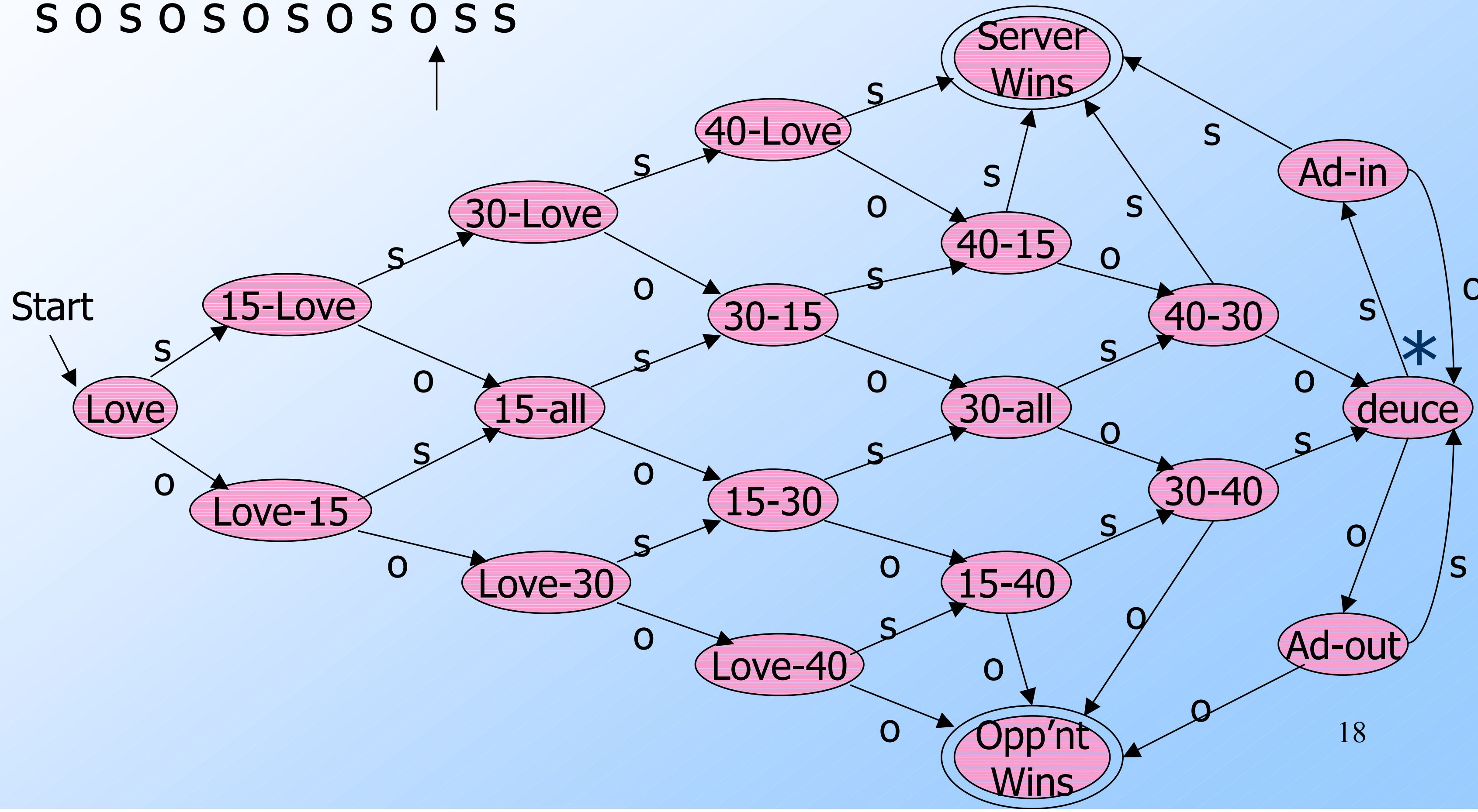
Example: Processing a String

s o s o s o s o s o s s



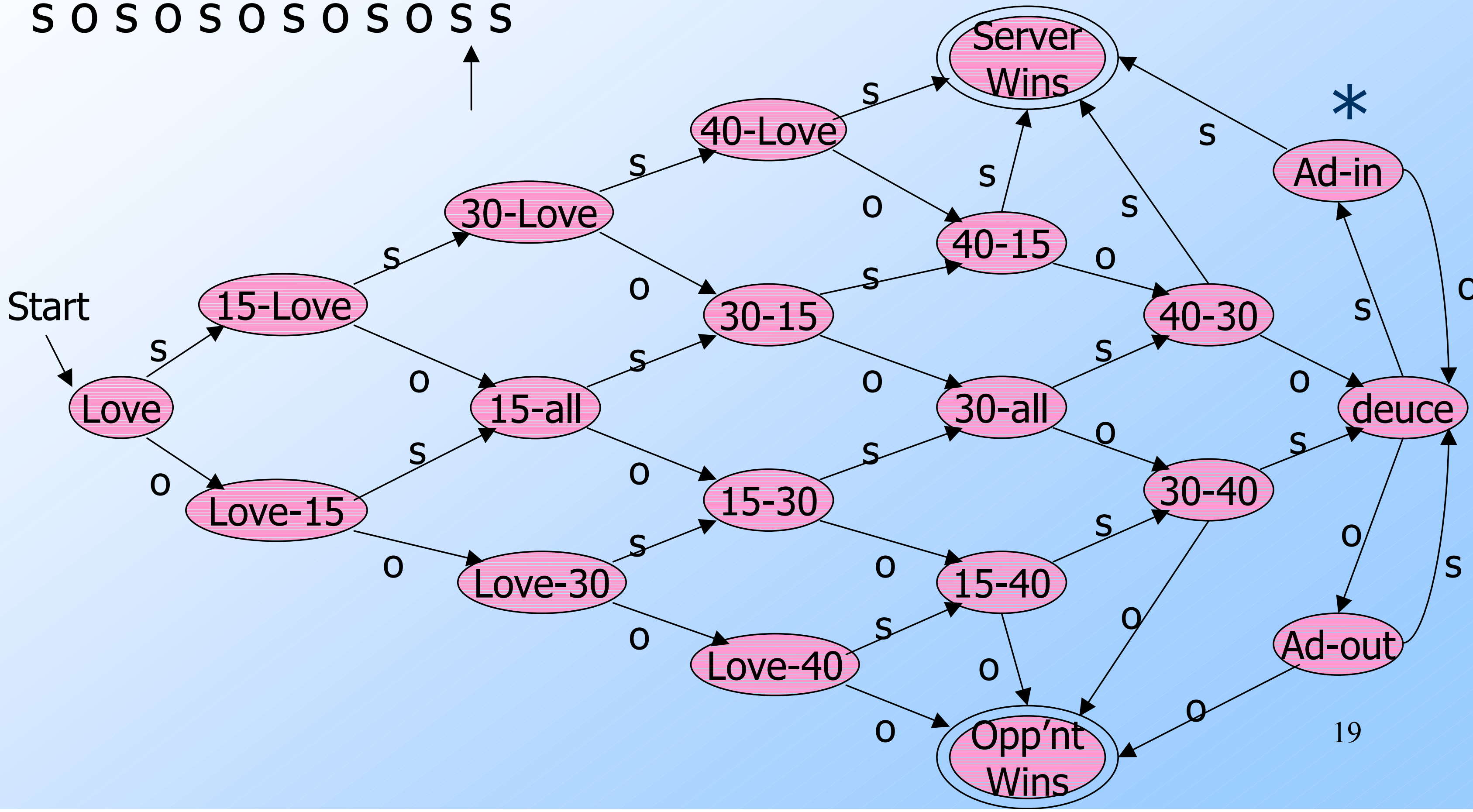
Example: Processing a String

s o s o s o s o s o s o s s

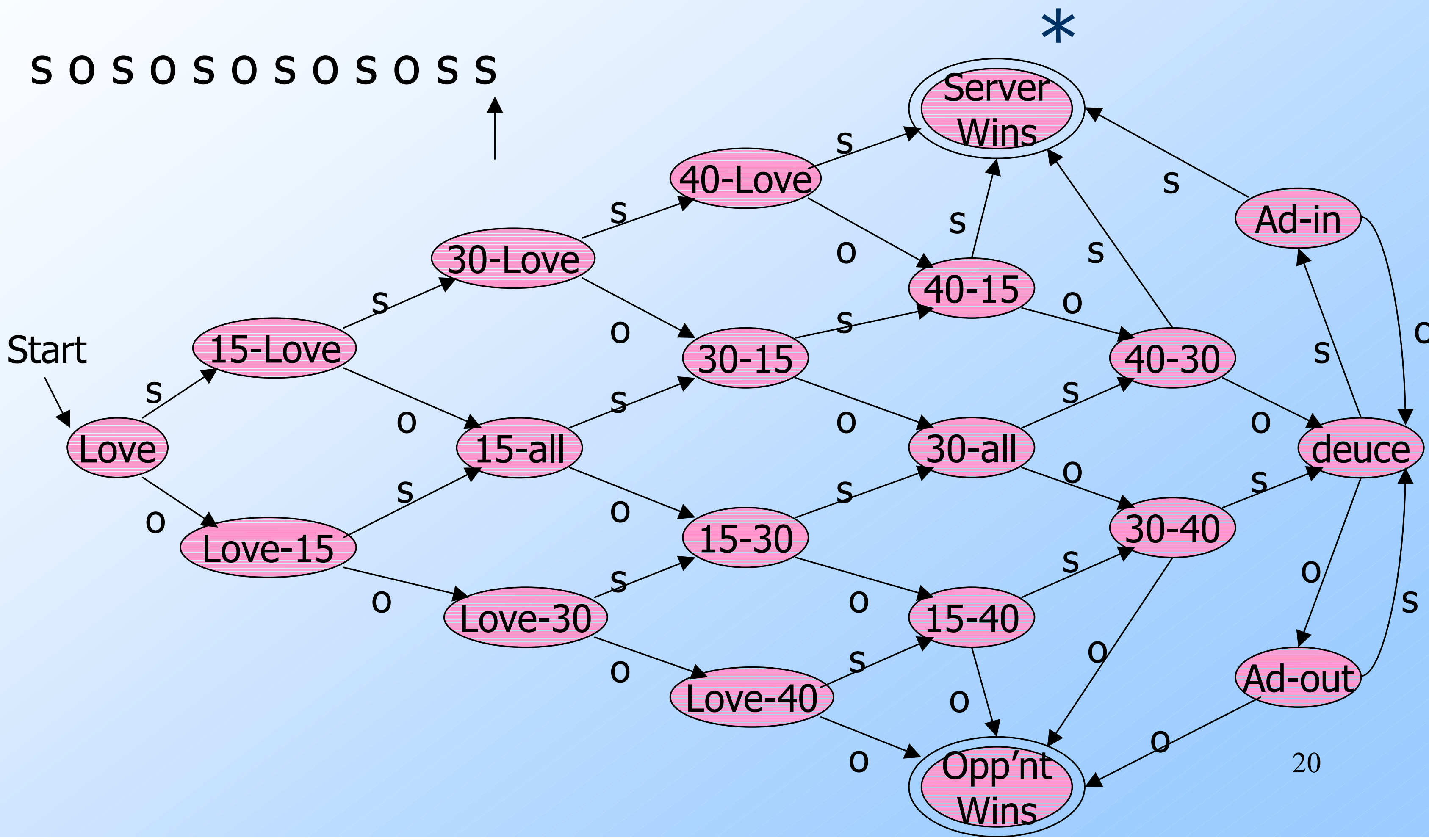


Example: Processing a String

s o s o s o s o s o s s



Example: Processing a String



Language of an Automaton

- the set of strings accepted by an automaton A is the language of A, denoted L(A)
- L(Tennis) = strings of {s, o} that determine the winner

$\boxed{\text{STRINGS}}$

Σ = Alphabet = Set of symbols.

$\Sigma = \{a, b, c, d\}$

(Always a <u>finite</u> set!)

STRING = A <u>finite</u> sequence of symbols.

baccada

ϵ = Empty string = Epsilon (also ε)

Length of a string

w = baccada

$|w| = |\text{baccada}| = 7$

$|\epsilon| = 0$

xy = Concatenation

x^R = Reverse of x

x = bac

y = cada

xy = baccad

x^R = cab

Language

"A language is a set of strings."

$L_1 = \{ab, bc, ac, dd\}$

$L_2 = \{\epsilon, ab, abab, ababab, \dots\}$

The empty string:

ϵ

The empty language:

$\{\} = \emptyset$

The language containing only ϵ:

$L_3 = \{\epsilon\}$

How to describe languages?

Enumeration $\{b, ac, da\}$

Regular Expressions $c(ab)^*(d|c)$

Context-Free Grammars

Set Notation

$\{x \mid x \in \dots\}$

$S \to dTd$

$T \to TT \mid aTb \mid \epsilon$

Construct Finite Automata for ...

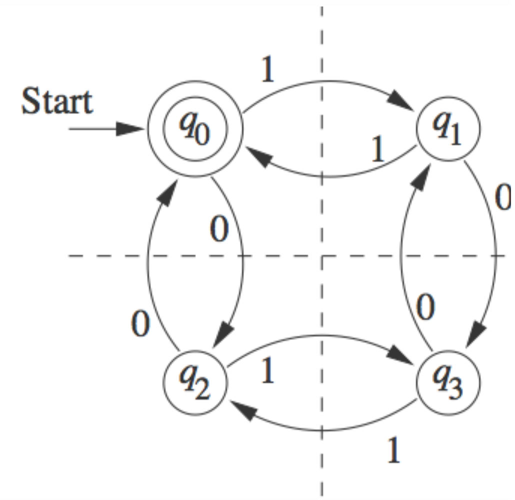
- L1 = { aa, ab, ac, ..., ba, bb, ..., zz }   (finite)
  - start state, final states
- L2 = *all letter sequences*   (infinite)
  - (cycle)
- L3 = *all alphanumeric strings that start with a letter*

Construct Finite Automata for ...

- L4 = *all letter strings with at least a vowel*

- L5 = *all letter strings with vowels in order*

- L6 = *all 01 strings with even number of 0's and even number of 1's*



Formal Definitions

Described by a 5-tuple:

$$M = (Q, \Sigma, \delta, q_0, F)$$

Q = Set of States
Finite Number of states

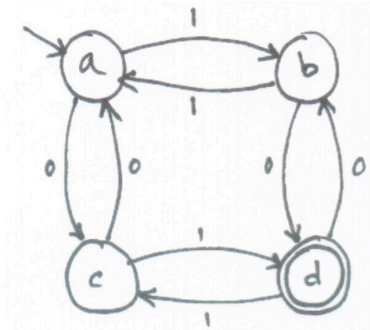
Σ = Alphabet, a Finite Set of Symbols

δ = The TRANSITION FUNCTION

$$\delta : Q \times \Sigma \rightarrow Q$$

q_0 = The STARTING STATE
$q_0 \in Q$ (or "INITIAL" STATE)

F = The set of ACCEPT states
(or "FINAL" STATES) $F \subseteq Q$



$Q = \{a, b, c, d\}$

$\Sigma = \{0, 1\}$

$q_0 = a$

$F = \{d\}$

$\delta =$

Symbol

States	0	1
a	c	b
b	d	a
c	a	d
d	b	c

Language, String, Machine

"THE LANGUAGE THAT M ACCEPTS IS A."

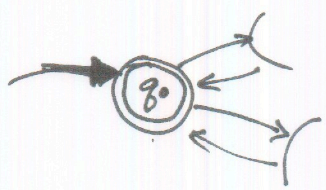
"THE LANGUAGE OF M"

"M RECOGNIZES A." ← yes

"~~M ACCEPTS A.~~" M accepts/rejects strings

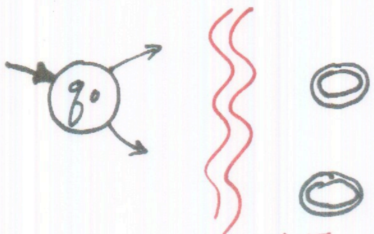
THE EMPTY STRING

ϵ (epsilon, ε)



THE EMPTY LANGUAGE

$\emptyset = \{\}$



NOT REACHABL

NOTE:

$\{\epsilon\} \neq \emptyset$

$\epsilon \neq \emptyset$

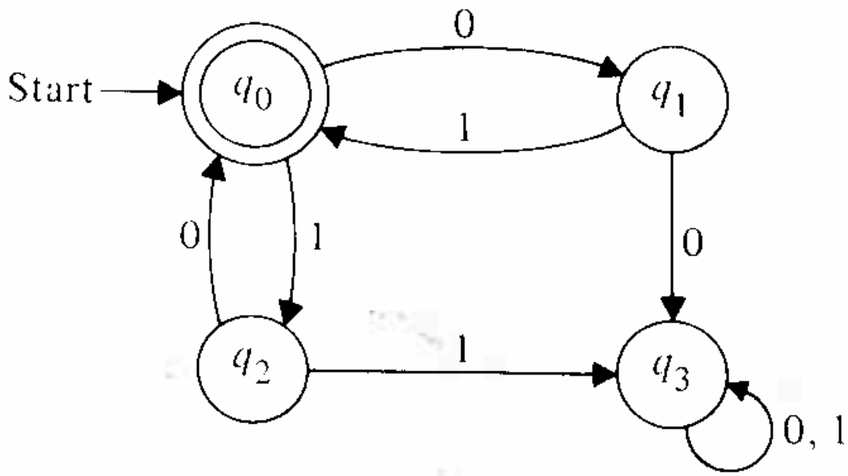
If a machine accepts NO strings then it recognizes the EMPTY LANGUAGE

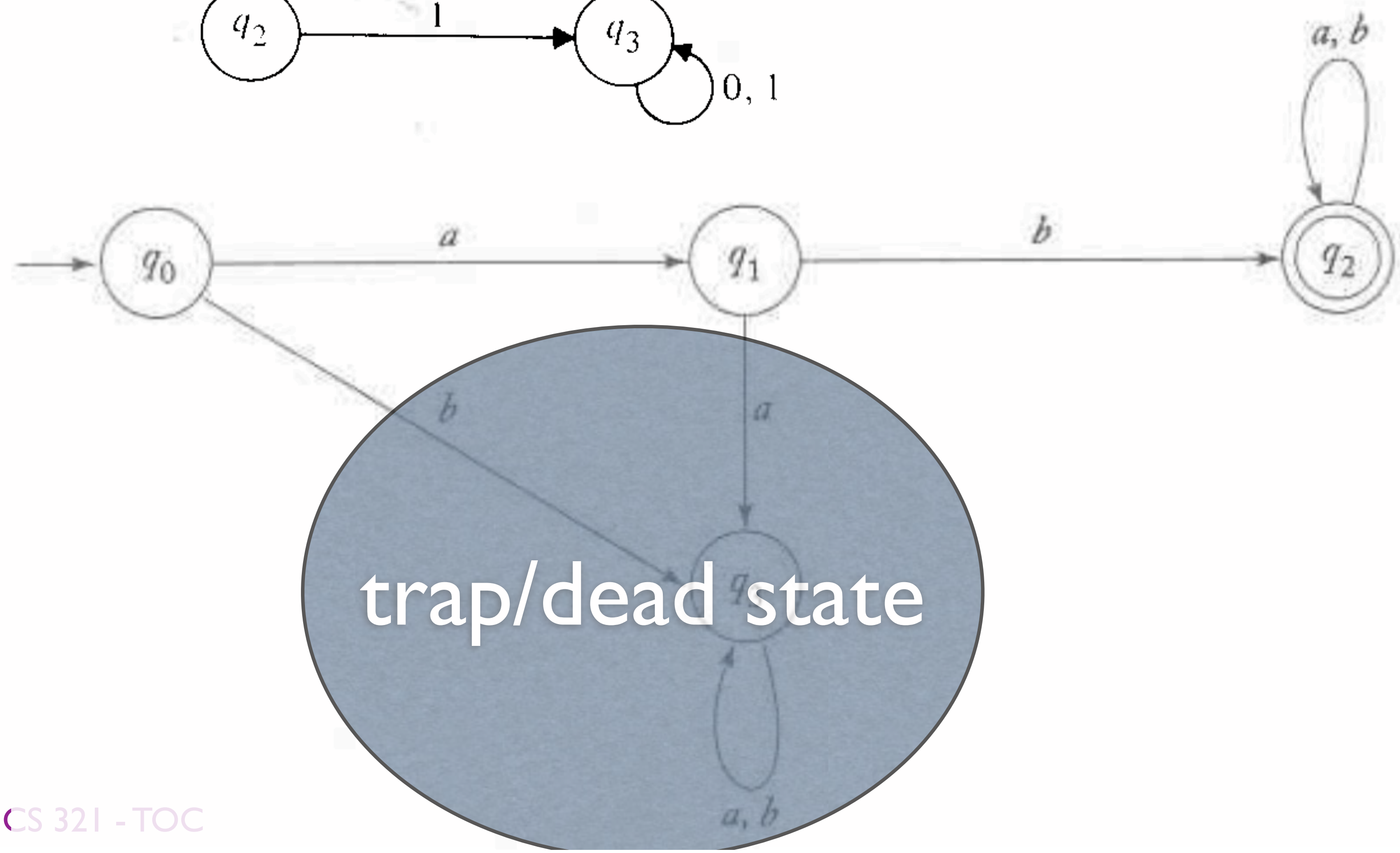
does M accept any string?

does M accept empty string?

does M recognizes the empty language?

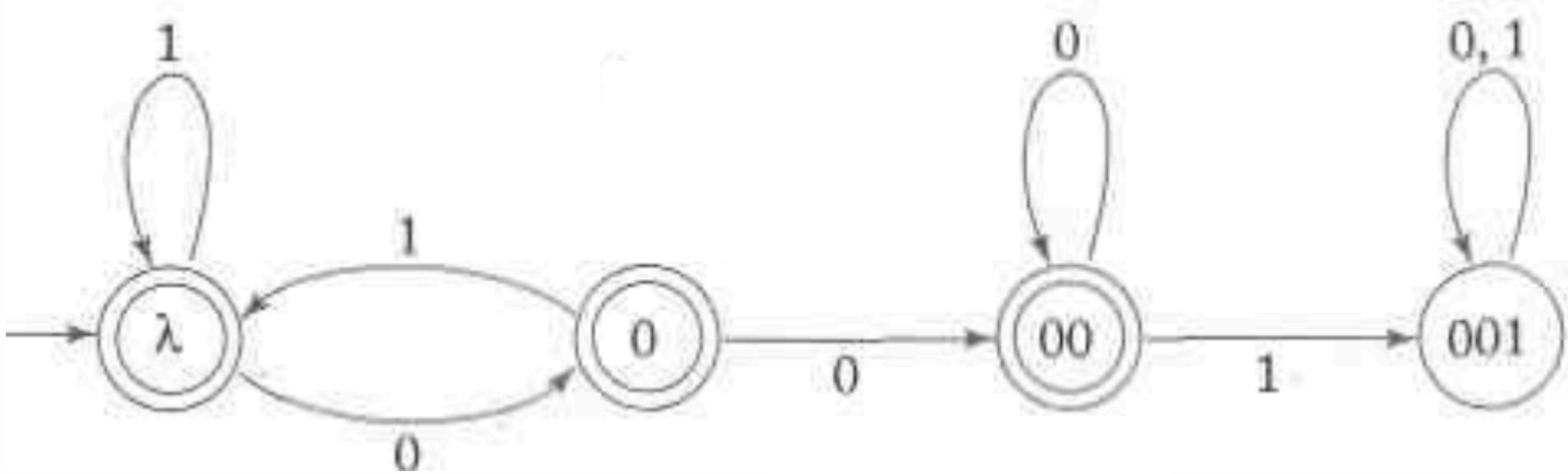
What's the language of ...





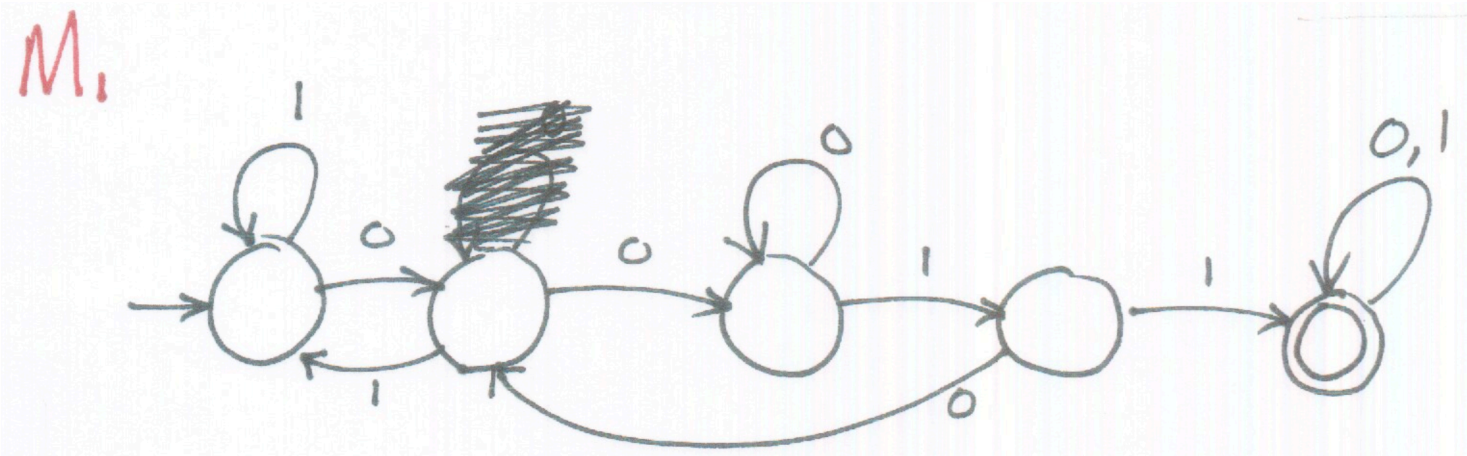
Construct Finite Automaton for ...

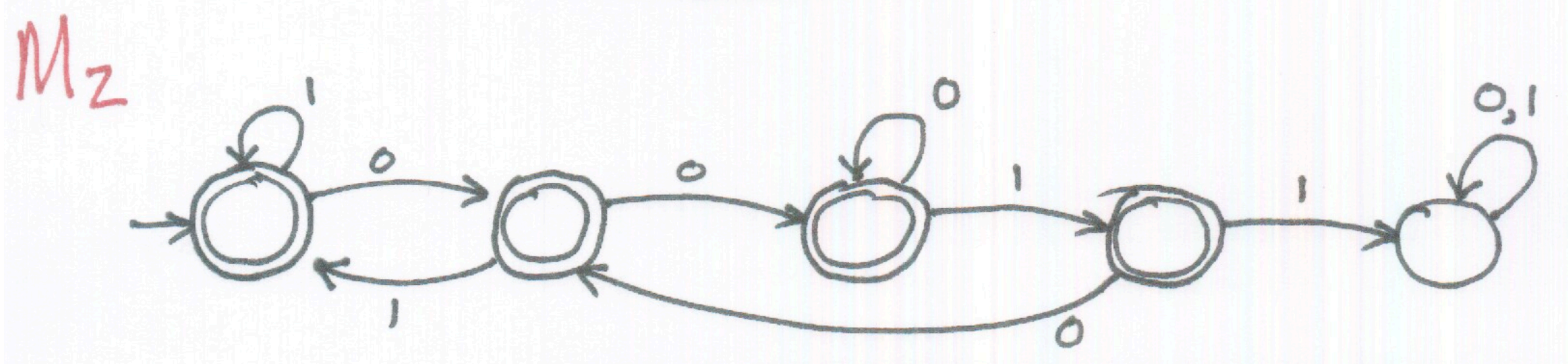
- any string that does not contain 001 in it
- try simpler problem: a string that does contain 001



Construct Finite Automaton for ...

- any string that does not contain 0011 in it
- try simpler problem: a string that does contain 0011





Compliment Language

Notation

$L(M_1)$ = The language that M_1 recognizes.
= The set of strings over $\{0,1\}^*$ that contain 0011 as a substring.

$L(M_2)$ = The set of strings over $\{0,1\}^*$ that do not contain 0011.

COMPLIMENTING A LANGUAGE

They are sets, after all.

$$L(M_1) = \overline{L(M_2)}$$

The "Universe"
All possible strings made with symbols from the alphabet.

$\Sigma = \{0,1\}$ Universe = $\{0,1\}^*$

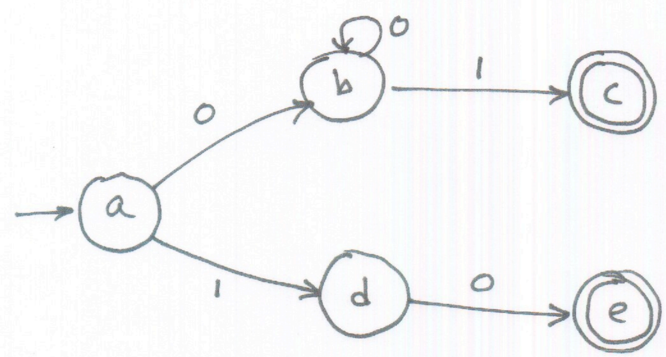
Set compliment is always relative to some Universe; (implicitly).

9

just flip final/non-final!

Dead States Formally Defined

WHAT DOES THIS F.S.M. RECOGNIZE?



Recognizes 10

Also 01, 001, 0001, ... 0^+1

$L = \{ w \mid$ w is either 10 <u>or</u> a string of at least one 0 followed by a single 1 $\}$

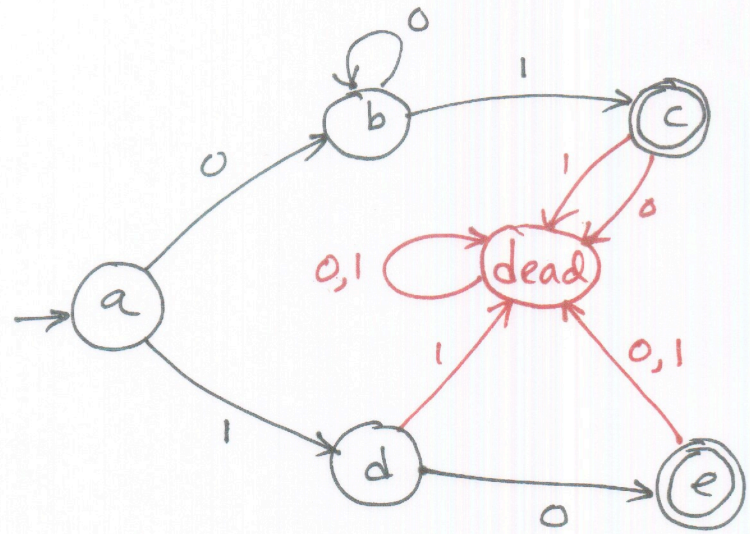
What about

111

1010

What happens?

DEAD STATES



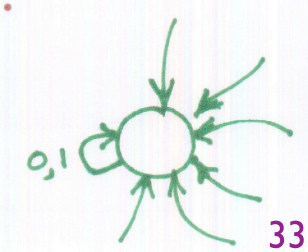
δ is a function

FORMALLY, must be defined

$\delta(c, 1) = ?$

If some transitions are missing, add a dead state.

(Often, prefer not to show dead state.)



Formal Definition of Computation

Let $M = (Q, \Sigma, \delta, q_0, F)$

Let $w = w_1 w_2 \cdots w_N$ be a string
where $w_i \in \Sigma$

M accepts w if there is a sequence of states
$r_0, r_1, r_2, \ldots, r_N$ in Q
such that

$r_0 = q_0$

$\delta(r_i, w_{i+1}) = r_{i+1}$ for $0 \leq i < N$

$r_N \in F$

We say...

M "recognizes" Language A
if $A = \{ w \mid M \text{ accepts } w \}$