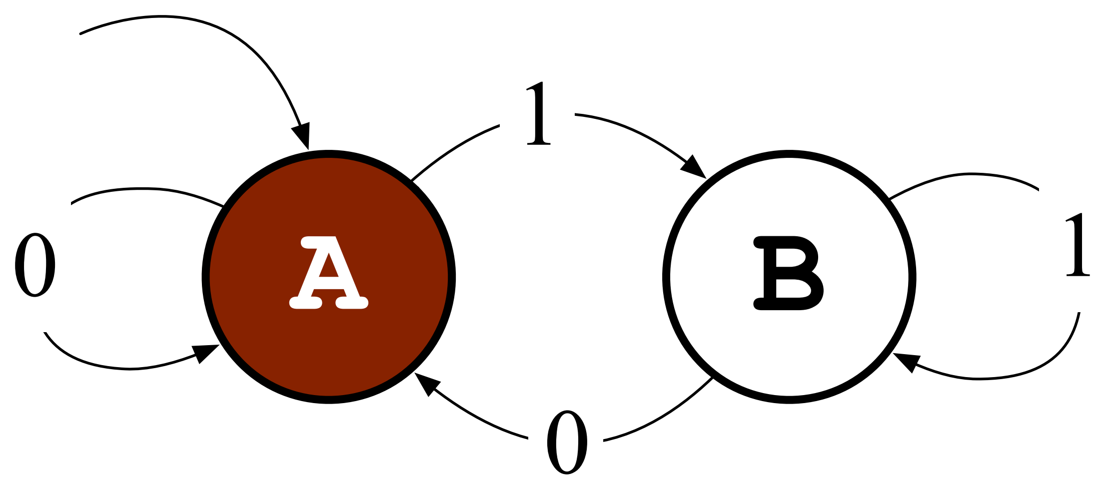
FSM has **states** and **transitions**.

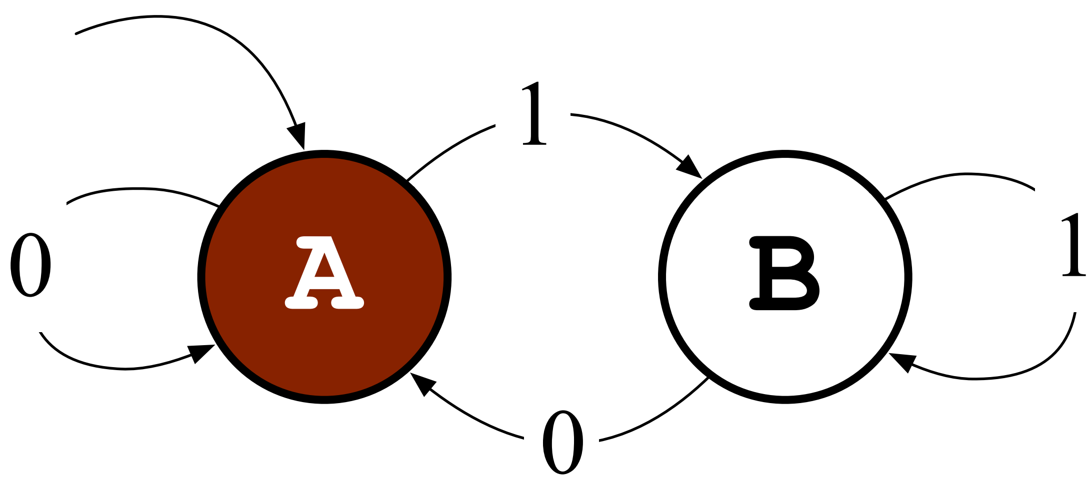
The **default state** is indicated with an arrow coming in from nowhere in particular.

The FSM can only be in one state at a time. The **current state** is indicated with color.

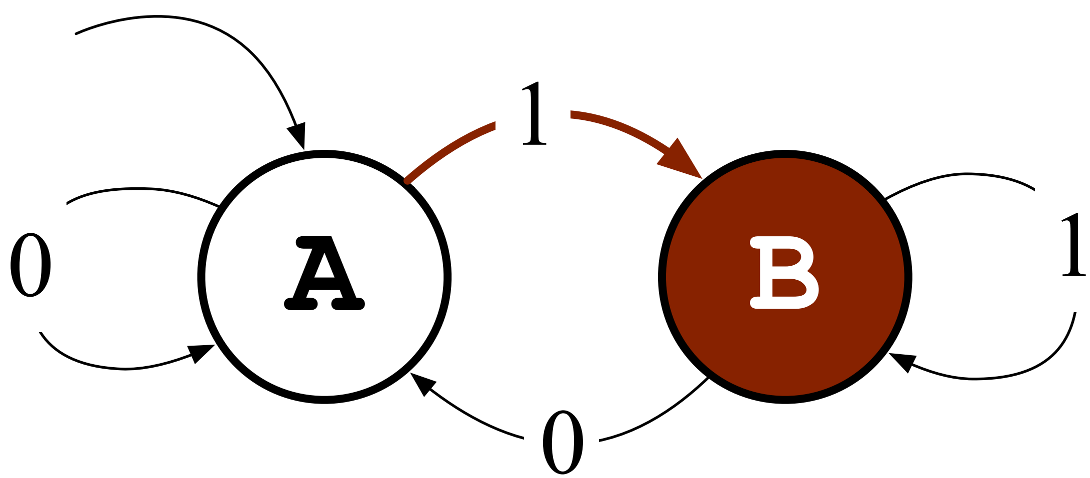


FSM has **states** and **transitions**.

The FSM can receive **signals**, also called **events**. These might cause the FSM to transition into another state. We only transition out of the current state if we receive a signal associated with an outgoing edge. In this case, A reacts to 0 and 1, but not 2 or 'monkey'.



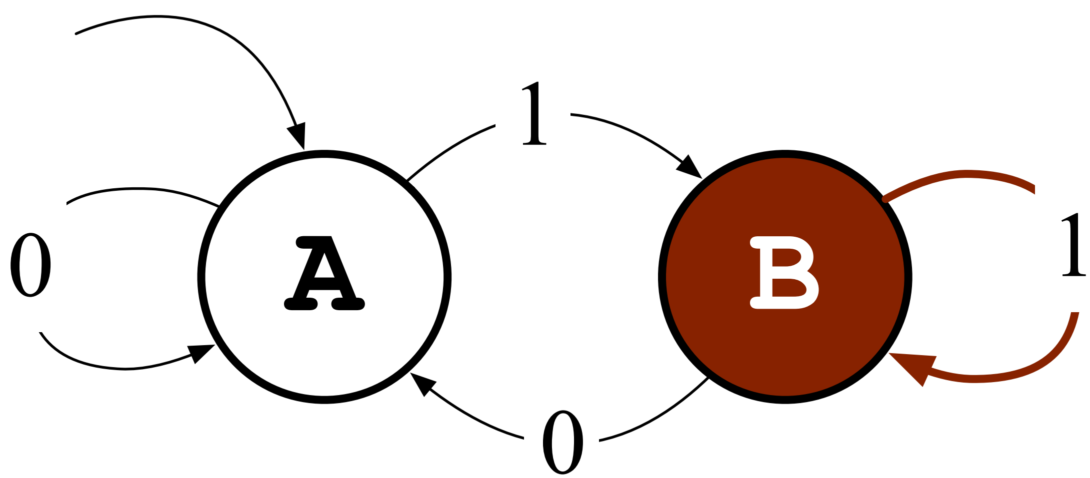
If we receive the signal 1, the transition from A to B is invoked and the FSM in now in state B.



Transitions can start and end on the same node. If we receive a 1, we transition from B… *to B*.

Why would we want to do that? *Actions.*

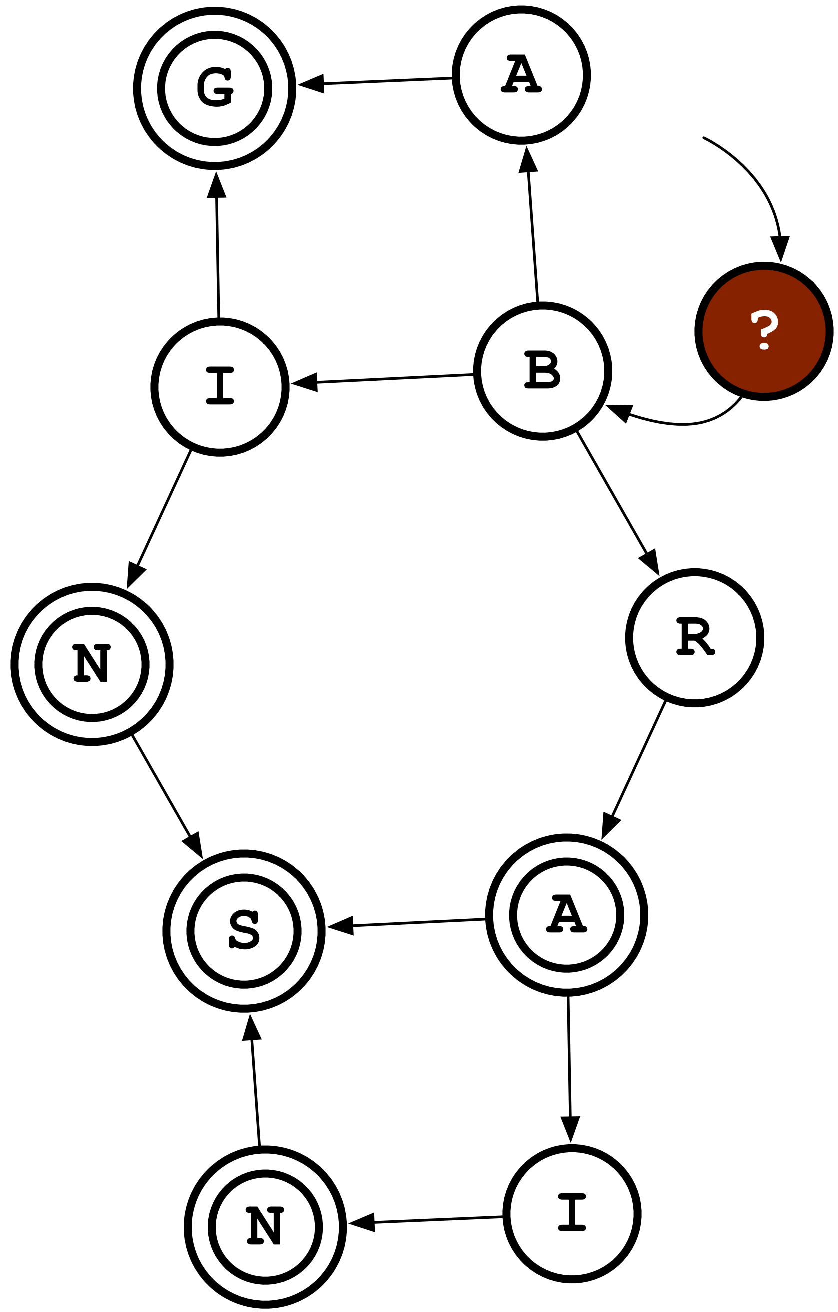
An **action** is a process that we run when we make a transition. There are three kinds: *exit, transition, and enter*. We do the exit and enter actions when we exit and enter states. This is independent of the transition.



This is a FSM that parses a limited set of Latin characters. We receive input and transition among states.

There is no persistent memory besides the current state.

This is one of the interesting and practical aspects of FSMs. They require an extremely small amount of memory, to the point where we can implement them with minimal hardware (e.g. analog circuits).

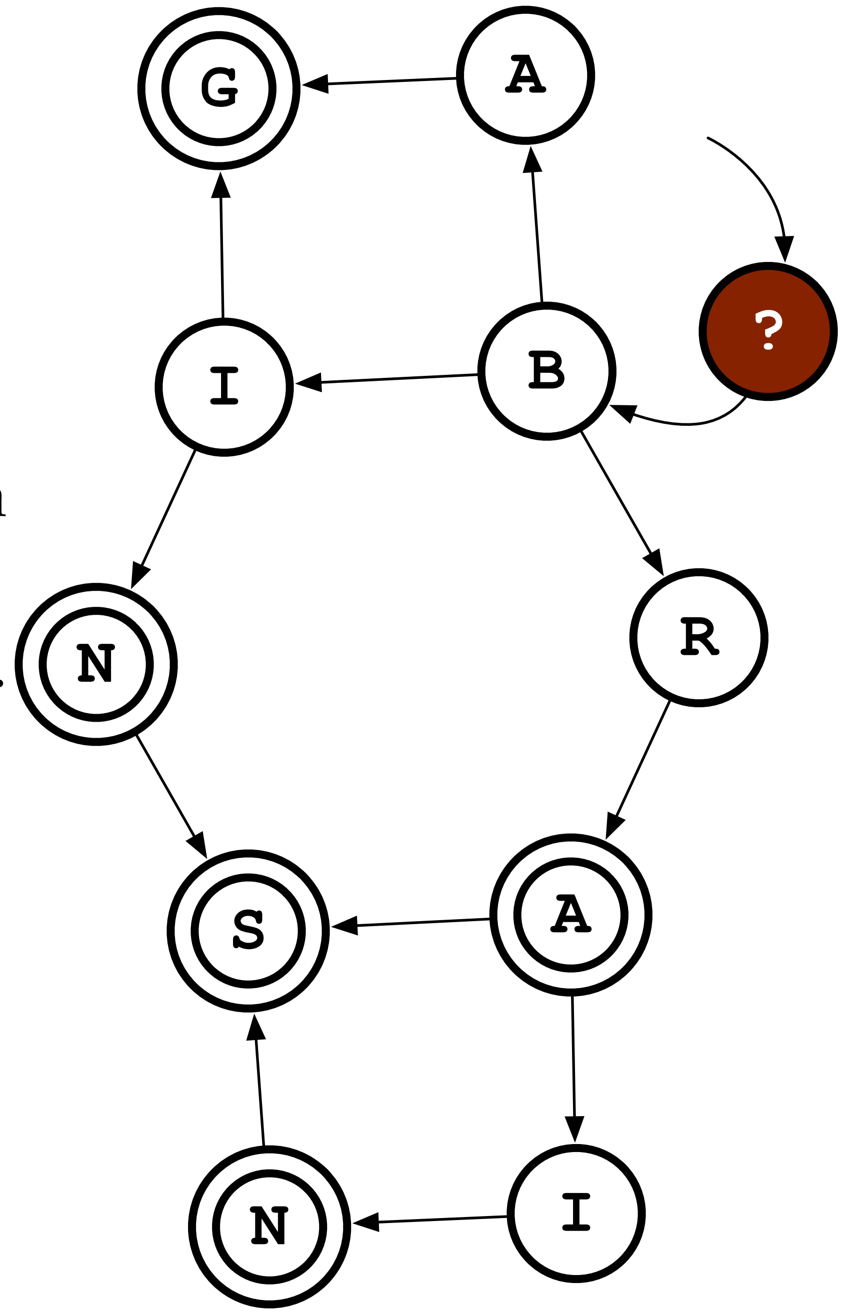


There is no persistent memory besides the current state.

This means we can compute a FSM's state without remembering anything. We could have received a billion signals in the past, and our FSM would still be chugging along.

So what's this FSM good for? It can transition based on reading characters, but what's it tell us?

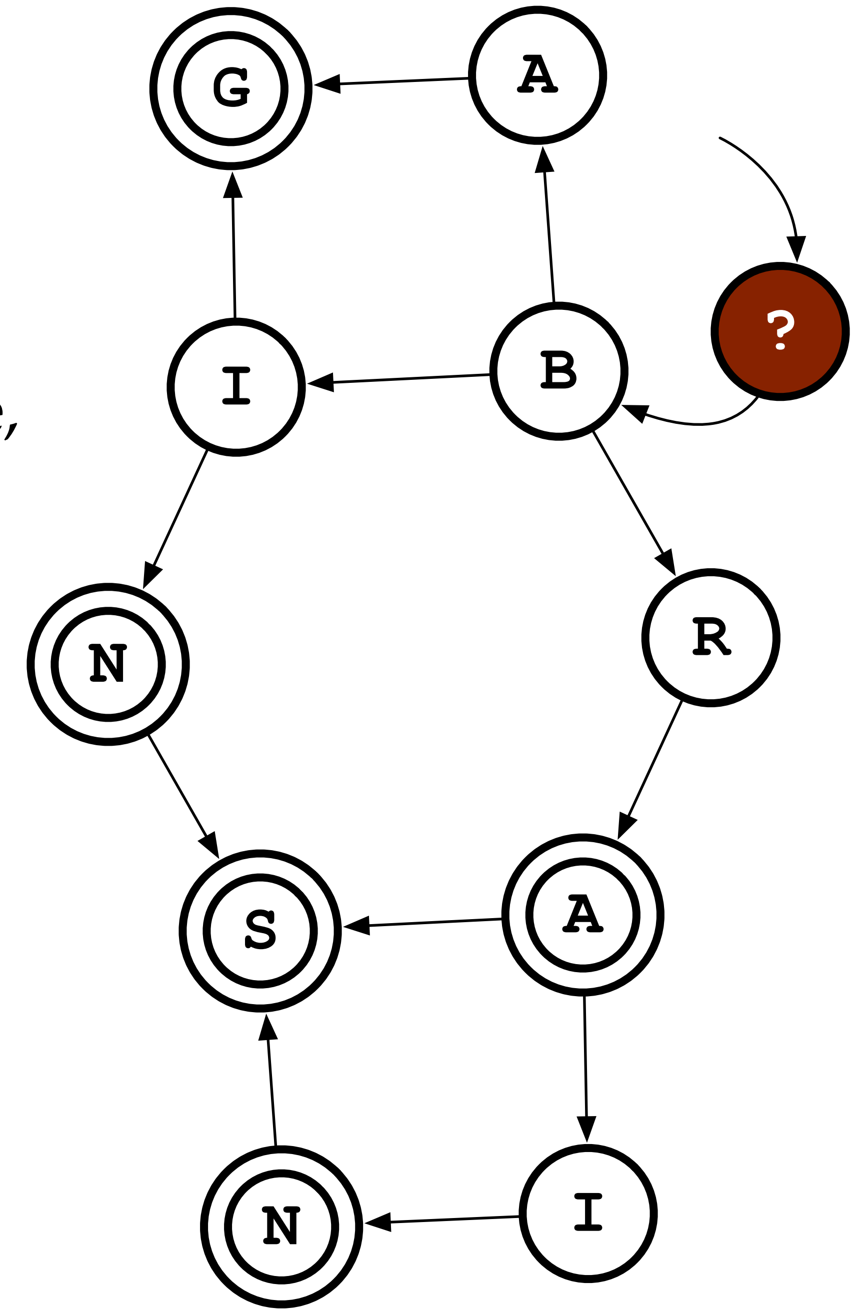
We can augment it such that states are either *accepting* or not.



Accepting states are indicated with the double circle. If we are in an accepting state, that means the input we have processed thus far matches some interesting sequence, such as a correctly spelled word.

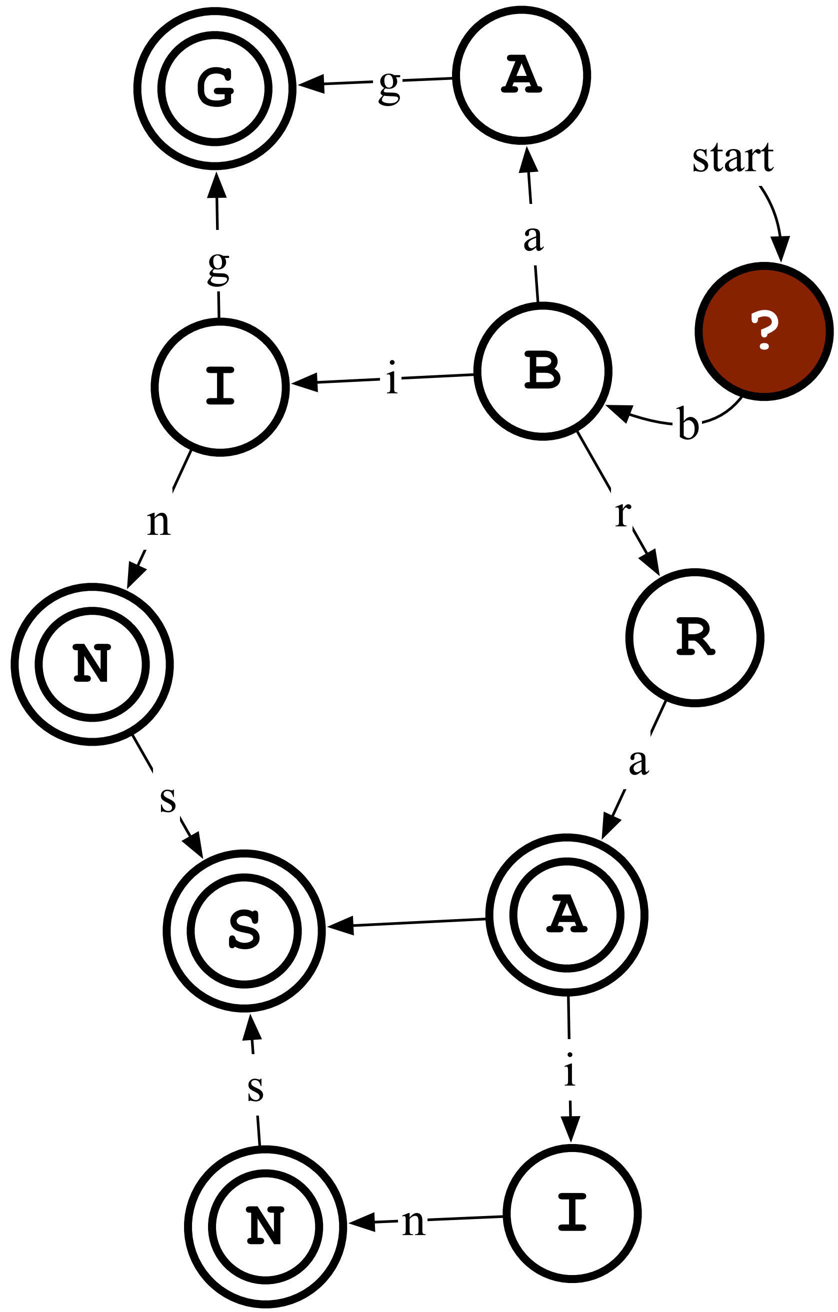
If the FSM is *not* in an accepting state it means we haven't (yet) found a correctly spelled word.

Lets look at some character sequence inputs to see what it does.



All these transitions have an associated event. They are all the lower case letter of the state they transition to. I'll put them on this slide, but I'll omit them from all the others.

They are still there, if only in our imaginations.

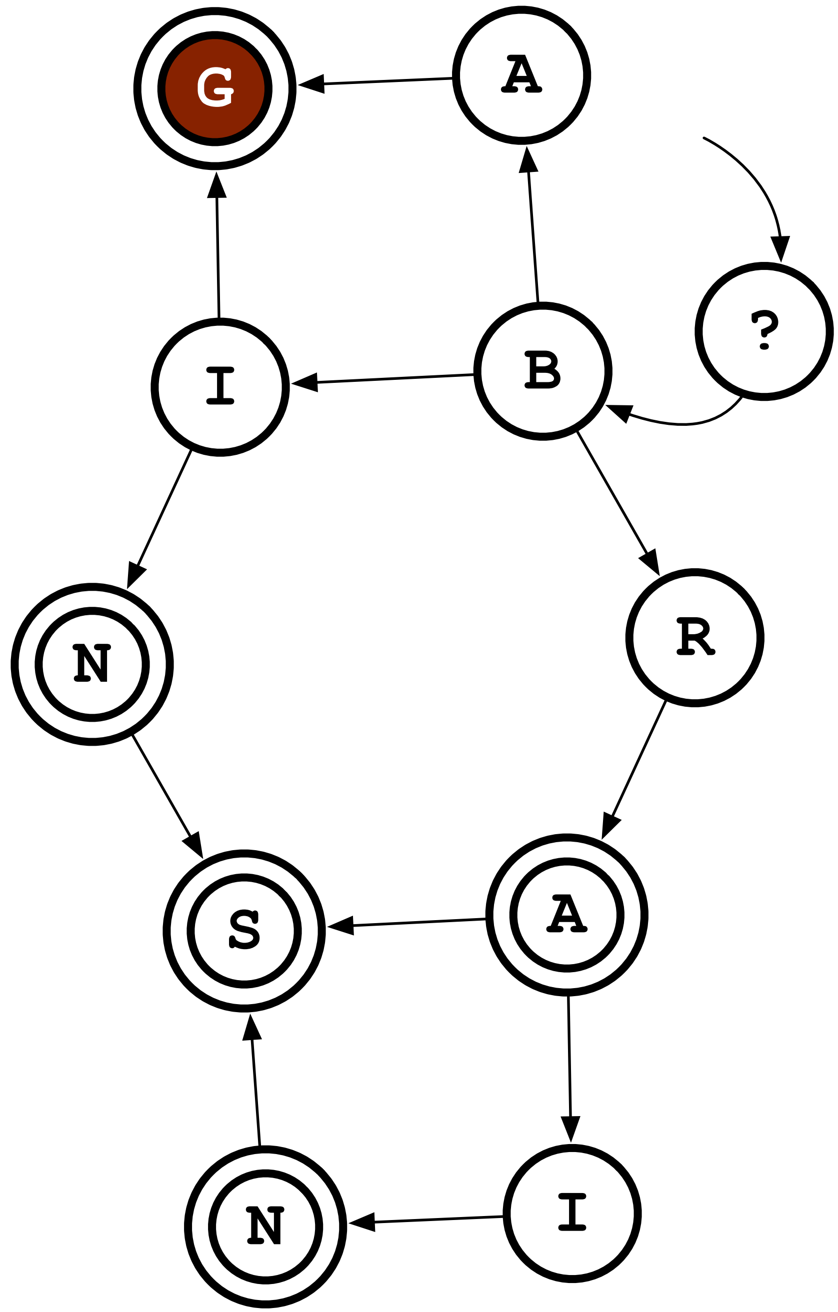


"big"

Starts from the default state (the question mark), receives b, then i, then g.

The end state G is an accepting state. This means it saw a correct word. Lets do another one.

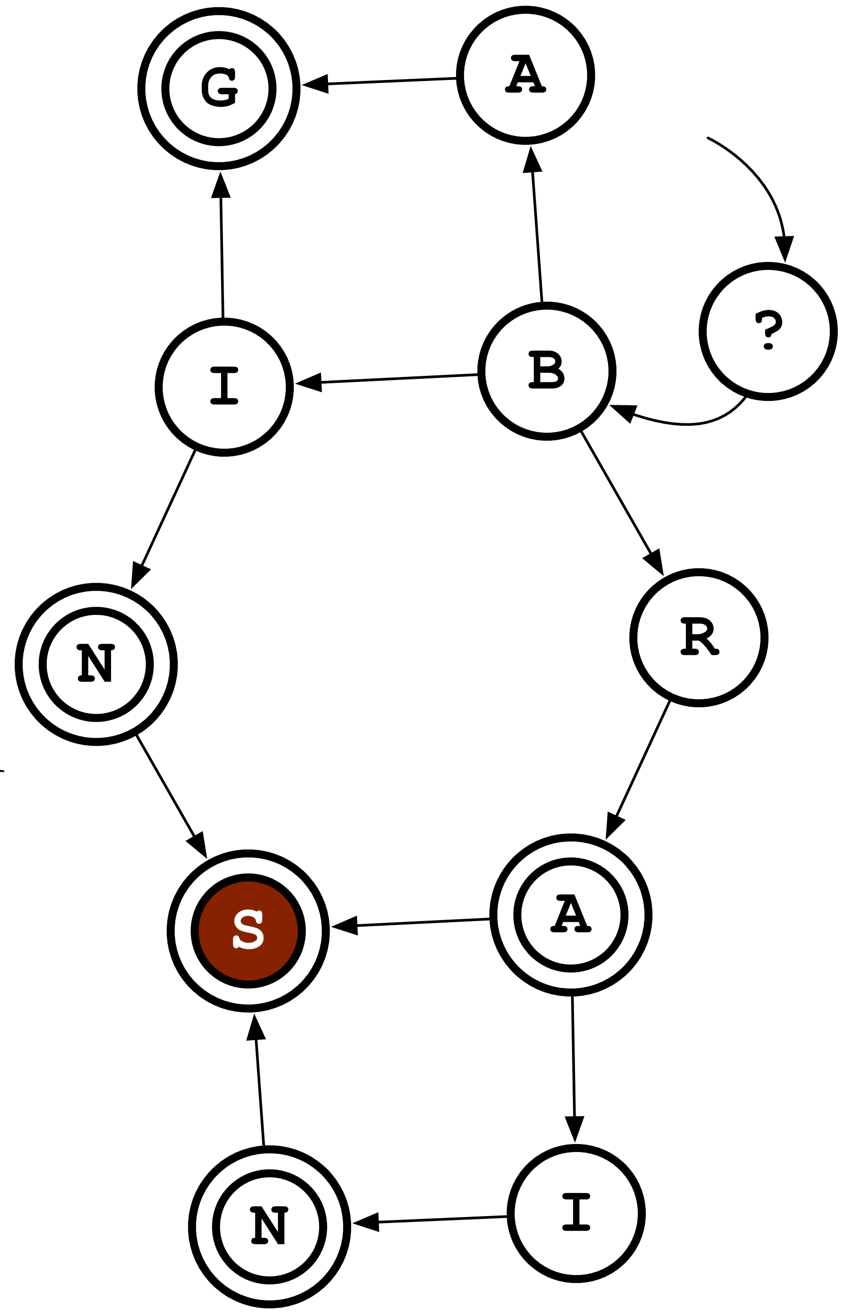
(Reset the FSM to the default state, indicated with the arrow from nowhere. It points at the ? node.)



"bins"

Again we end up at an accepting state, which means we know how to spell three- and four-letter words. Yay us!

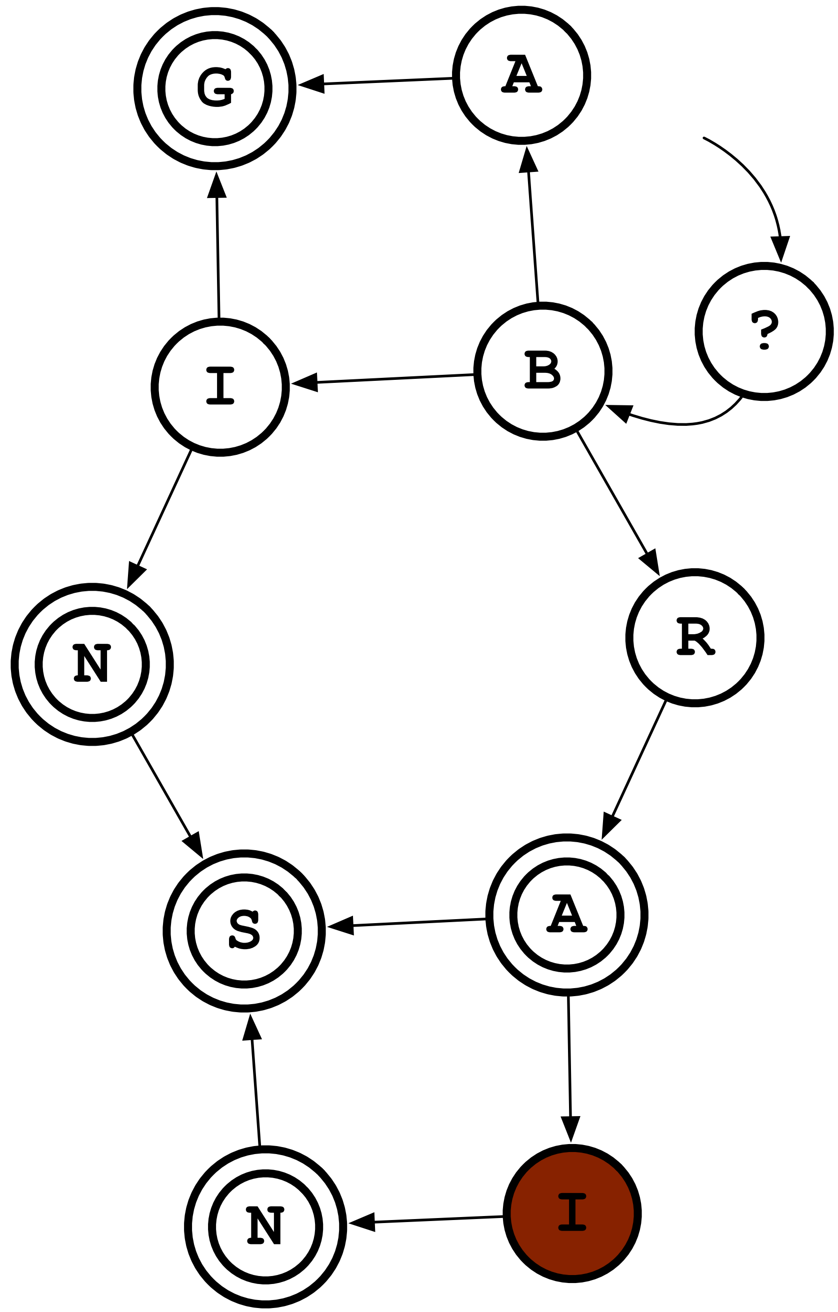
But what if we send it weird input? Right now the FSM is not equipped to do anything with signals it doesn't understand. Often this is what we want.



"braid"

(Starting from the ? again)

We are able to follow b, r, a, i, to leave us at state *I*, but then we receive the signal *d*, which doesn't correspond to any transition in the FSM. So in this case, the current state remains *I* (and we are in a non-accepting state, though this is coincidental).

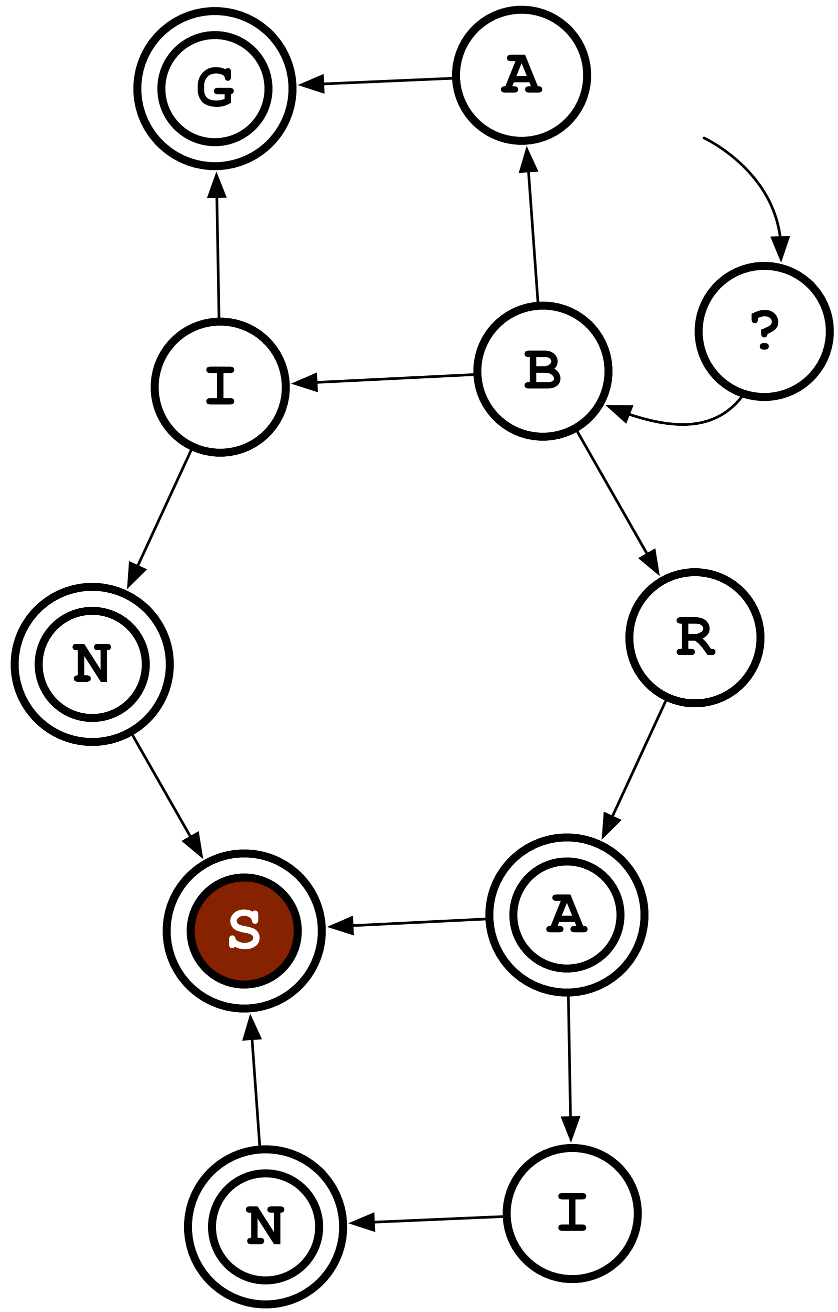


"braidings"

(Starting from the ? again)

Every time we receive a signal that we can act on, we do. If we receive a signal that doesn't make sense for the current state, we do nothing. Given this input string, we end up in the accepting state *S*.

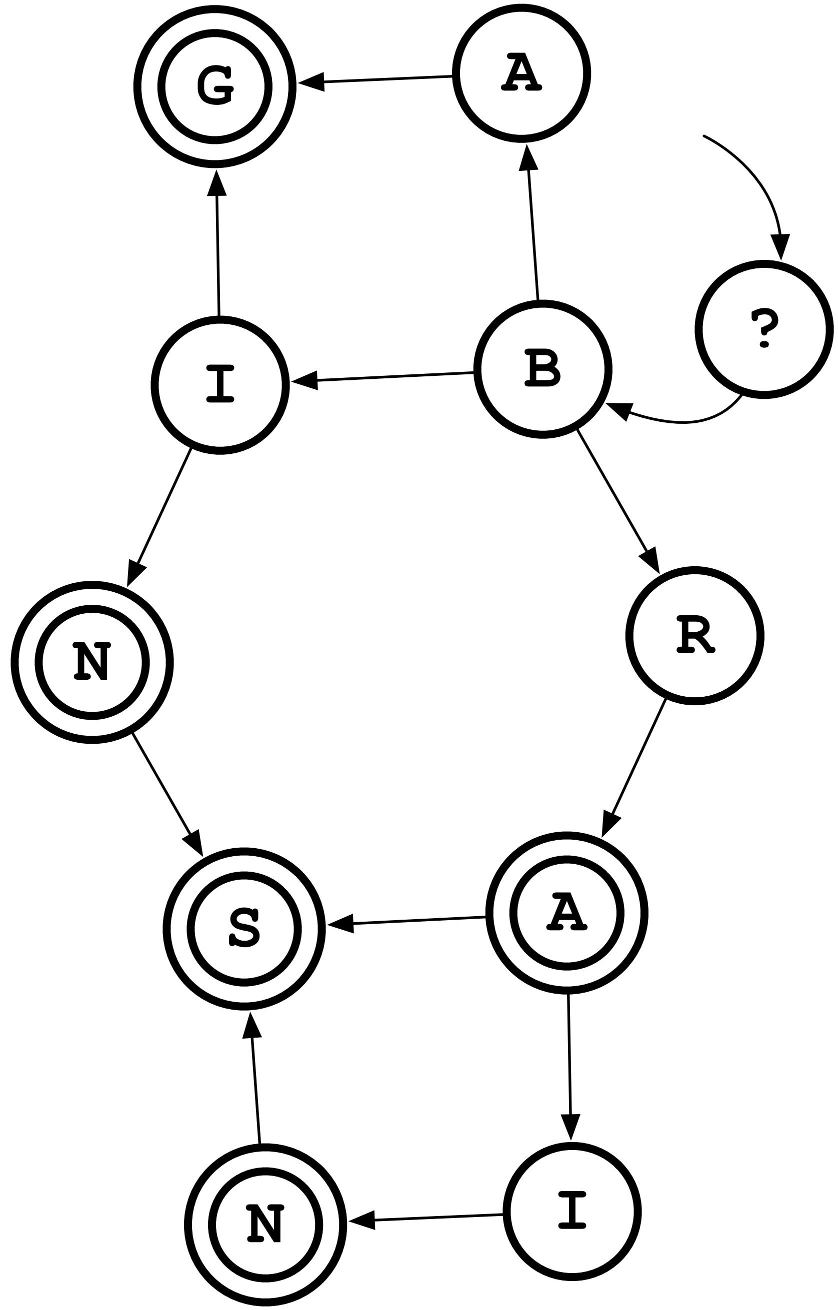
Obviously this would be a pretty terrible spell checker. Let's fix that.



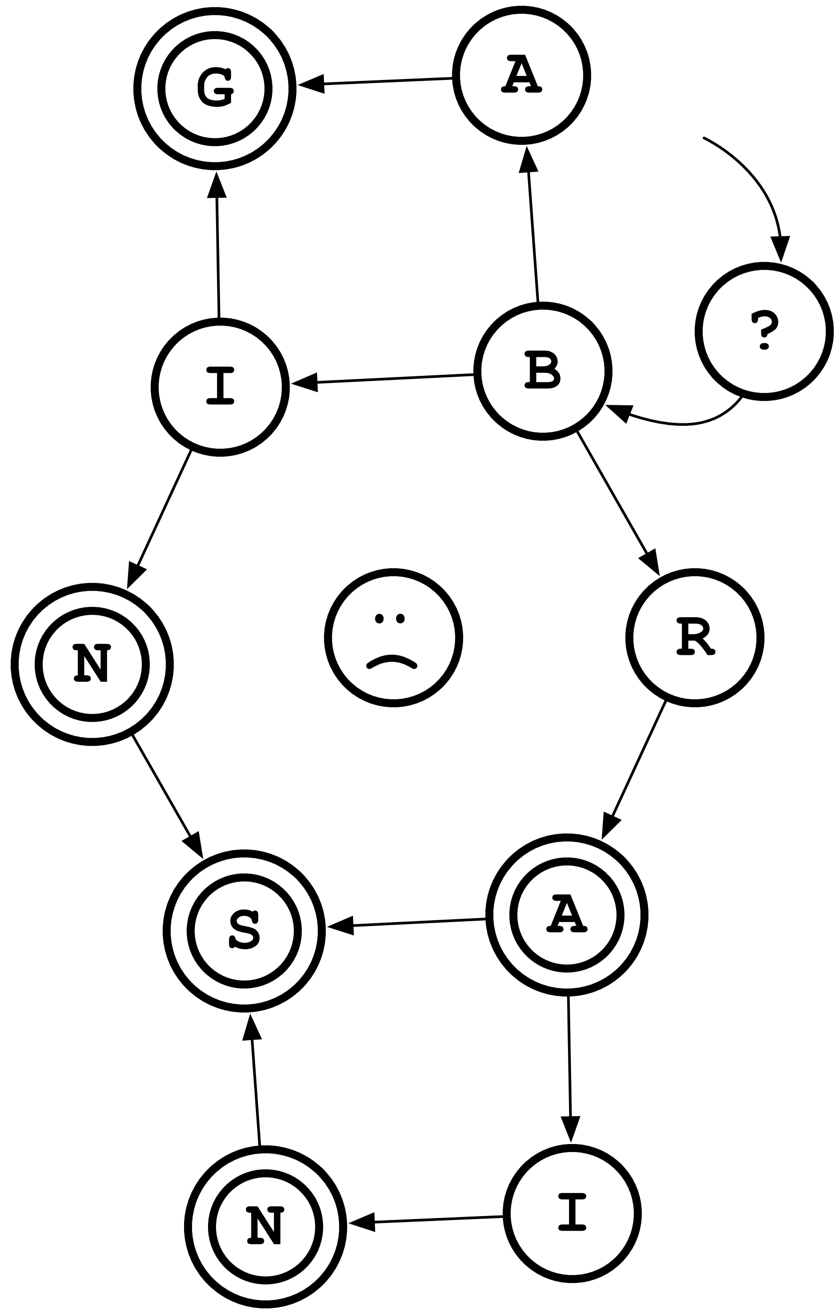
We will create a 'failure' state that we can transition to whenever a node receives something that it does *not* expect.

Two things going on here.

First: we need to make a new kind of transition that is triggered when the signal we receive is not associated with any other transition. This is a *failure* transition.

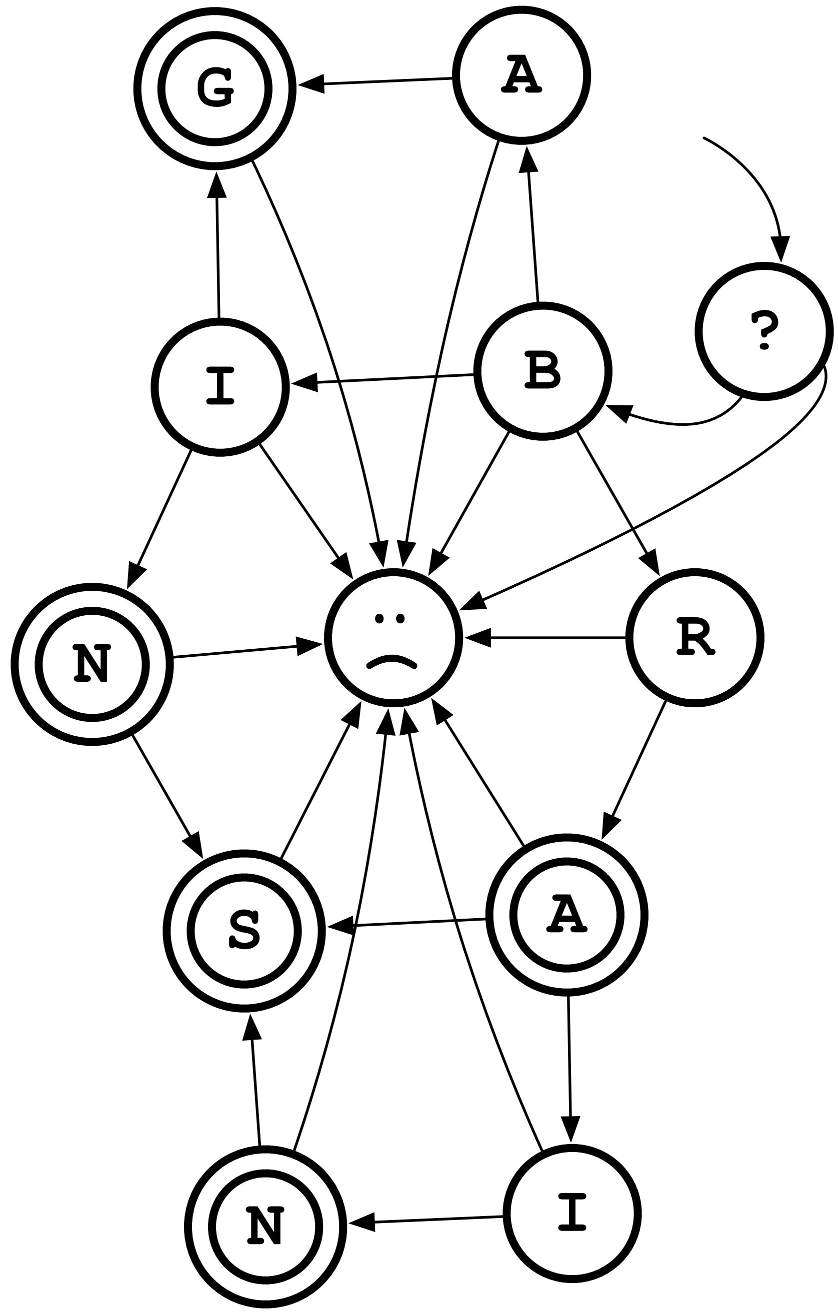


Second: we need to make a *failure* state that we transition to whenever we receive a signal that doesn't match any of our current state's transitions.



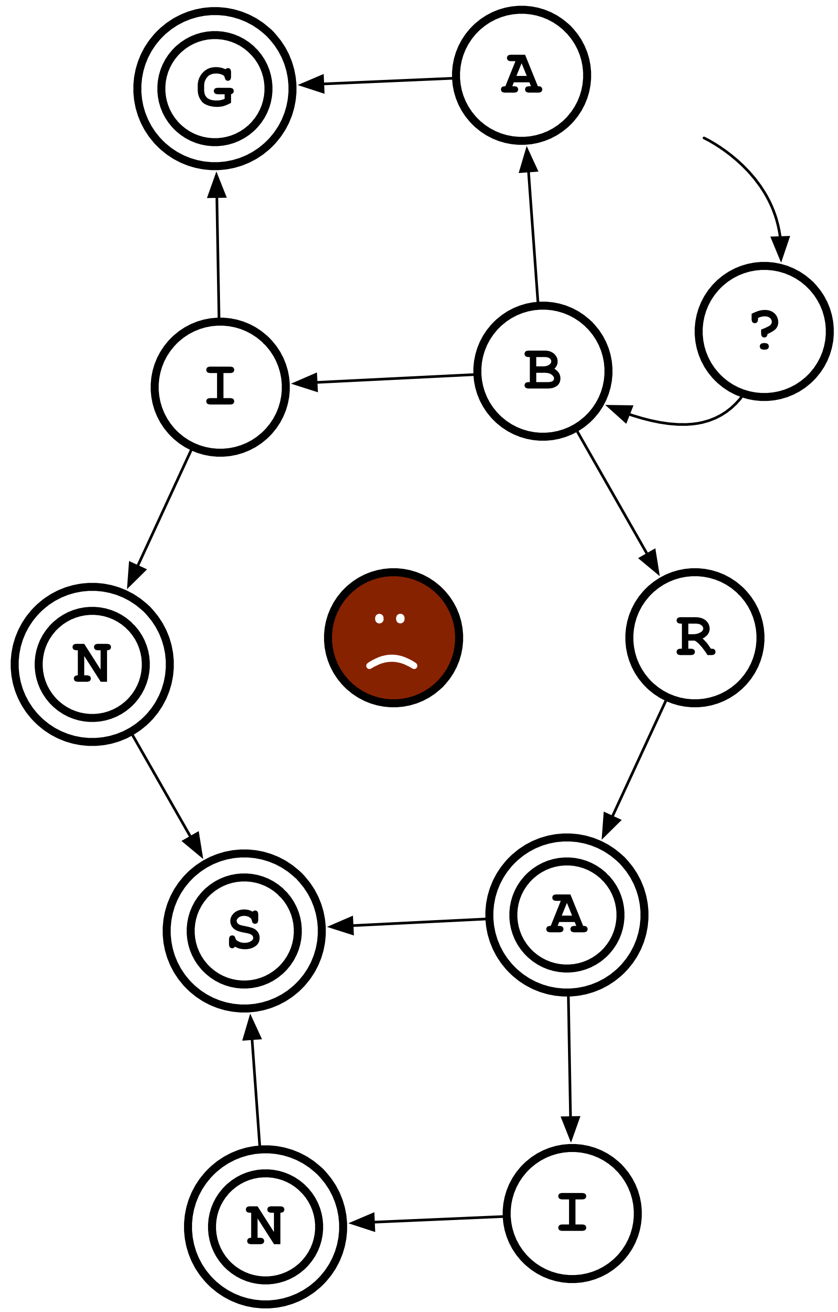
If we actually show all of those failure edges, we have a linguini-like FSM. This is hard to read so I won't draw those edges.

But, like the signal labels on the transitions, the failure edges are still there in our imaginations.



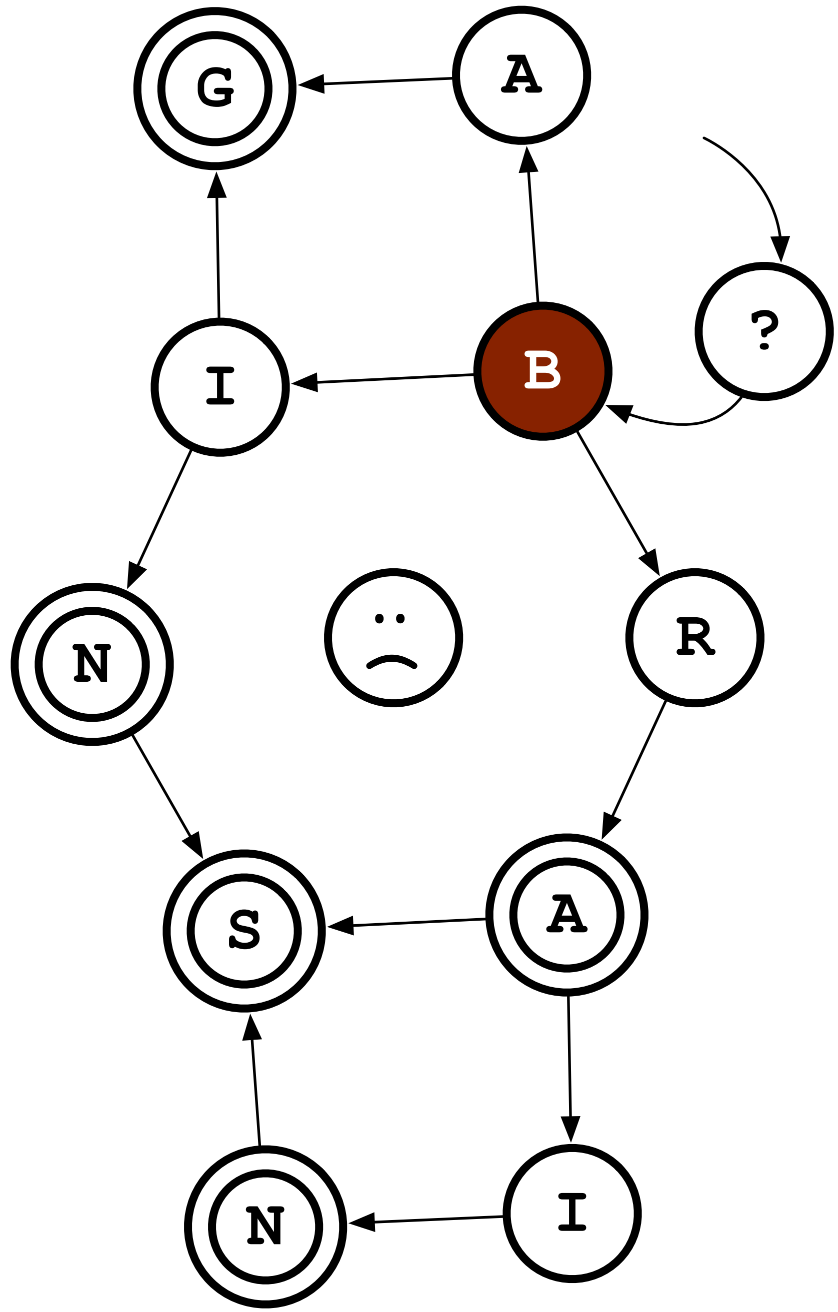
So now if we receive strange input like "strange input", the FSM immediately enters the failure state drawn with Mr. Unhappy there.

So lets watch as we type a word. What state are we in? Is it an accepting state?



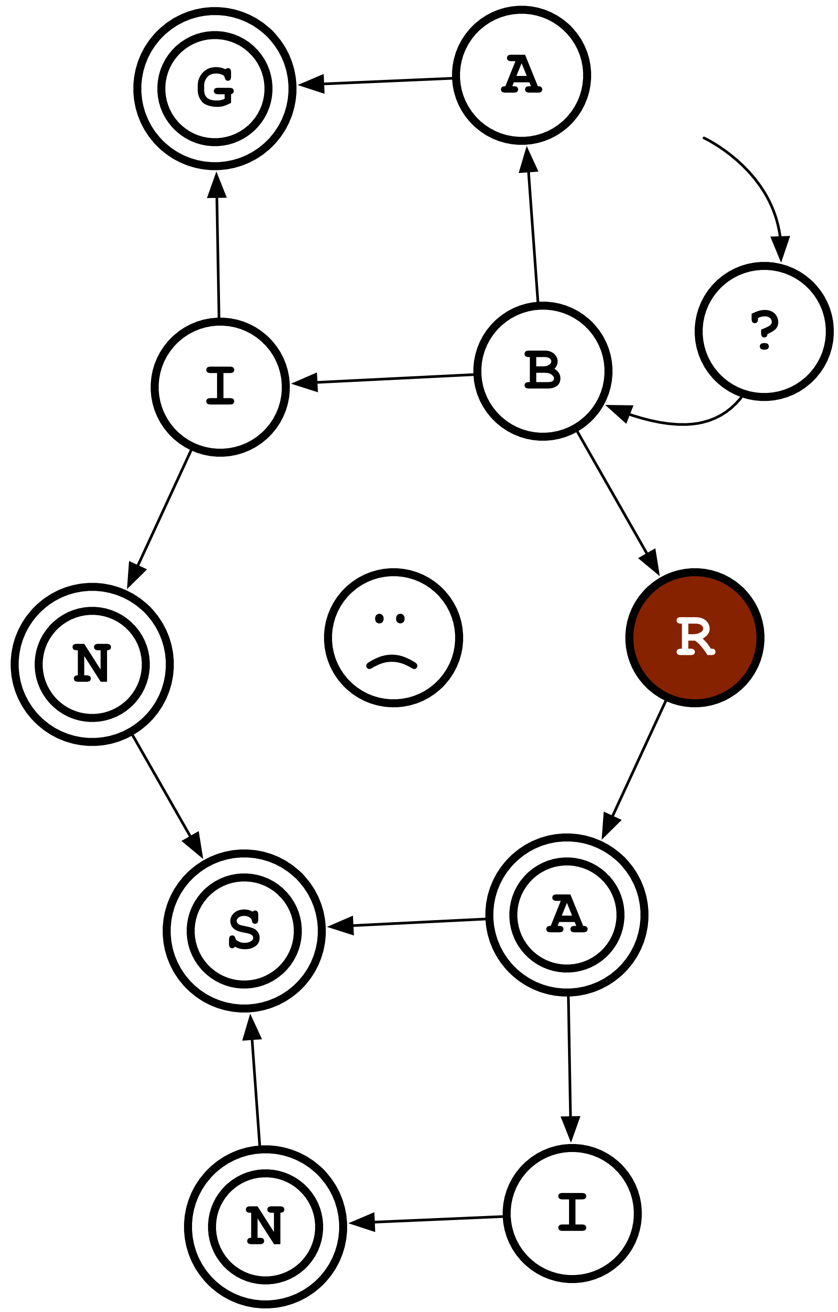
B

✕ **wrong**



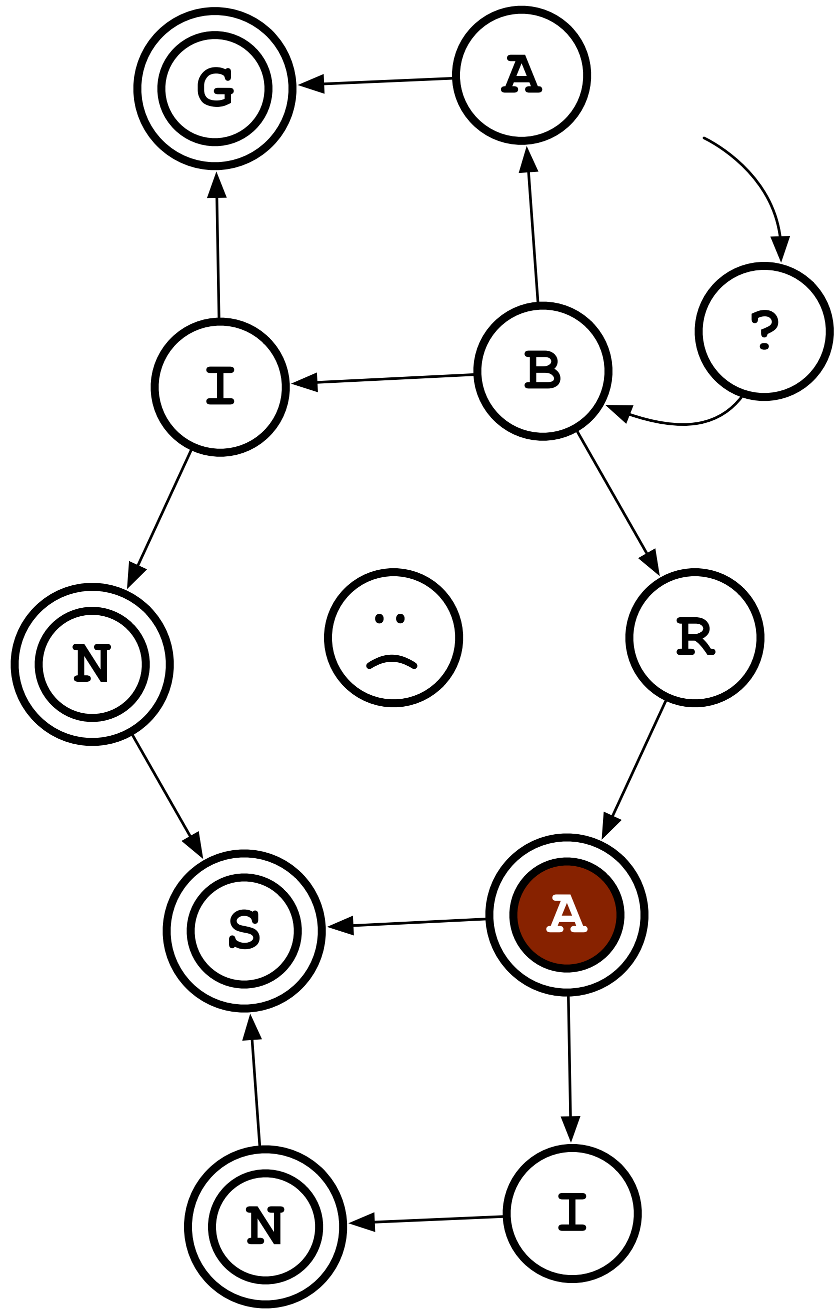
BR

✕ wrong



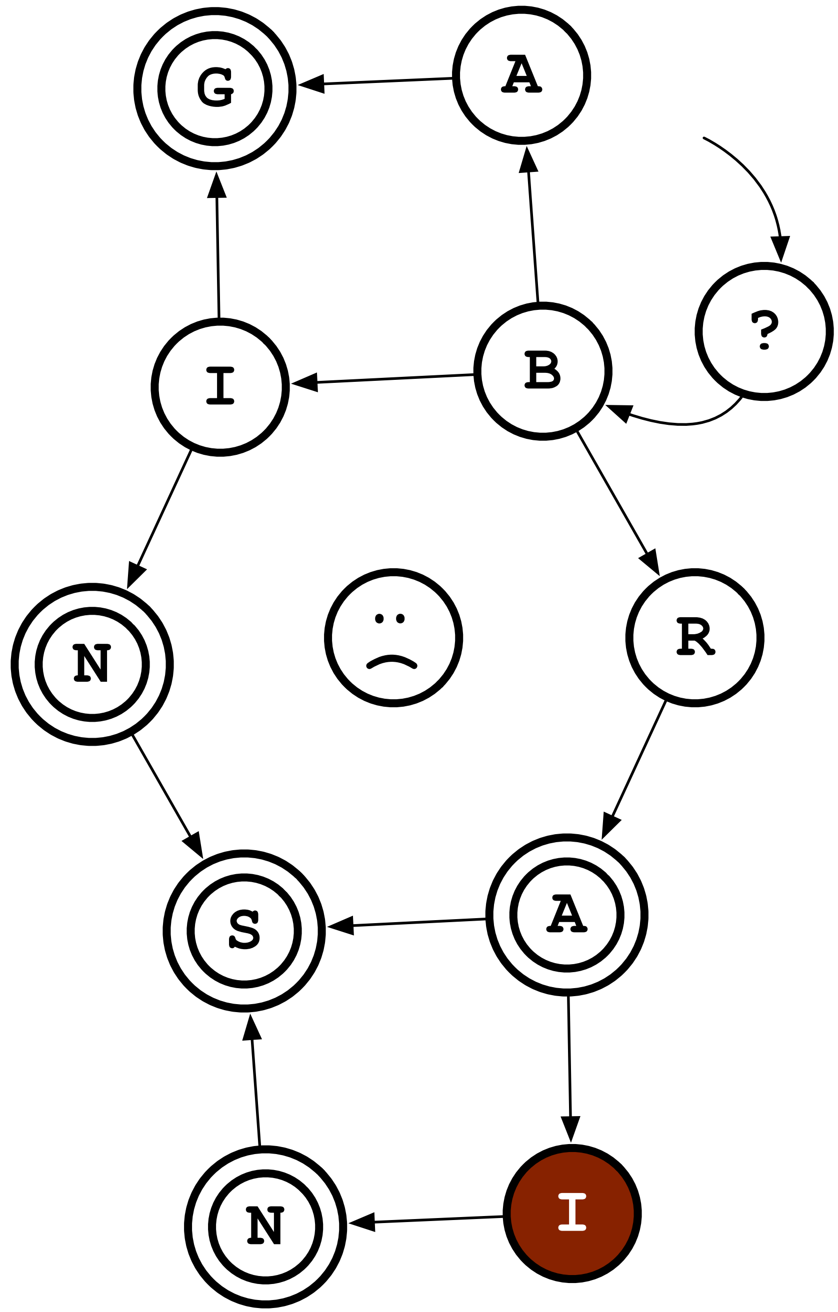
✓ right

BRA



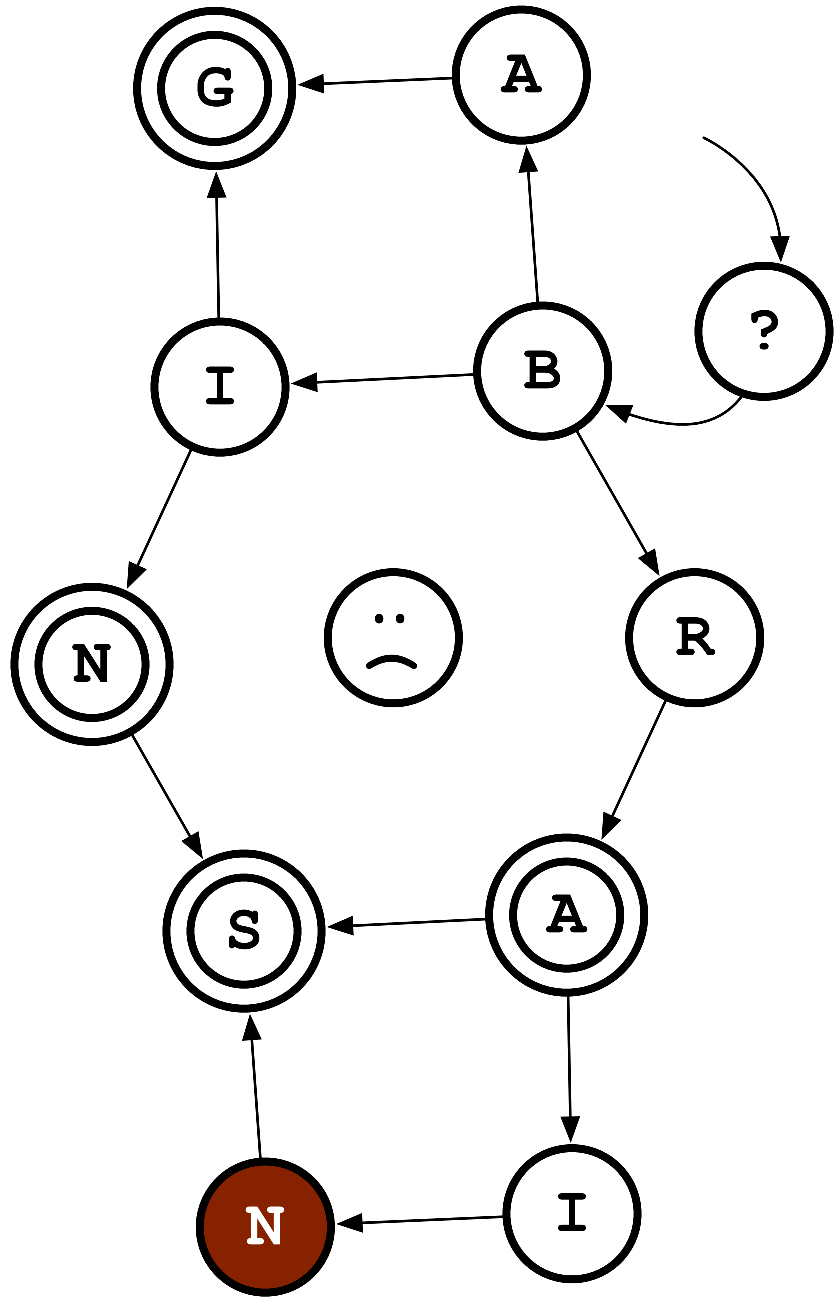
BRAI

✕ wrong



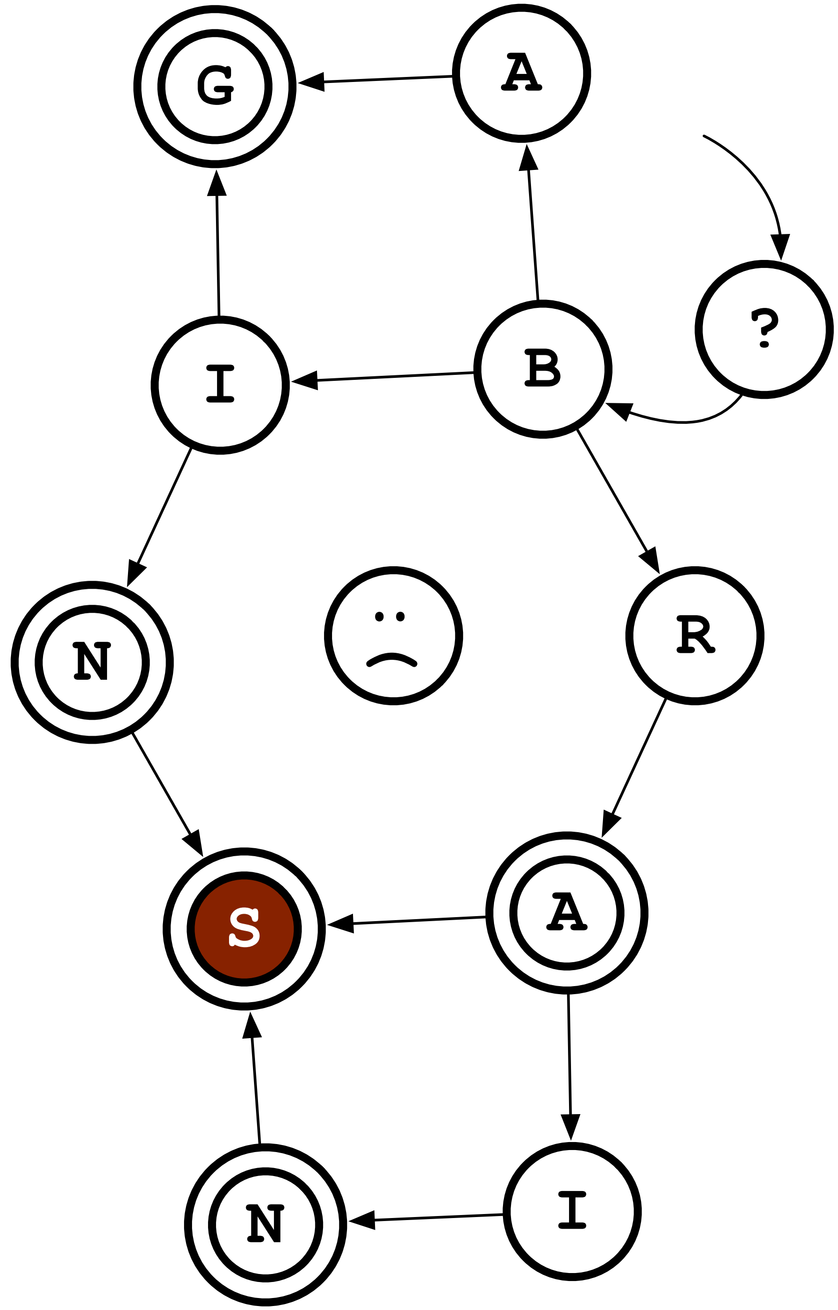
✓ right

BRAIN



✓ right

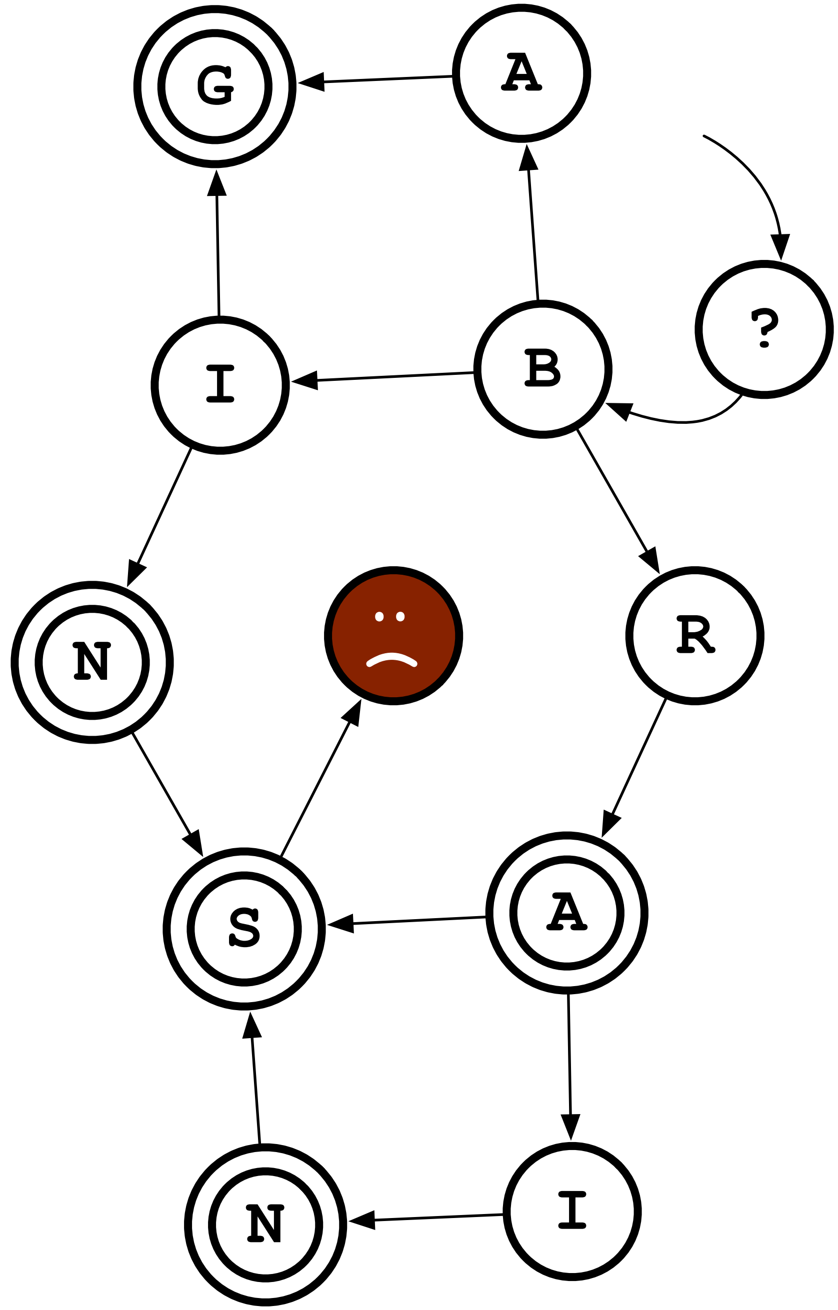
BRAINS



BRAINS

N

wrong



BRAINS NOM NOM

✗ wrong