Great Theoretical Ideas In Computer Science

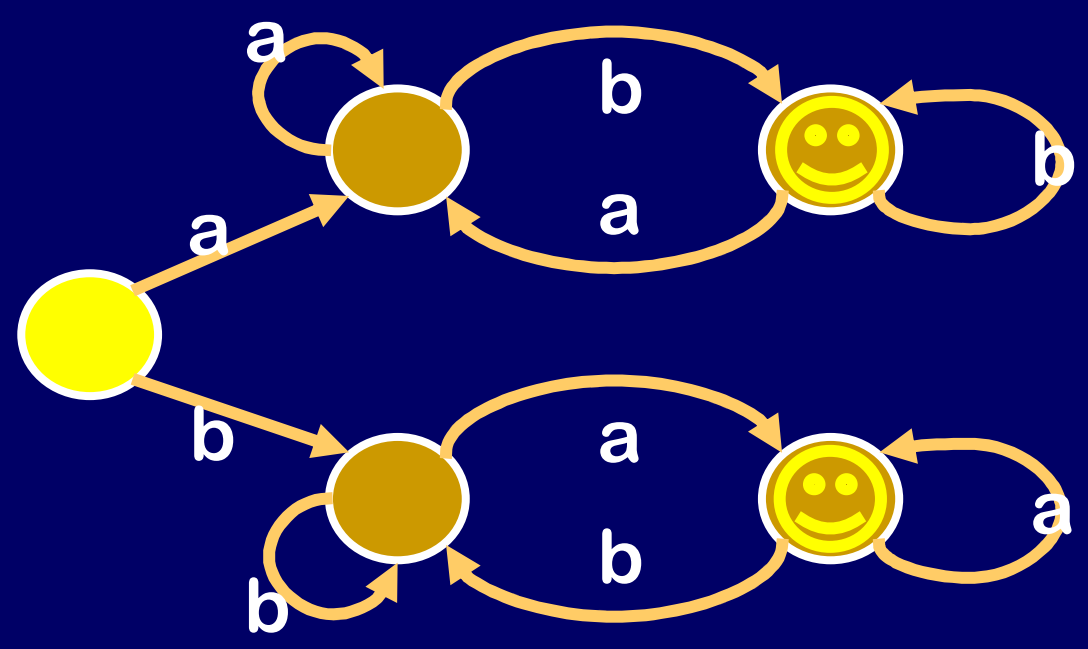
John Lafferty

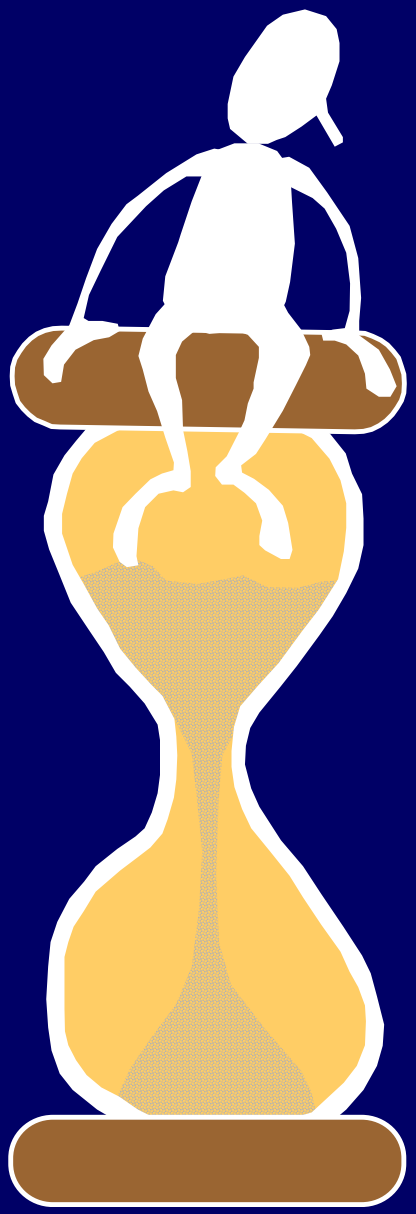
CS 15-251 Fall 2005

Lecture 10 Sept 29 2005

Carnegie Mellon University

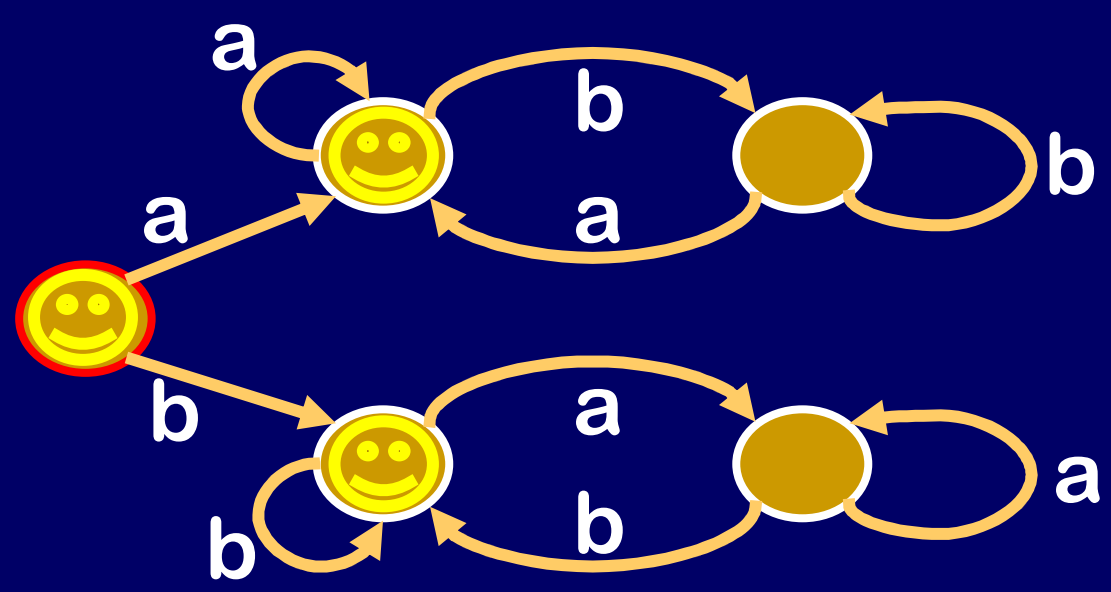
One Minute To Learn Programming: Finite Automata





Let me teach you a programming language so simple that you can learn it in less than a minute.

Meet "ABA" The Automaton!

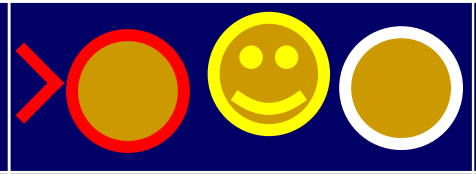
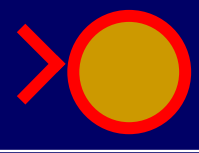
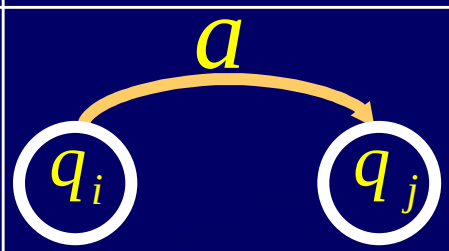


Input String	Result
aba	Accept
aabb	Reject
aabba	Accept
ε	Accept

The Simplest Interesting Machine:

Finite State Machine
OR
Finite Automaton

Finite Automaton

Finite set of states		$Q = \{q_o, q_1, q_2, \ldots, q_k\}$
A start state		q_o
A set of accepting states		$F = \{q_{i_1}, q_{i_2}, \ldots, q_{i_r}\}$
A finite alphabet	$a \quad b \quad \#$ $x \quad 1$	Σ
State transition instructions		$\partial : Q \times \Sigma \to Q$ $\partial(q_i, a) = q_j$

How Machine M operates.

M "reads" one letter at a time from the input string (going from left to right)

M starts in state q_0.

If M is in state q_i reads the letter a then

If $\delta(q_i. a)$ is undefined then CRASH.

Otherwise M moves to state $\delta(q_i,a)$

Let $M=(Q,\Sigma,F,\delta)$ be a finite automaton.

M ***accepts*** the string *x* if when *M* reads *x* it ends in an accepting state.

M ***rejects*** the string *x* if when *M* reads *x* it ends in a non-accepting state.

M ***crashes*** on *x* if *M* crashes while reading *x*.

The set (or language) accepted by M is:

$$L_M = \{x \in \Sigma^* \mid M \text{ accepts } x\}$$

$\Sigma^k \equiv$ All length k strings over the alphabet Σ

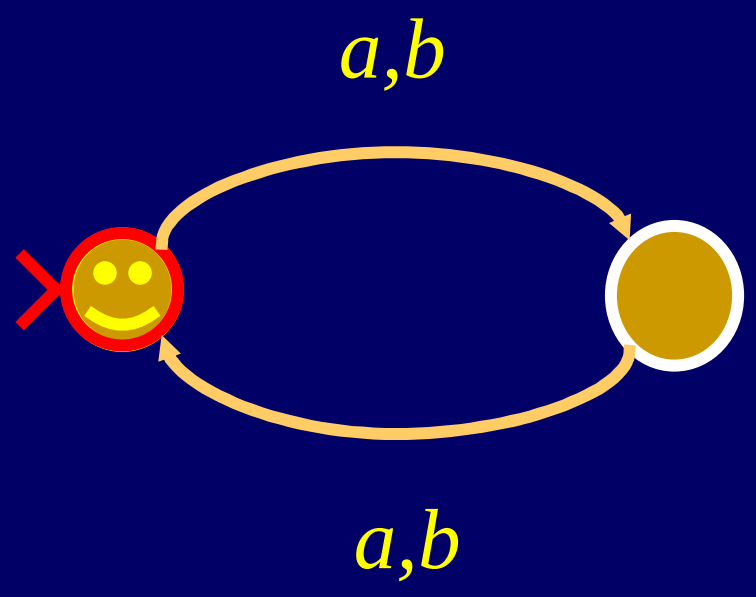
$$\Sigma^* \equiv \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$

Notice that this is $\{\varepsilon\}$

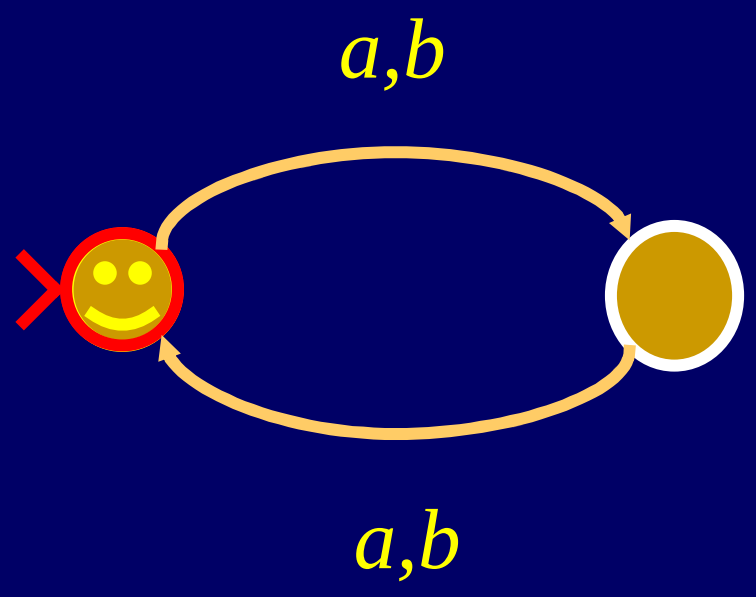
What is the language accepted by this machine?

$L = \{a,b\}^*$ = all finite strings of *a*'s and *b*'s

What is the language accepted by this machine?



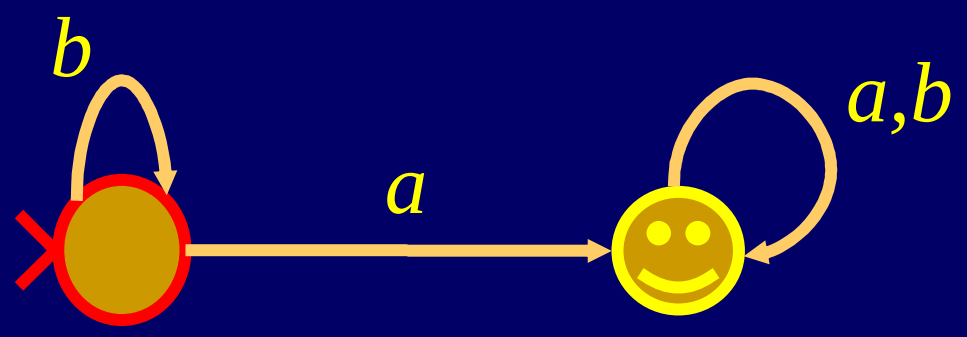
What is the language accepted by this machine?



L = all even length strings of a's and b's

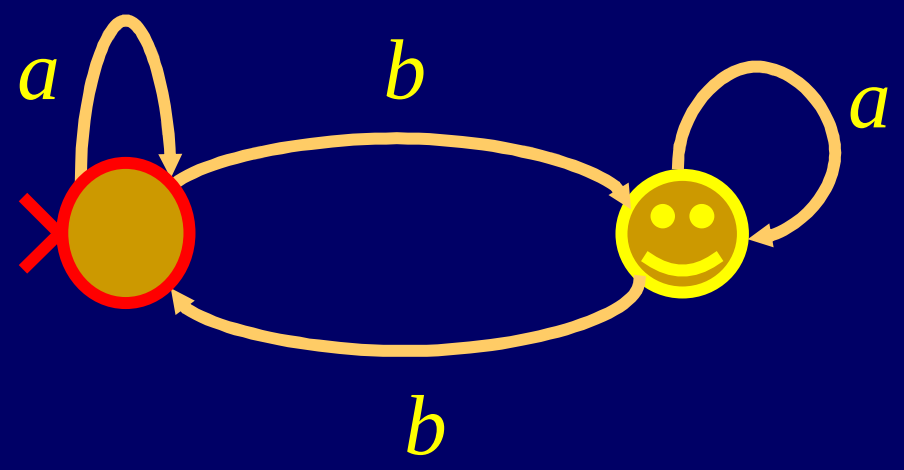
What machine accepts this language?

L = **all strings in** $\{a,b\}^*$ **that contain at least one** a

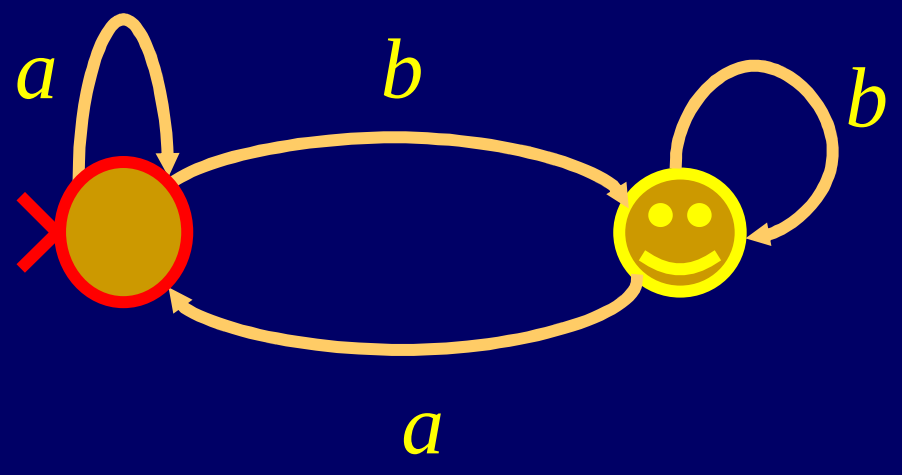


What machine accepts this language?

L = strings with an odd number of b's and any number of a's

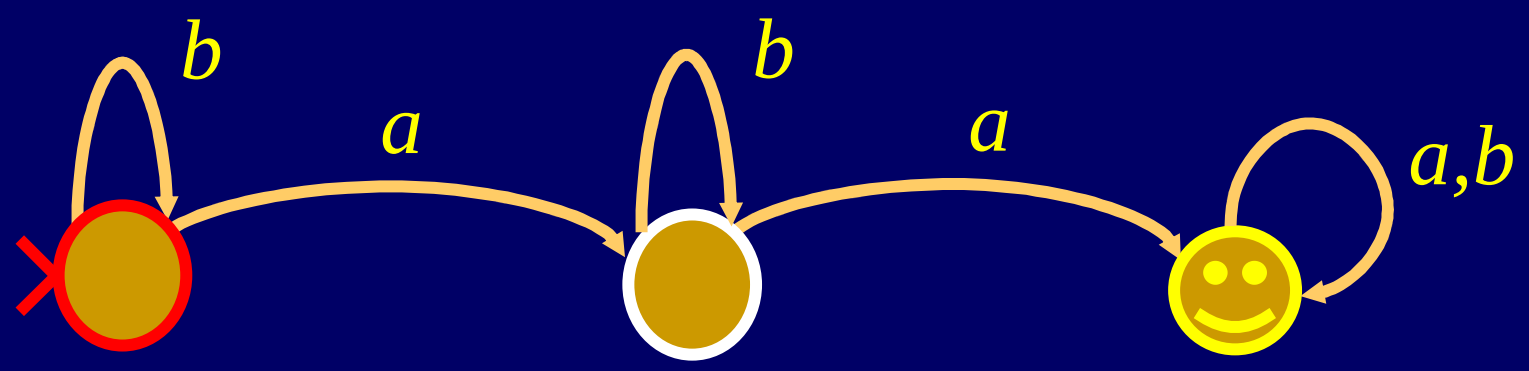


What is the language accepted by this machine?

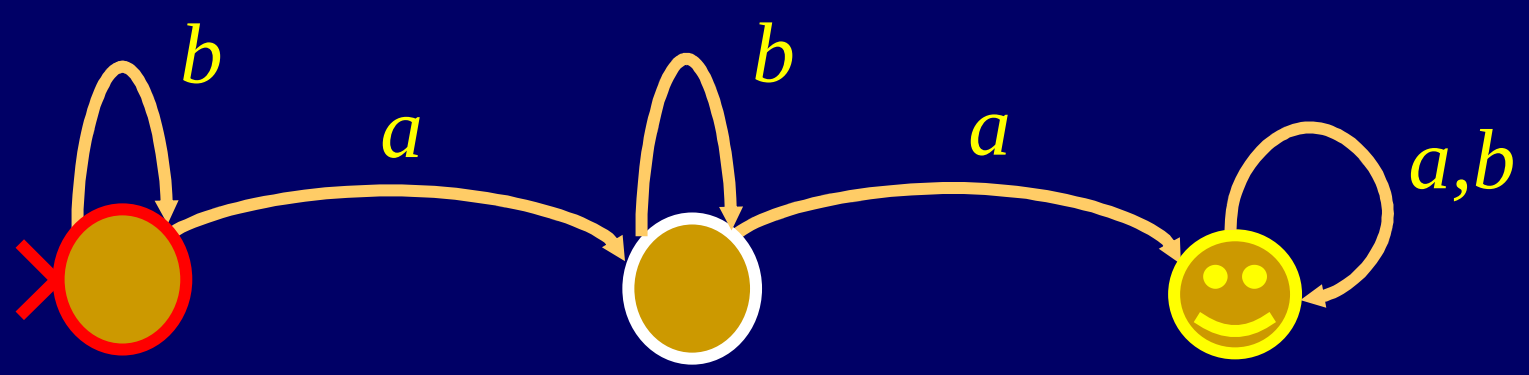


L = any string ending with a *b*

What is the language accepted by this machine?



What is the language accepted by this machine?



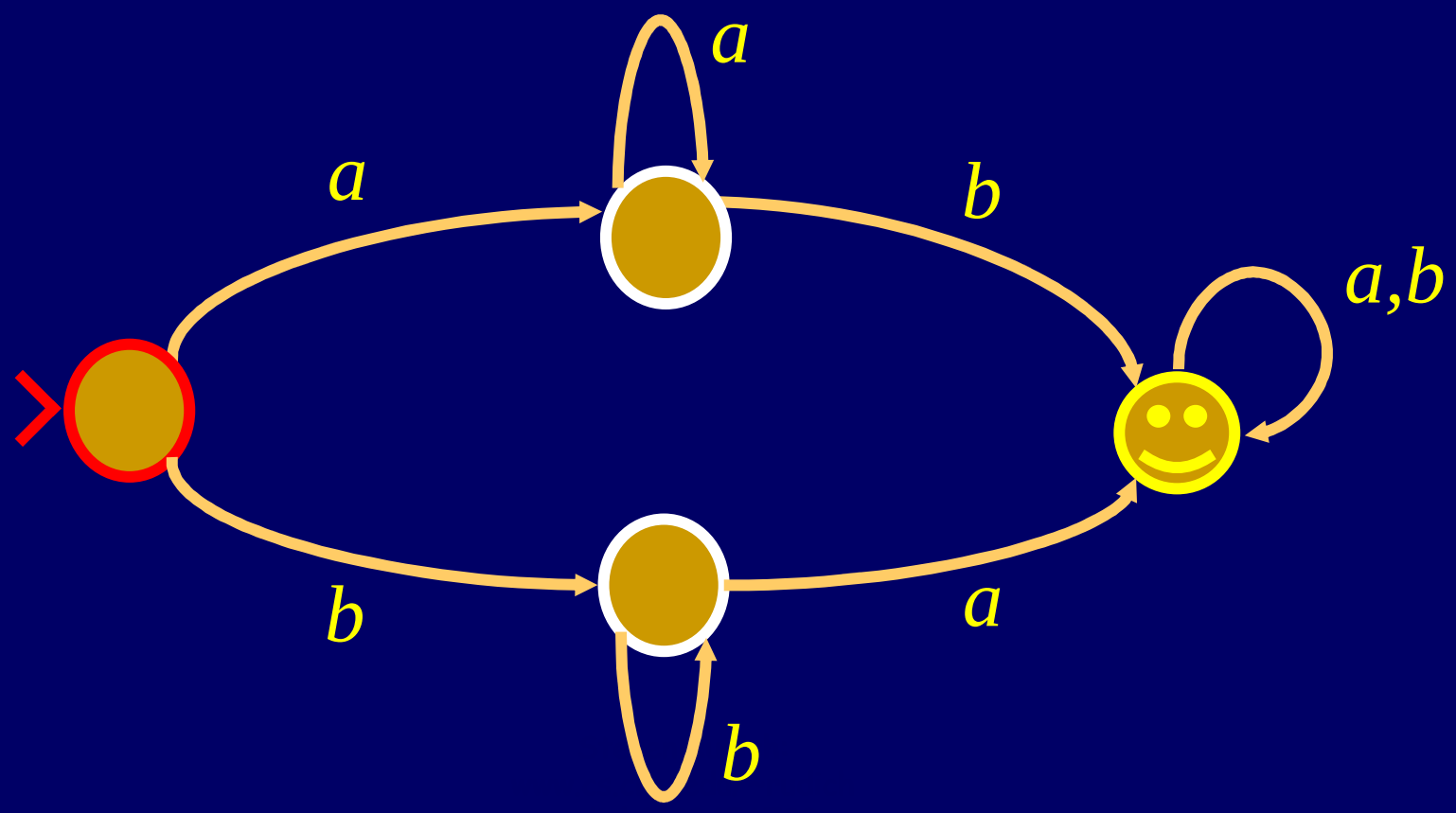
L = any string with at least two a's

What machine accepts this language?

L = any string with an a and a b

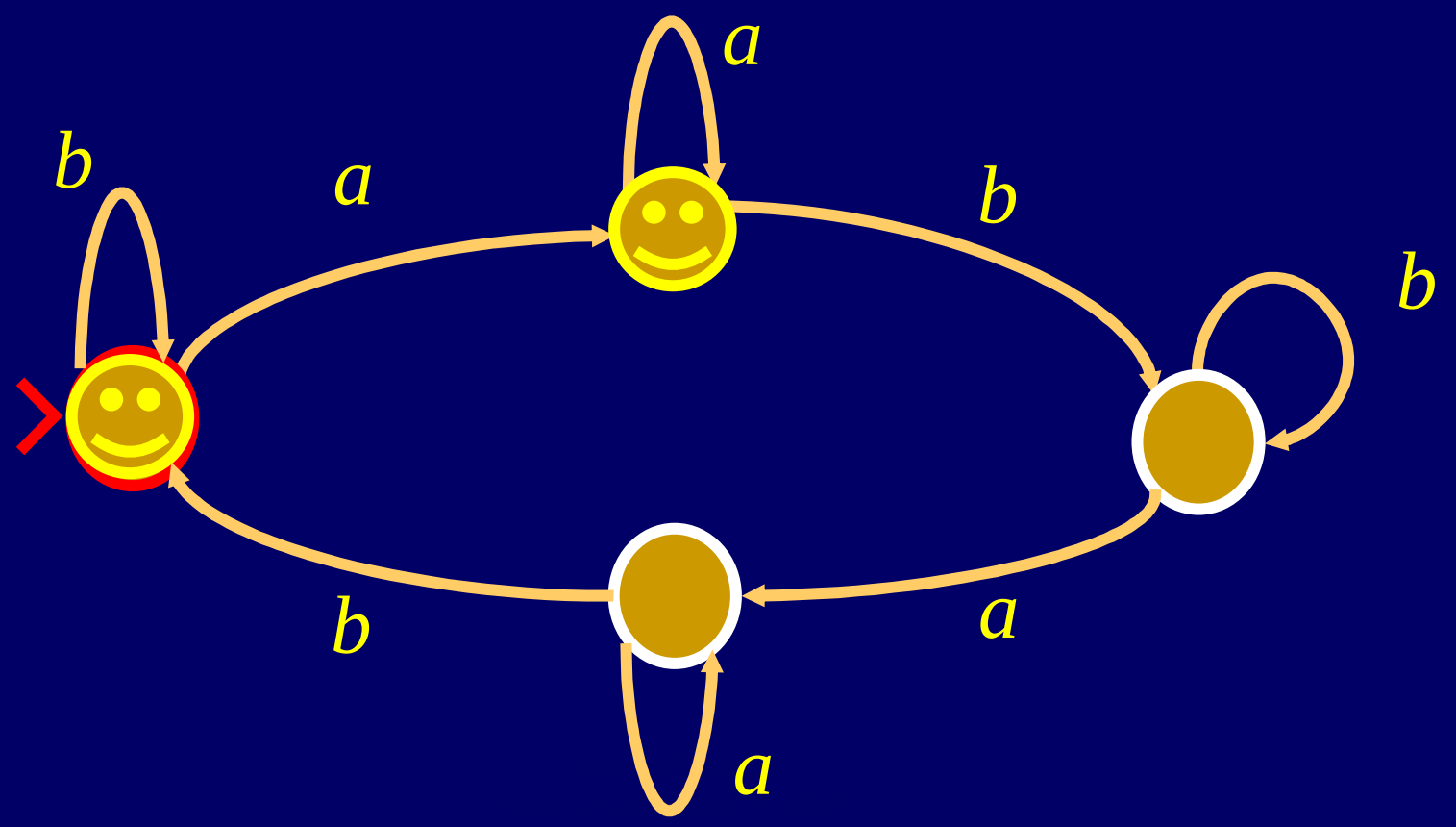
What machine accepts this language?

L = any string with an *a* and a *b*



What machine accepts this language?

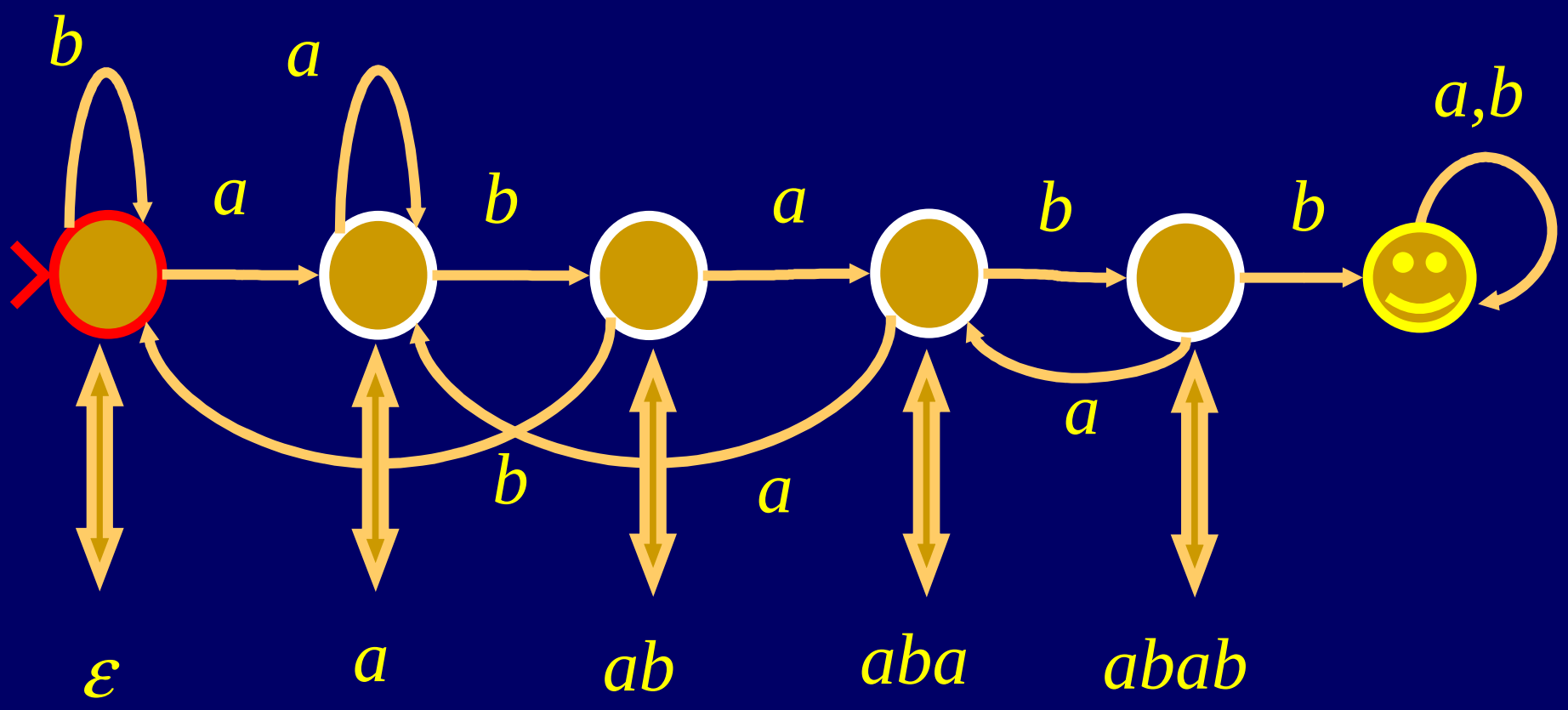
L = strings with an even number of *ab* pairs



$L =$ all strings containing $ababb$ as a consecutive substring

Consider : ababbb

L = all strings containing *ababb* as a consecutive substring



Invariant: I am state **s** exactly when **s** is the longest suffix of the input (so far) that forms a prefix of *ababb.*

The "grep" Problem

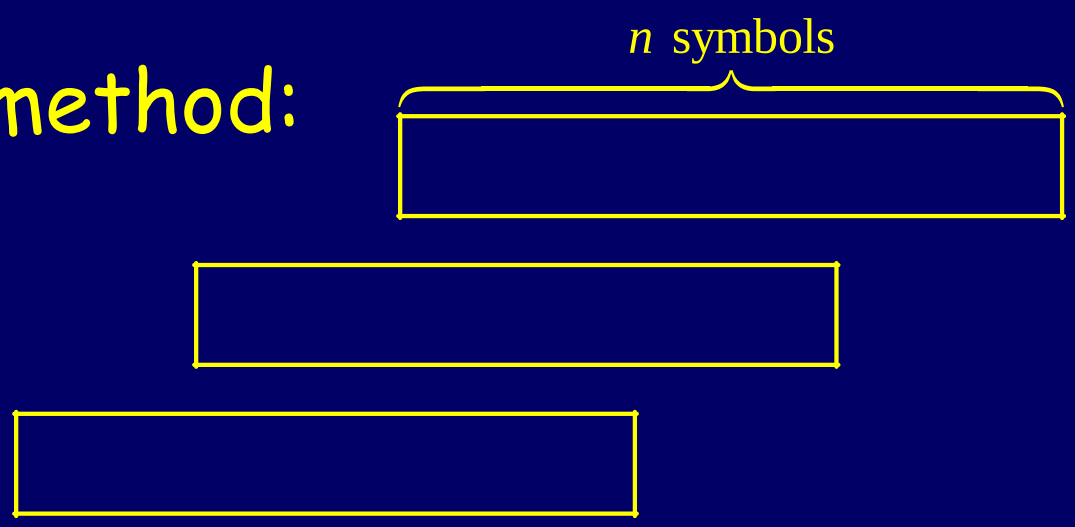
Input:

- text T of length t
- string S of length n

Problem:

Does the string S appear inside the text T?

Naïve method:



$$a_1, a_2, a_3, \ldots\ldots\ldots\ldots\ldots\ldots, a_t$$

Cost: $O(nt)$ comparisons

Automata Solution

- Build a machine *M* that accepts any string with *S* as a consecutive substring.
- Feed the text to *M*.
- Cost: *t* comparisons + time to build *M*.
- As luck would have it, the Knuth, ***Morris***, Pratt algorithm builds *M* quickly.

Real-life uses of finite state machines

- grep
- coke machines
- thermostats (fridge)
- elevators
- train track switches
- lexical analyzers for parsers

Any $L \subseteq \Sigma^*$ is defined to be a <u>language</u>.

L is just a set of strings. It is called a language for historical reasons.

Let $L \subseteq \Sigma^*$ be a language.

L is called a regular language if there is some finite automaton that accepts L.

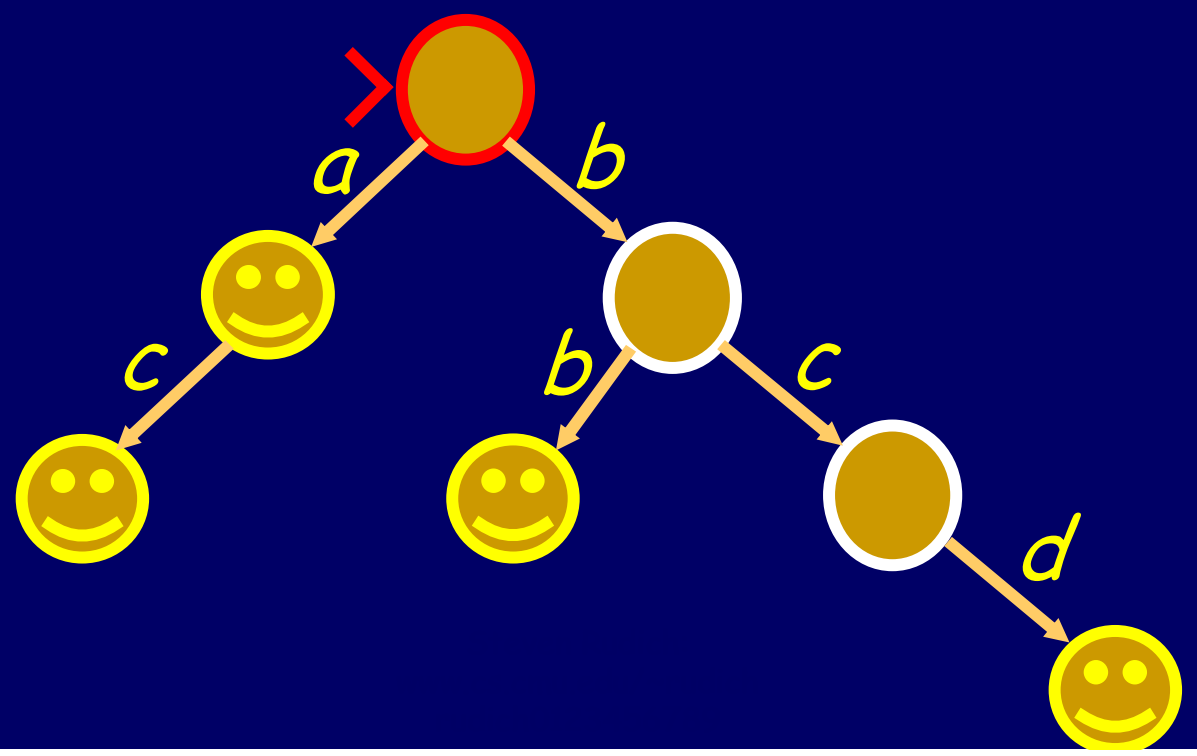
In this lecture we have seen many regular languages.

- Σ^*
- even length strings
- strings containing *ababb*

Theorem: Any finite langage is regular.

Proof: Make a machine with a "path" for each string in the language, sharing prefixes

Example: $L = \{a, bcd, ac, bb\}$



Are all languages regular?

Consider the language

$$a^n b^n = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$$

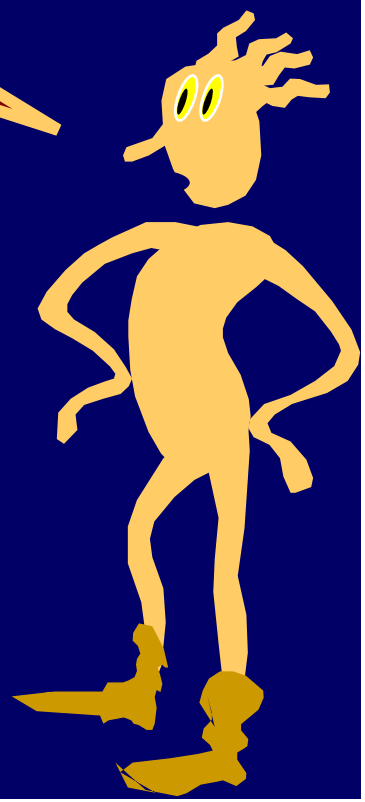
i.e., a bunch of *a*'s
followed by an equal
number of *b*'s

No finite automaton accepts this language.

Can you prove this?

$a^n b^n$ is not regular.
No machine has enough states to keep track of the number of a's it might encounter.

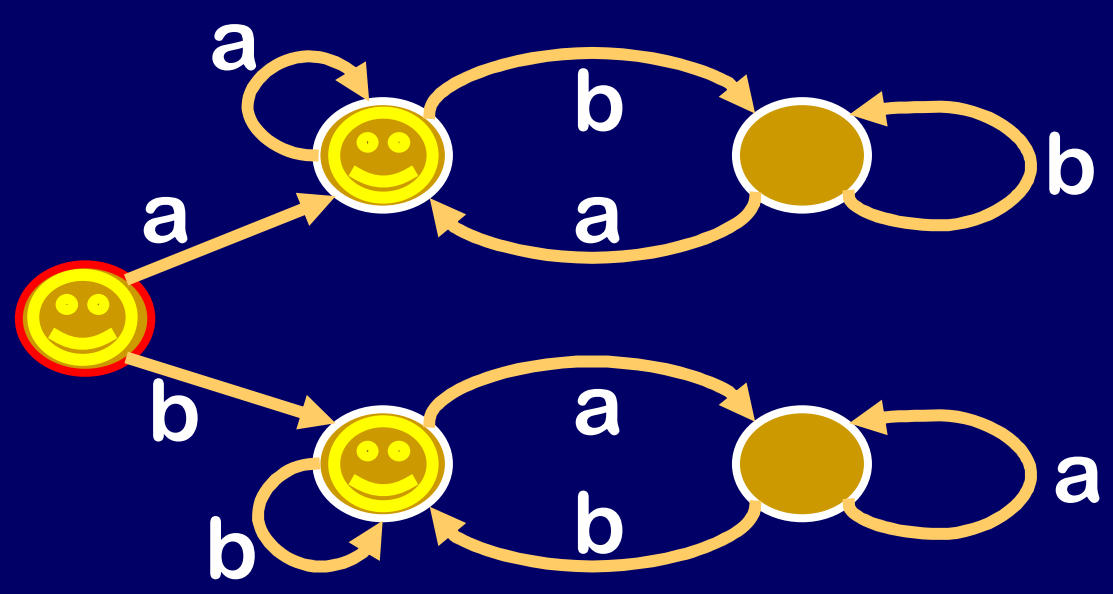
That is a fairly weak argument. Consider the following example…



L = strings where the # of occurrences of the pattern *ab* is equal to the number of occurrences of the pattern *ba*

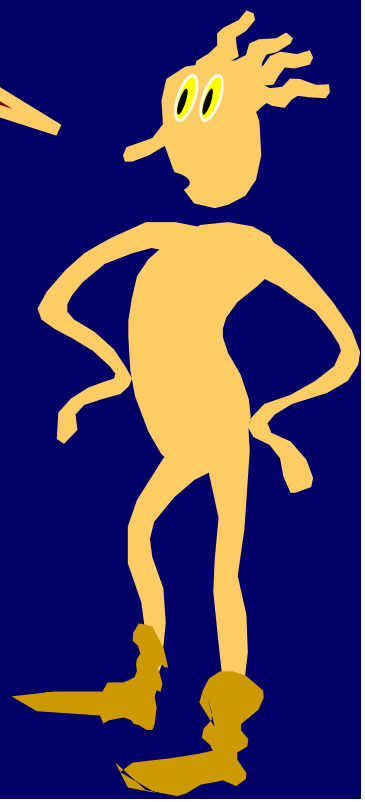
Can't be regular. No machine has enough states to keep track of the number of occurrences of *ab*.

Remember "ABA"?



ABA accepts only the strings with an equal number of *ab*'s and *ba*'s!

Let me show you a professional strength proof that $a^n b^n$ is not regular….



Pigeonhole principle: Given n boxes and $m > n$ objects, at least one box must contain more than one object.

Letterbox principle: If the average number of letters per box is a, then some box will have at least a letters. (Similarly, some box has at most a.)

Theorem: $a^n b^n$ is not regular.

Suppose it is. Then there is some machine M that accepts it.

Suppose M has k states.

For $\underbrace{0 \le i \le k}_{k+1}$, let S_i be the state of M after reading a^i

By pigeonhole, there are some i, j with $S_i = S_j$

Thus machine is in same state after reading $a^i b^i$ and $a^j b^i$.

Contradiction! Thus, no such M exists.

Professional Strength Proof

Theorem: $a^n b^n$ is not regular.

Proof: Assume that it is. Then $\exists$ M with k states that accepts it.

For each $0 \leq i \leq k$, let S_i be the state M is in after reading a^i.

$\exists i,j \leq k$ s.t. $S_i = S_j$, but $i \neq j$

M will do the same thing on $a^i b^i$ and $a^j b^i$.

But a valid M must reject $a^j b^i$ and accept $a^i b^i$.

$\Rightarrow\Leftarrow$

MORAL:

Finite automata can't count.

Advertisement

You can learn much more about these creatures in the FLAC course.

Formal Languages, Automata, and Computation

- There is a unique smallest automaton for any regular language

- It can be found by a fast algorithm.

Cellular Automata

- Line up a bunch of ***identical*** finite automata in a straight line.

- Transitions are based on the states of the machine's two neighbors or an indicator that a neighbor is missing. (There is no other input.)

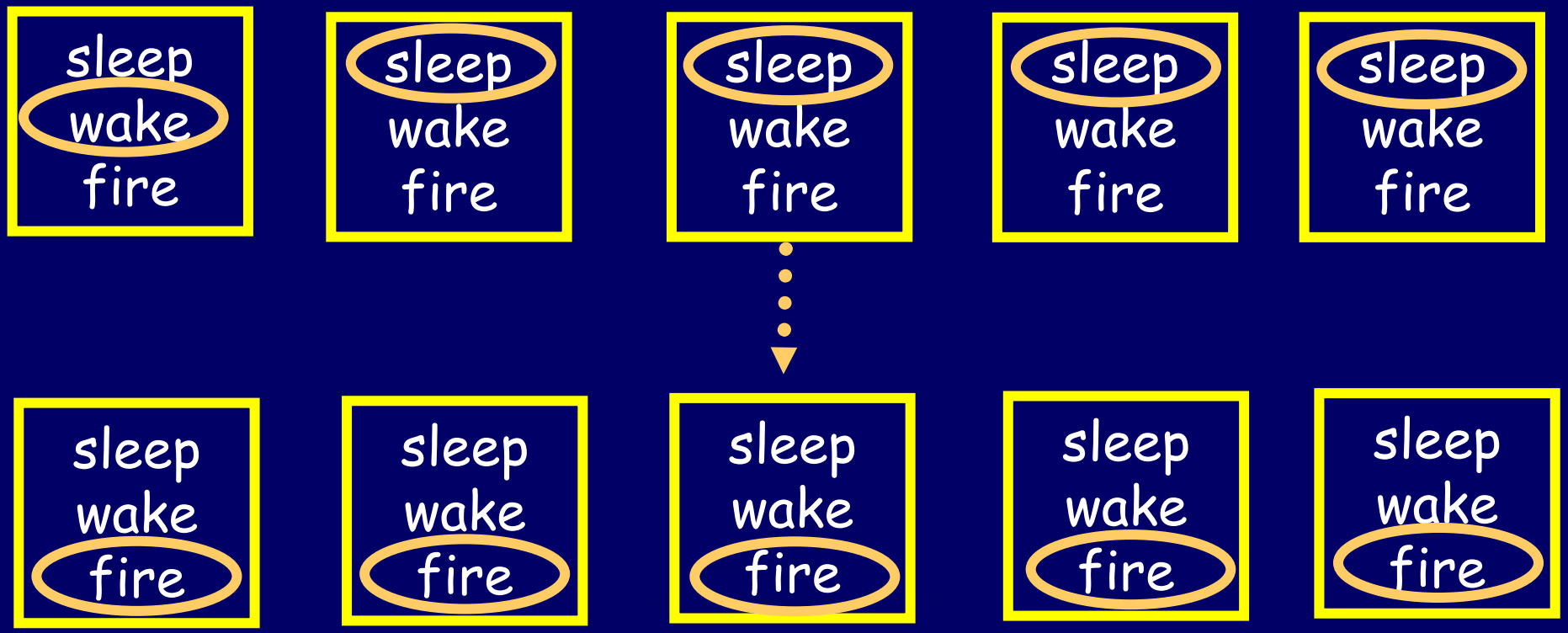
$$Q \times Q \times Q \rightarrow Q$$

$$(q_k, q_i, q_j) = q_l$$

- All cells move to their next states at the same time: synchronous transition

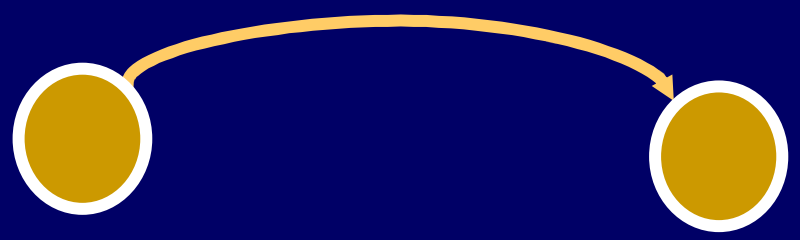
The Firing Squad Problem

- Five "soldiers" all start in the sleep state. You change the one on the left to the wake state.
- All five must get to the fire state at the same time (for the first time).

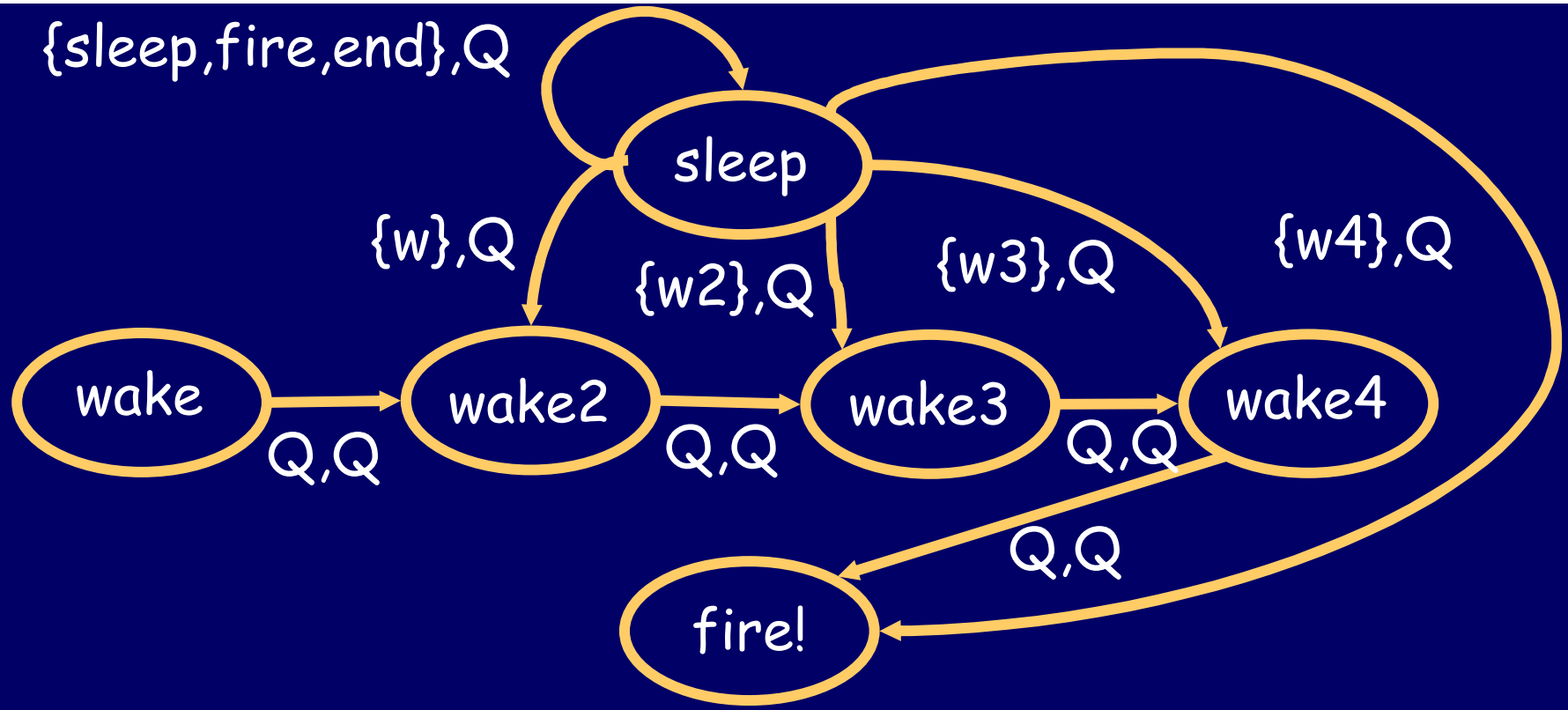


Shorthand

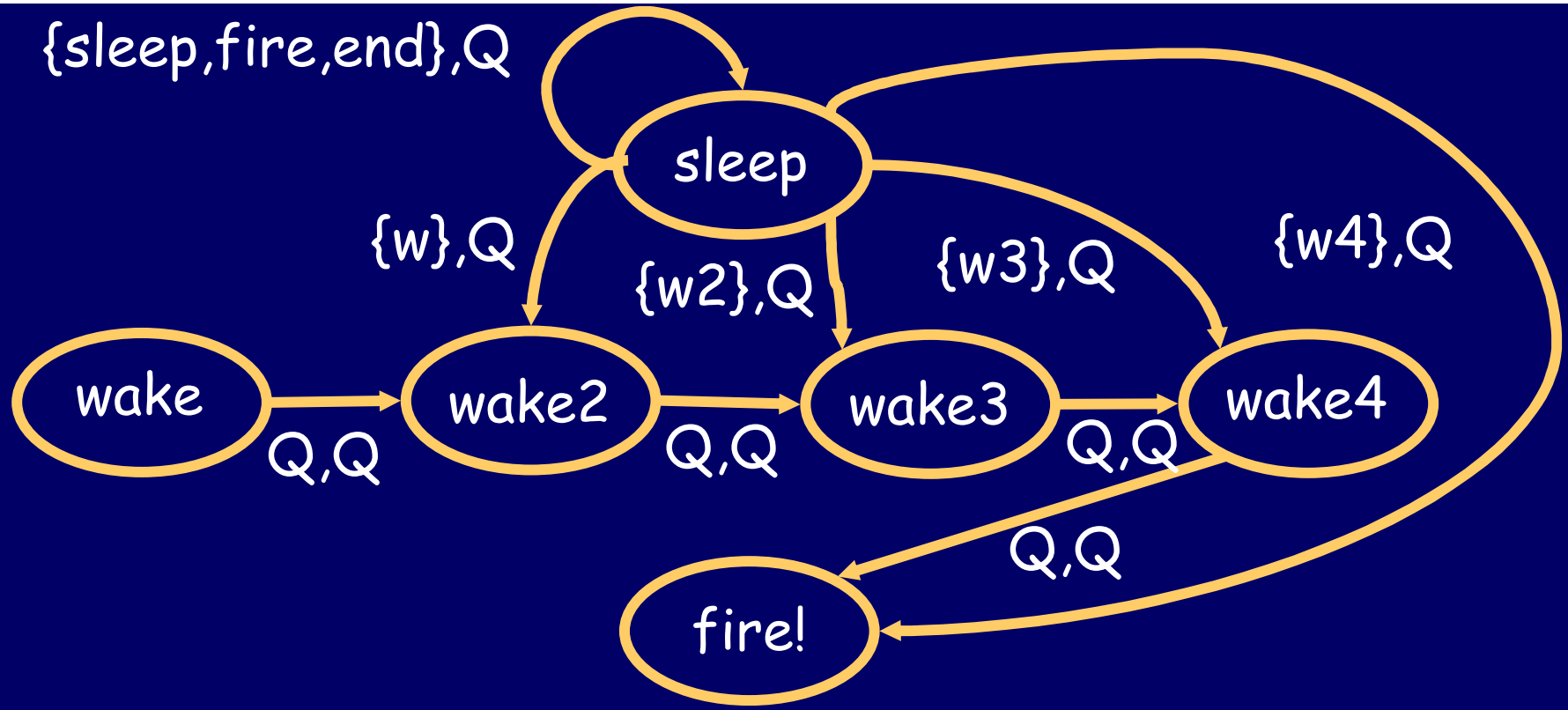
$$Q, \{a, b, d\}$$



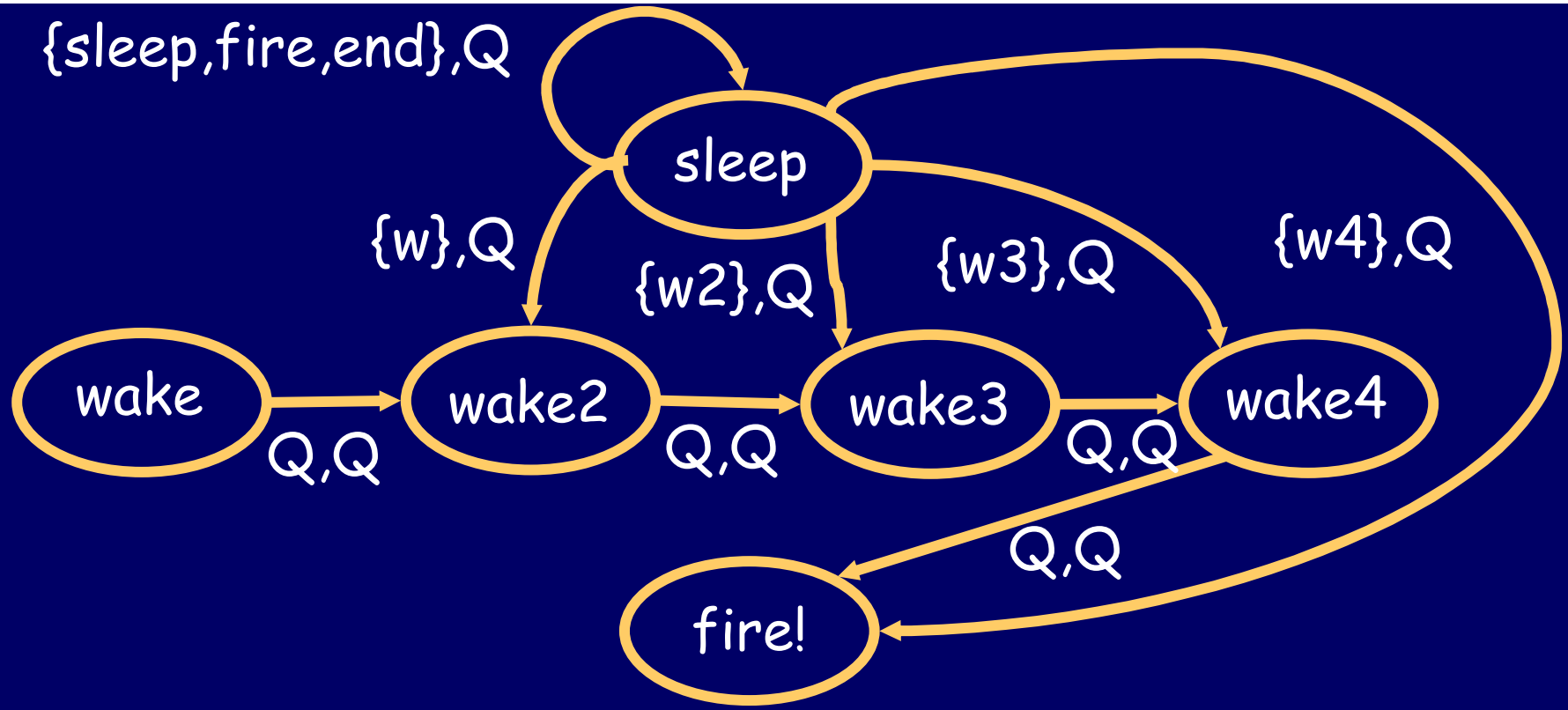
Means use this transition when your left neighbor is in any state at all and your right neighbor is in state *a*,*b*, or *d*.



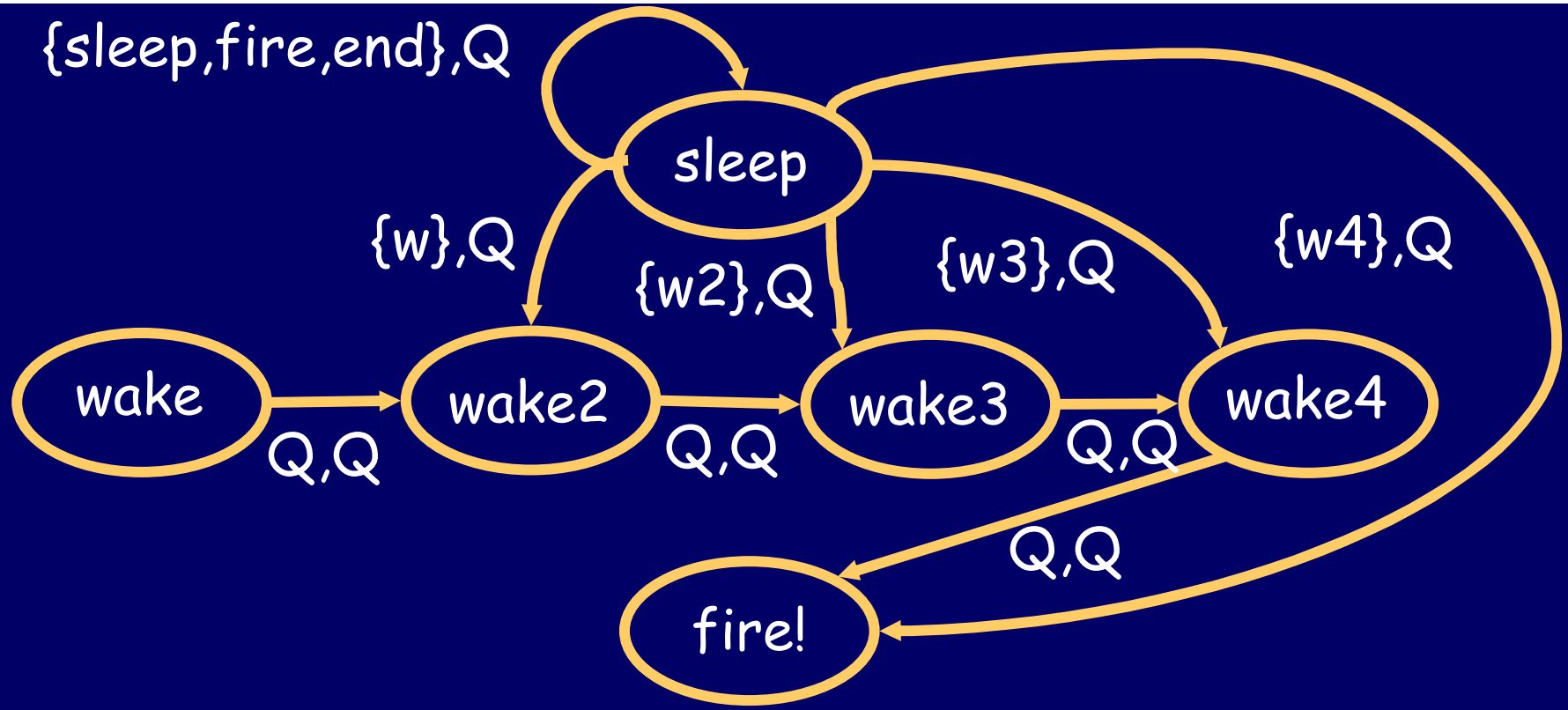
	1	2	3	4	5
	sleep	sleep	sleep	sleep	sleep
	sleep	sleep	sleep	sleep	sleep
	sleep	sleep	sleep	sleep	sleep
	sleep	sleep	sleep	sleep	sleep
	sleep	sleep	sleep	sleep	sleep



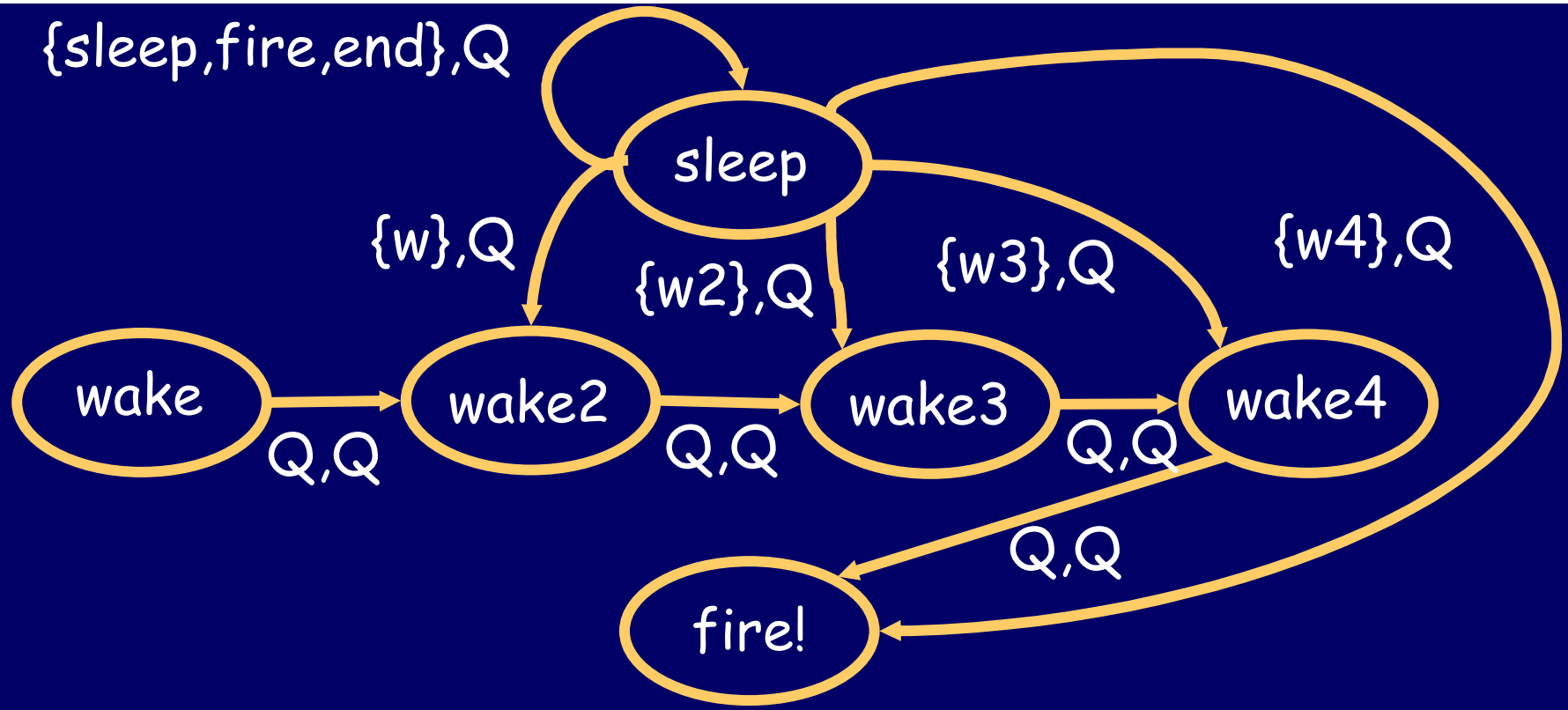
	1	2	3	4	5
	wake	sleep	sleep	sleep	sleep



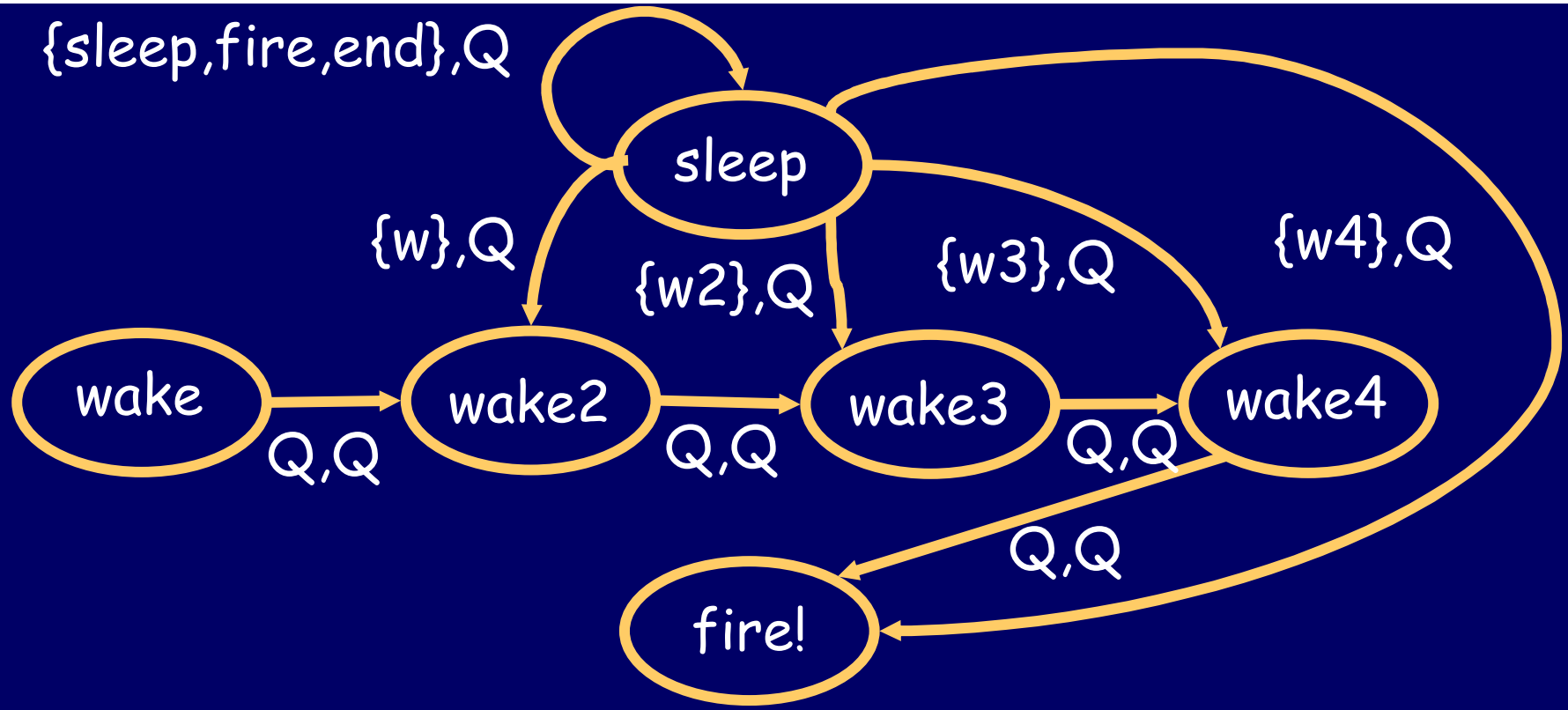
	1	2	3	4	5
	wake	sleep	sleep	sleep	sleep
	wake2	wake2	sleep	sleep	sleep



	1	2	3	4	5
	wake	sleep	sleep	sleep	sleep
	wake2	wake2	sleep	sleep	sleep
	wake3	wake3	wake3	sleep	sleep



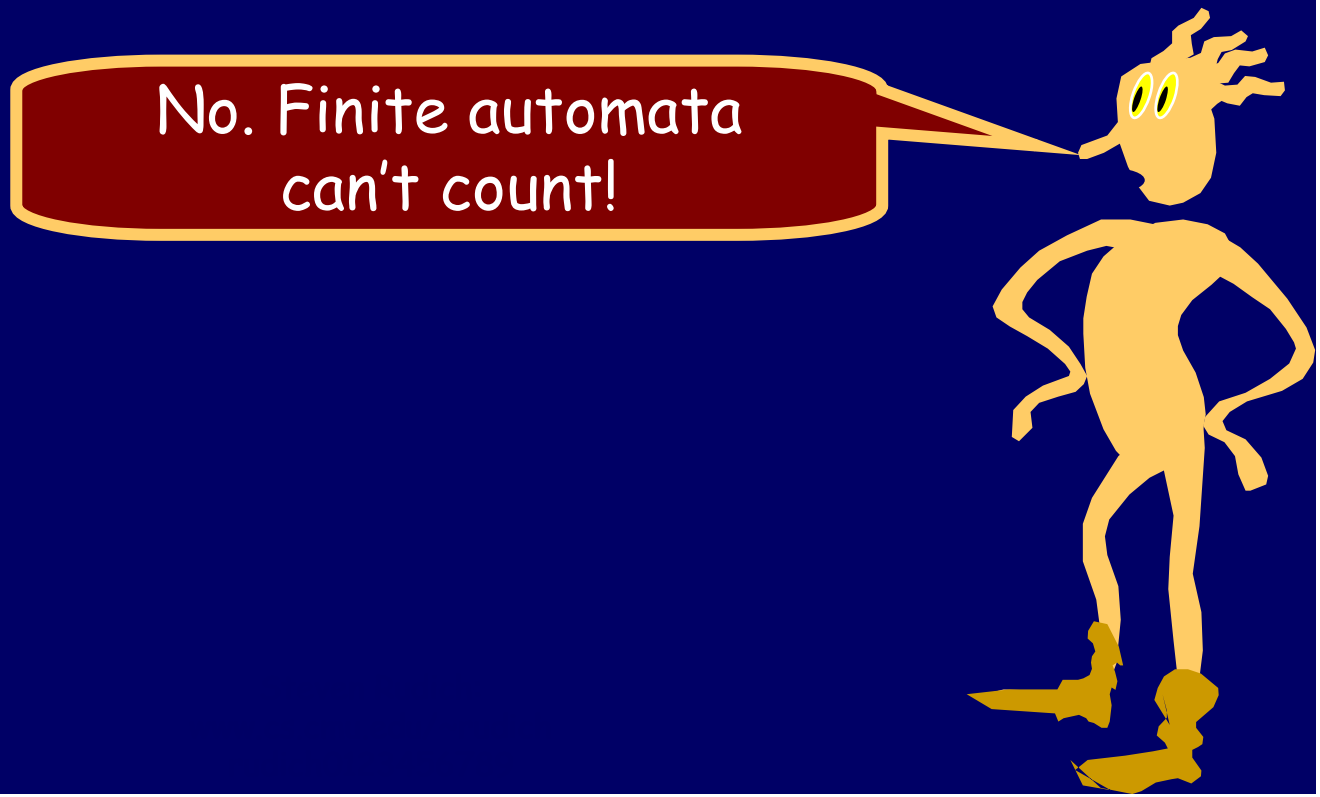
	1	2	3	4	5
	wake	sleep	sleep	sleep	sleep
	wake2	wake2	sleep	sleep	sleep
	wake3	wake3	wake3	sleep	sleep
	wake4	wake4	wake4	wake4	sleep



	1	2	3	4	5
	wake	sleep	sleep	sleep	sleep
	wake2	wake2	sleep	sleep	sleep
	wake3	wake3	wake3	sleep	sleep
	wake4	wake4	wake4	wake4	sleep
	fire	fire	fire	fire	fire

Question

Can you build the soldier's finite automaton brain before you know how many soldiers will be in the line?



Don't jump to conclusions! It *is* possible to design a single cellular automaton that works for any number of soldiers!