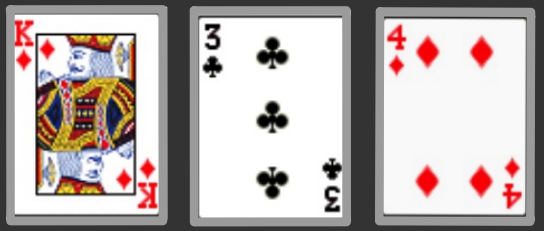
markov decision processes

CSCI 373

dealer

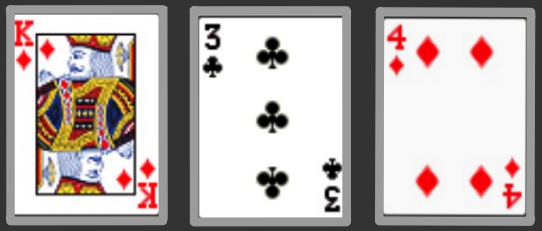
player



when we modeled blackjack with expectimax, we made several **assumptions**

dealer

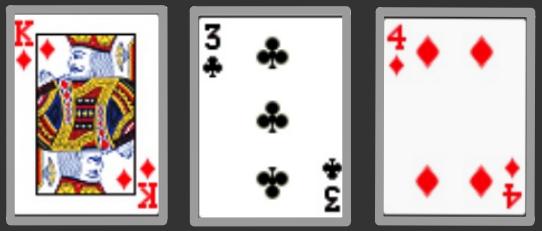
player



we ignored the dealer's "up card"

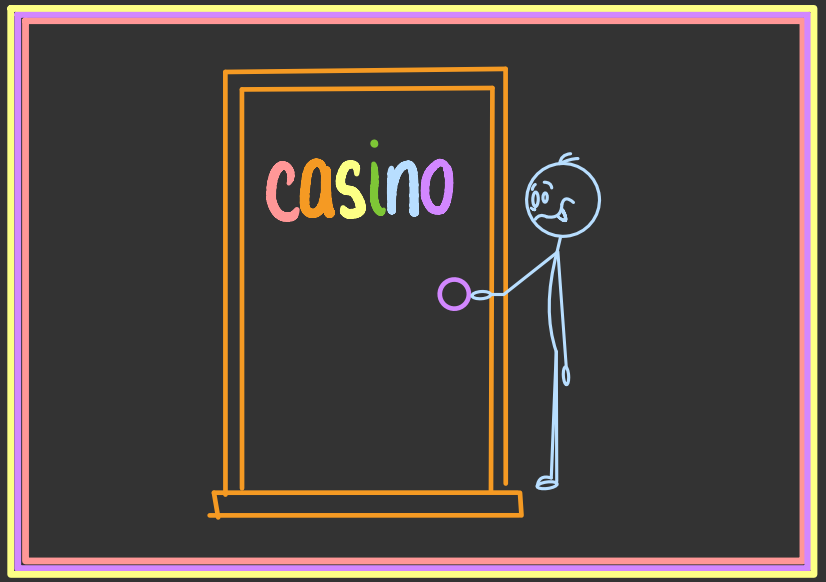
player's final total	probability of win	probability of draw	probability of loss	expected utility $1 \cdot P(\text{win}) + 0 \cdot P(\text{draw}) - 1 \cdot P(\text{loss})$
21	.88	.12	0	.88
20	.70	.18	.12	.58
19	.57	.13	.30	.27
18	.43	.14	.43	0
17	.28	.15	.57	-.29
16	.28	0	.72	-.44
bust	0	0	1	-1

player



we assumed we knew the expected payout for any card total

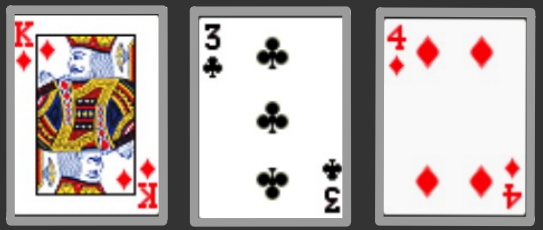
we want to relax
these assumptions



ideally we'd like to walk into a casino, knowing nothing, and learn to play blackjack well

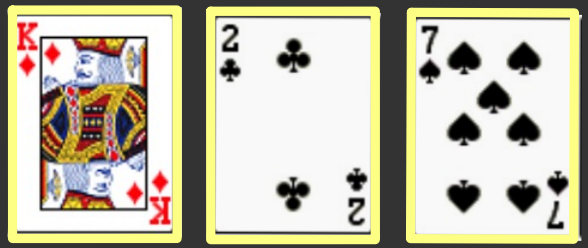
player's final total	probability of win	draw	loss	expected utility $1 \cdot P(\text{win}) + 0 \cdot P(\text{draw}) - 1 \cdot P(\text{loss})$
21	.88	.12	0	.88
20	.70	.18	.12	.58
19	.57	.13	.30	.27
18	.43	.14	.43	0
17	.28	.15	.57	-.29
16	.28	0	.72	-.44
bust	0	0	1	-1

player



but before we do that, let's restore our previous assumptions and use them in the context of a model called a markov decision process

first, build a state machine describing what could happen if we have a card total of 19

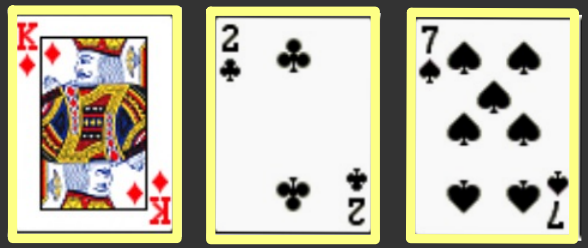


state machine

short for "active"

state A19 describes our current state:

a card total of 19 and we're still playing

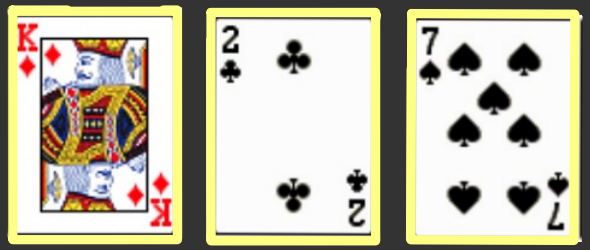


A19

state machine

if we stand in state A19, we move to state S19

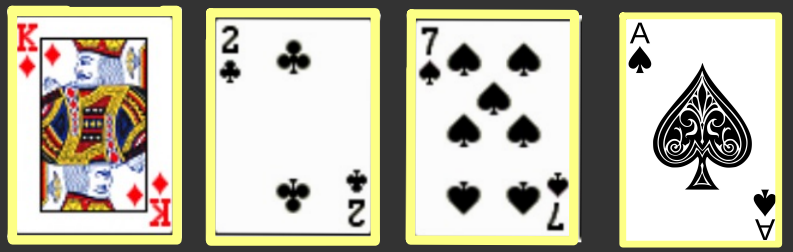
↖ short for "stood with 19"

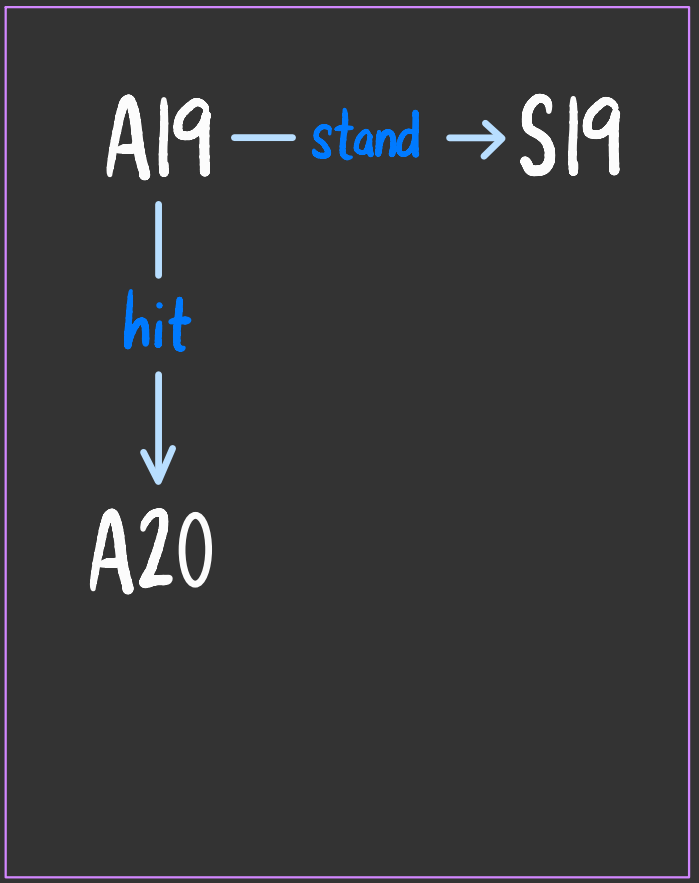


A19 — stand → S19

state machine

if we hit in state A19, we move to state A20 if we draw an ace



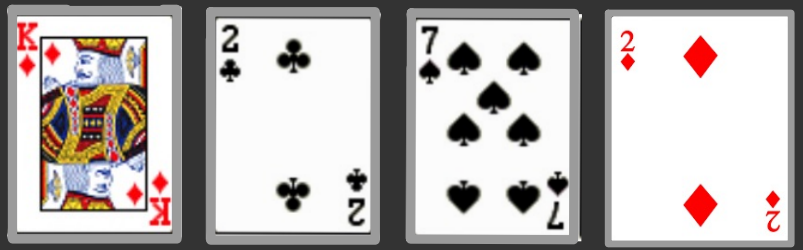


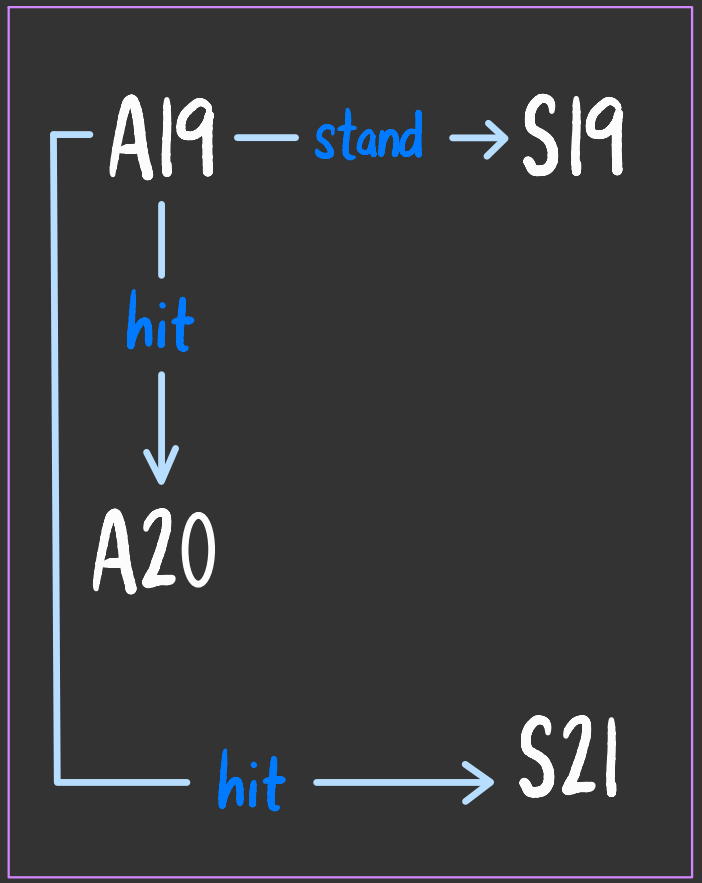
state machine

if we hit in state A19, we move to state S21

we stand automatically at 21

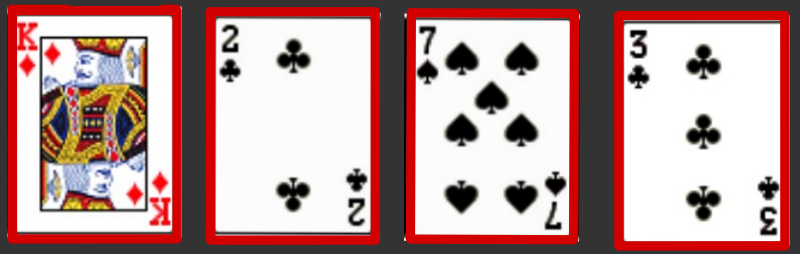
if we draw a two

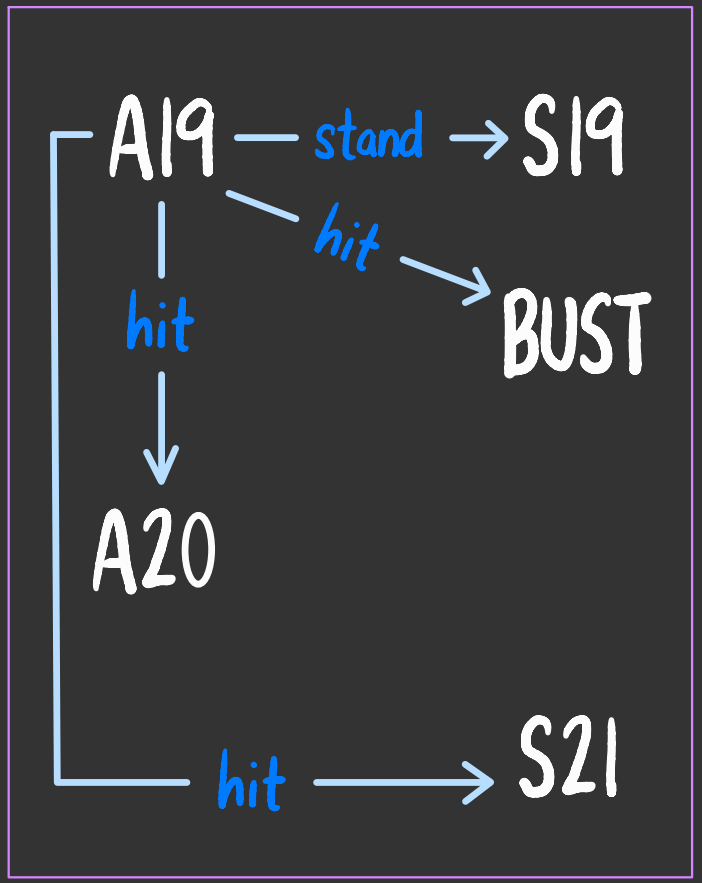




state machine

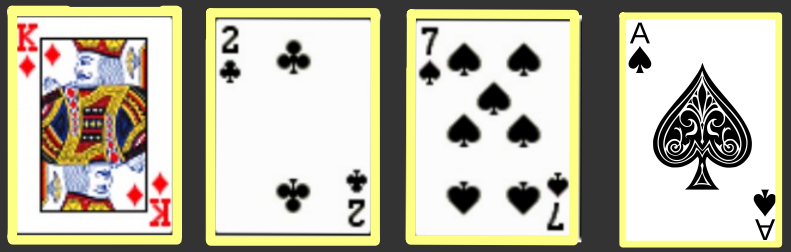
if we hit in state A19, we move to state BUST if we draw anything else

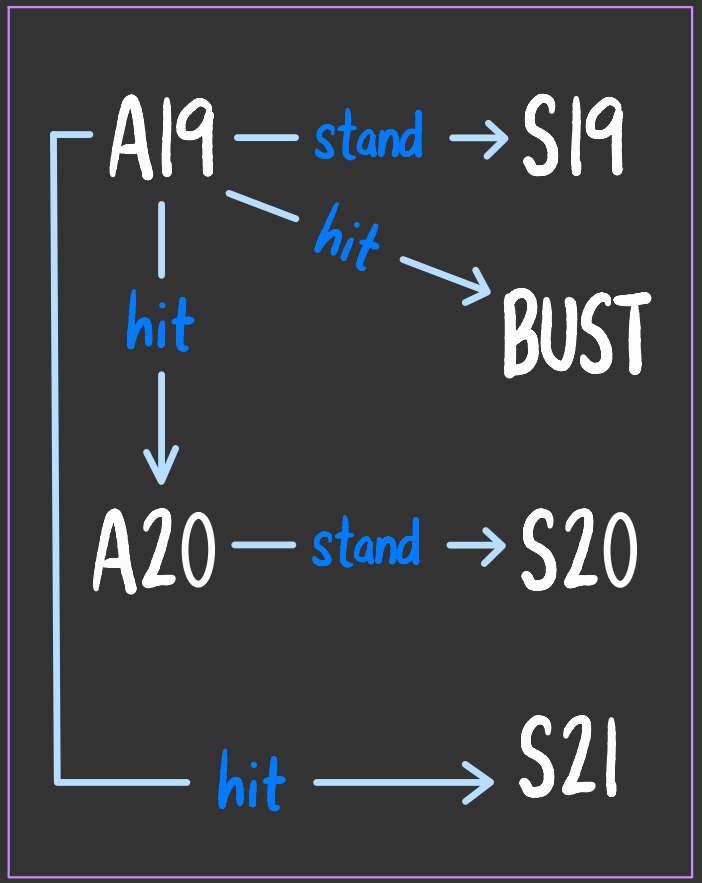




state machine

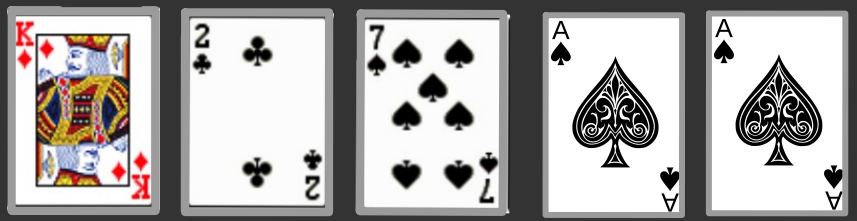
if we stand in state A20, we move to state S20

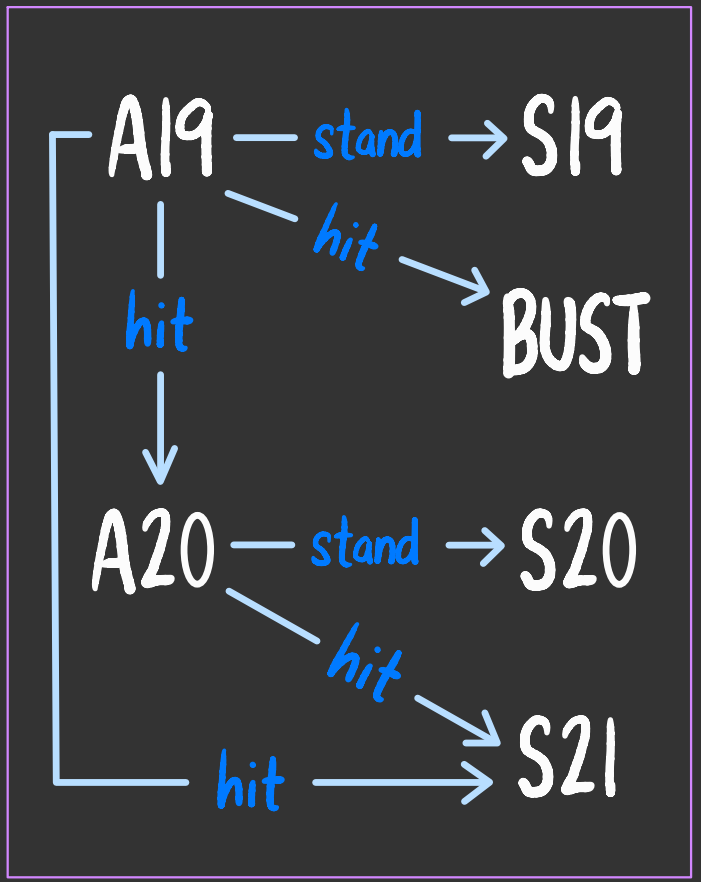




state machine

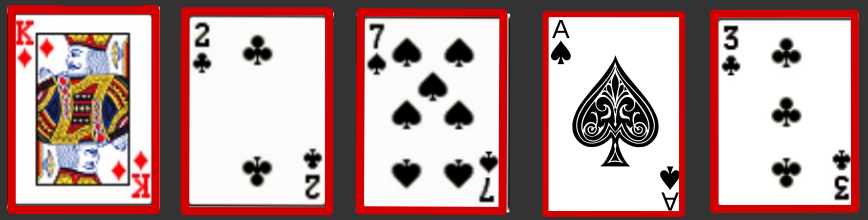
if we hit in state A20, we move to state S21 if we draw an ace

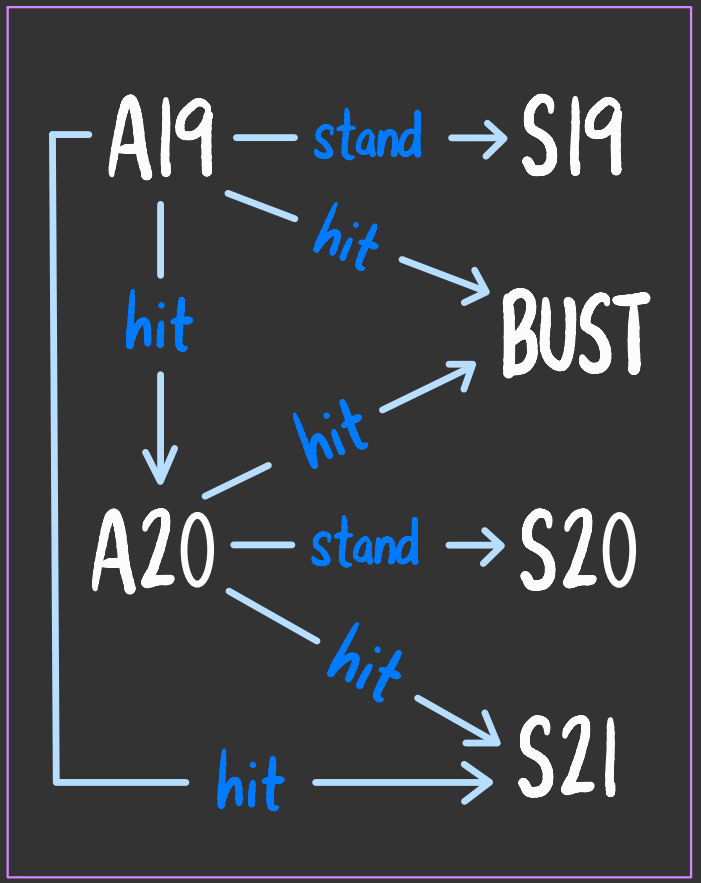




state machine

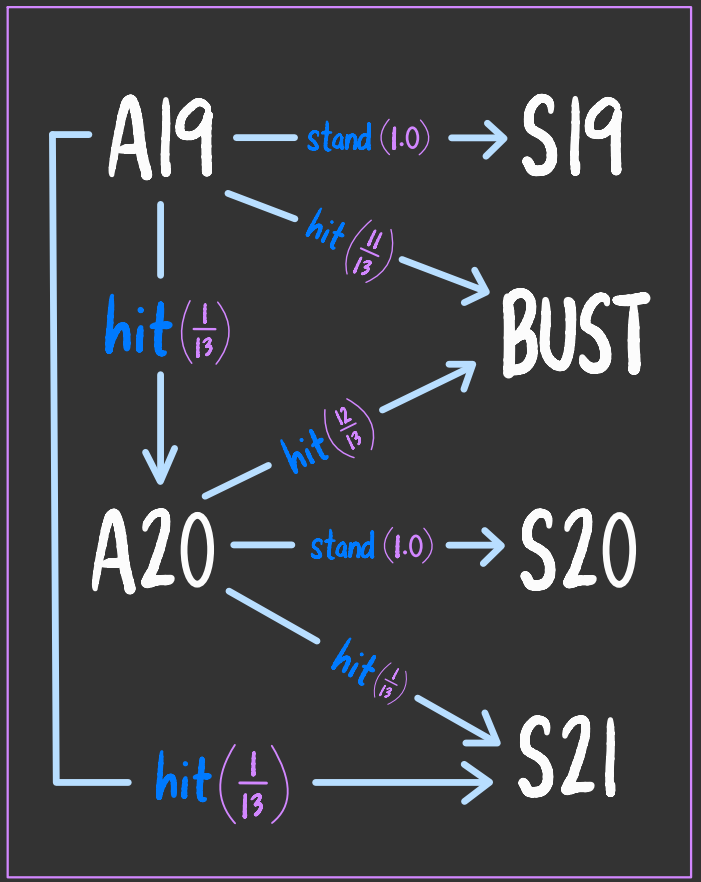
if we hit in state A20, we move to state BUST if we draw anything else



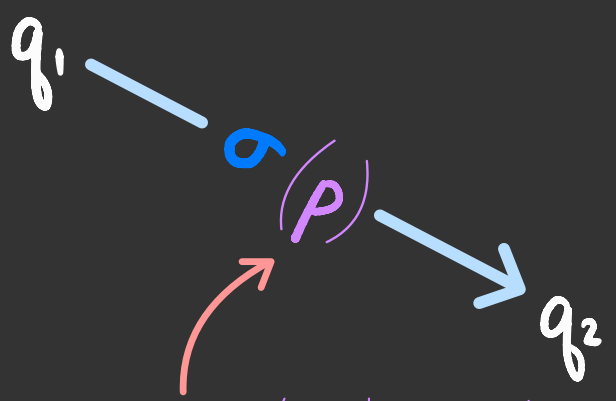


state machine

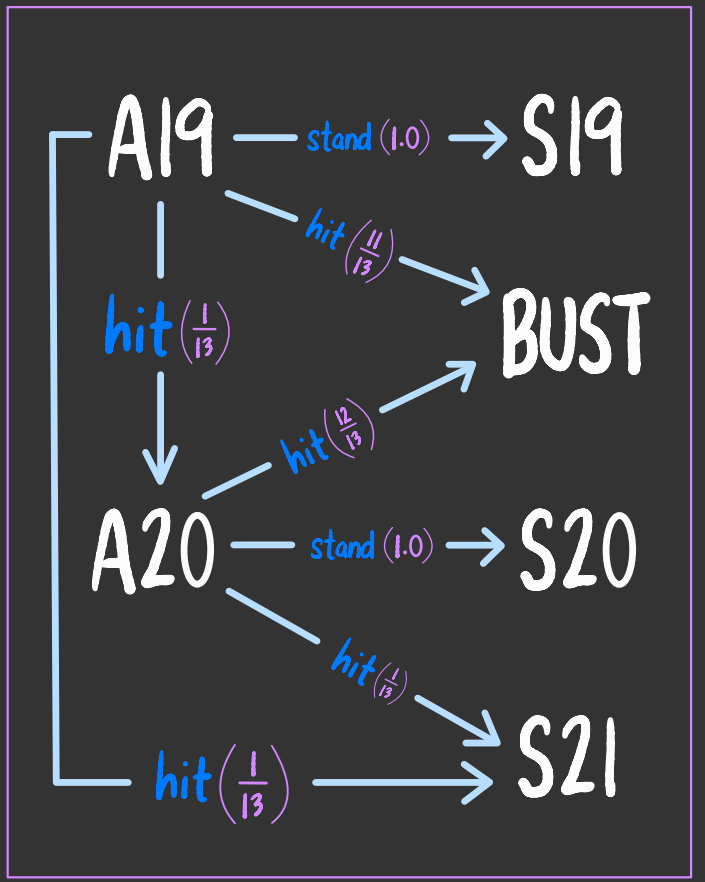
we will annotate each transition with a probability



state machine

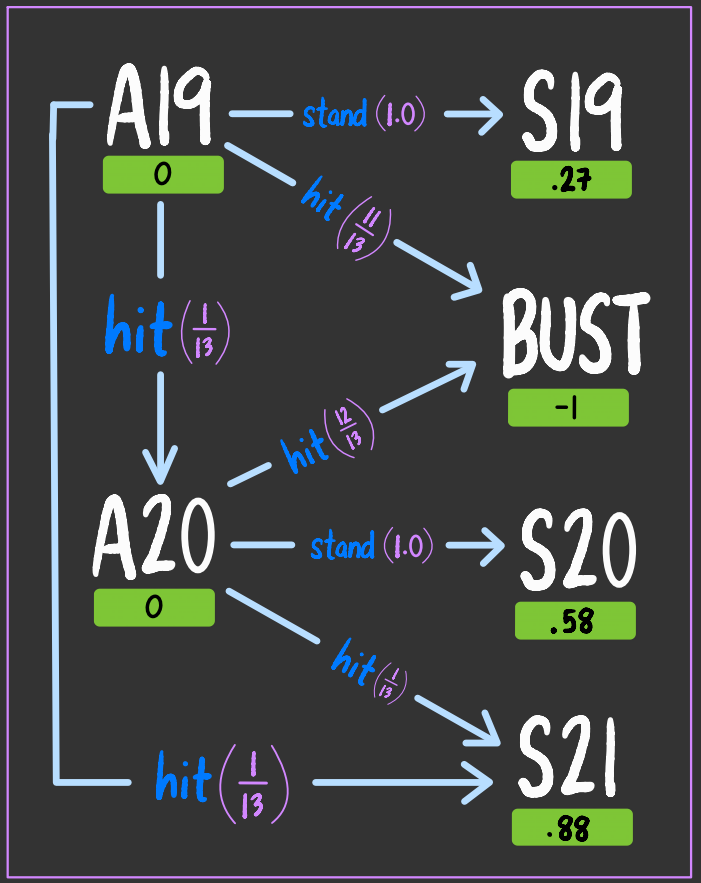


this is $P(q_2 \mid q_1, \sigma)$, the probability of going to state q_2 if we do action σ in state q_1



state machine

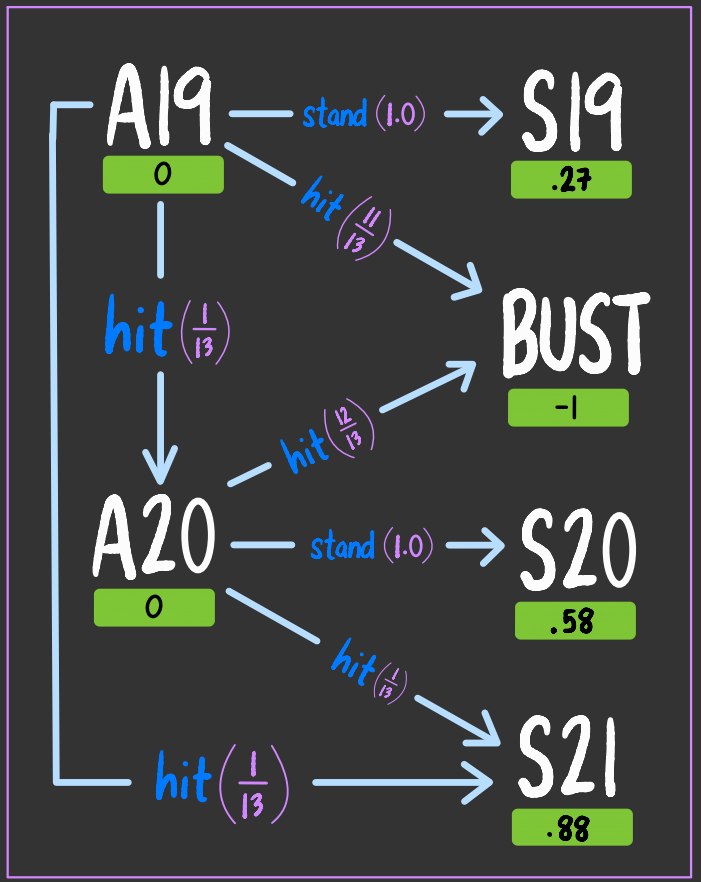
finally we will associate each state with a **reward**



state machine

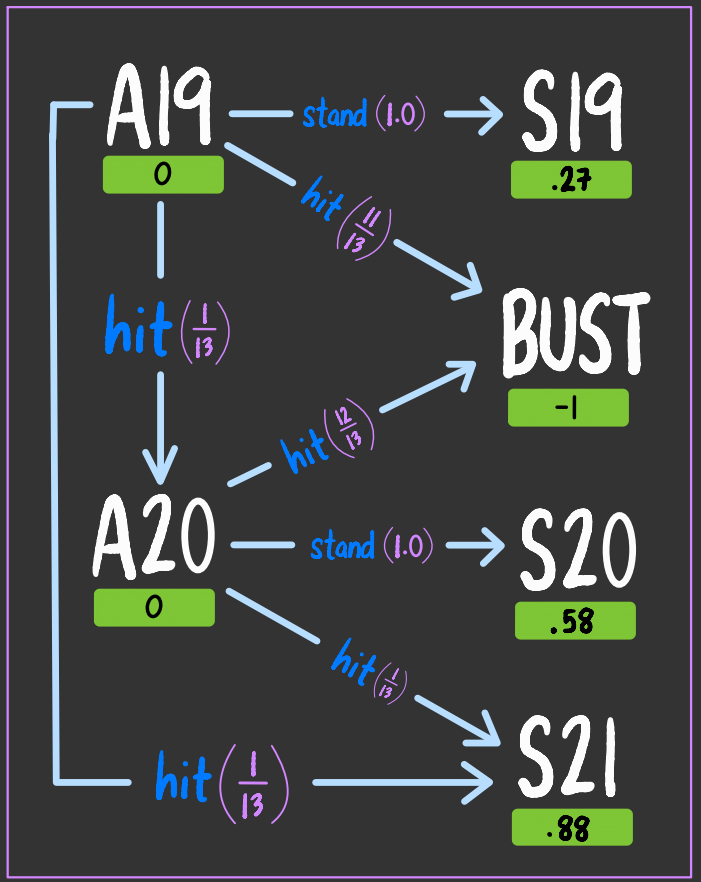
for now we'll just use our table of expected utilities

player's final total	expected utility $1 \cdot P(\text{win}) + 0 \cdot P(\text{draw}) - 1 \cdot P(\text{loss})$
21	.88
20	.58
19	.27
bust	-1



state machine

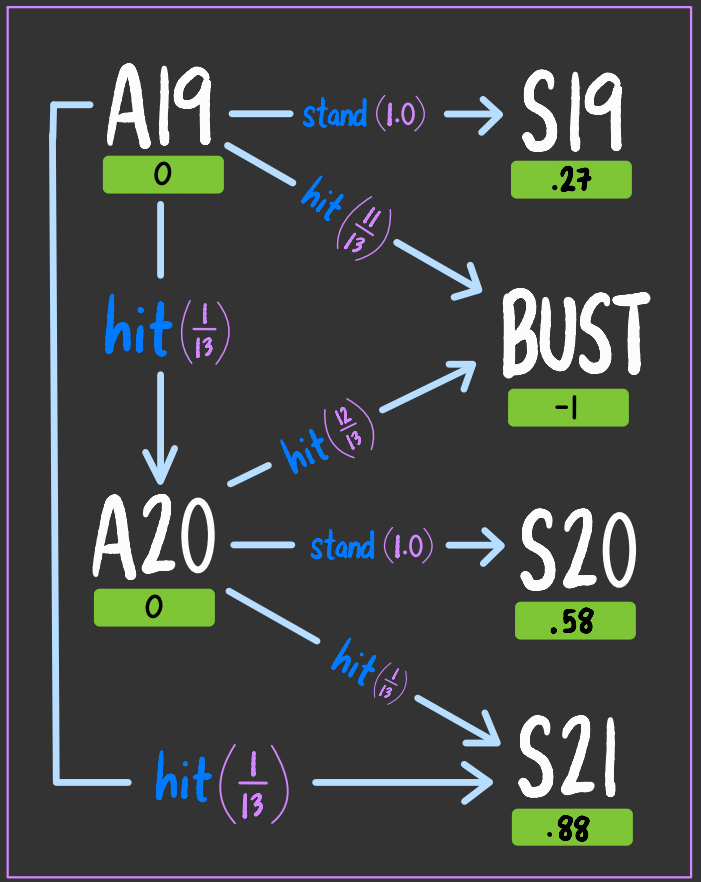
this augmented state machine is called a markov decision process



a markov decision process is a triple (M, R, ω) where:

- $M = (Q, \Sigma, \Delta, q_0, F)$ is a state machine
- $R : Q \to \mathbb{R}$ is a reward function
- ω assigns a positive weight to each transition such that for every state q and action σ

$$\sum_{\delta \in \Delta : \delta = (q, \sigma, q')} \omega(\delta) = 1$$

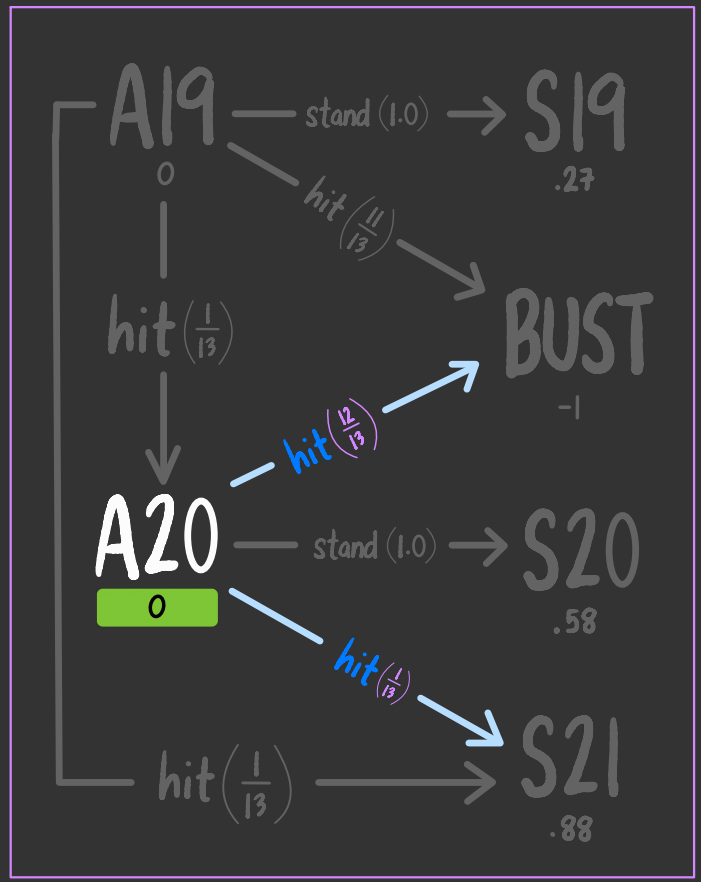


the transitions
originating from the same state,
labeled with the same action...

... their weights must sum to **one**

- ω assigns a positive weight to each transition such that for every state q and action σ

$$\sum_{\delta \in \Delta : \delta = (q, \sigma, q')} \omega(\delta) = 1$$

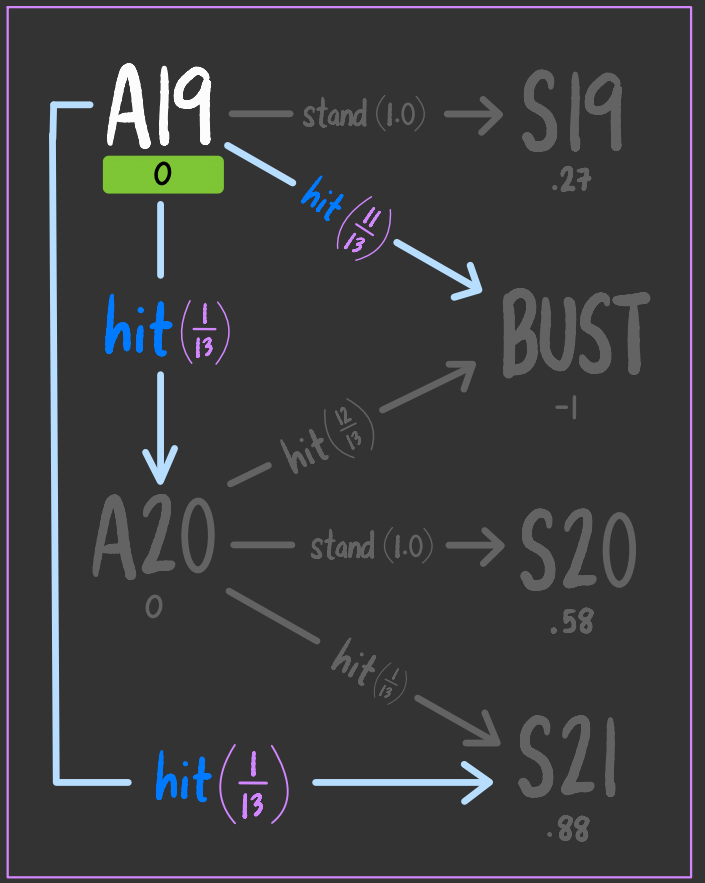


the transitions originating from the same state, labeled with the same action...

... their weights must sum to **one**

- ω assigns a positive weight to each transition such that for every state q and action σ

$$\sum_{\delta \in \Delta : \delta = (q, \sigma, q')} \omega(\delta) = 1$$

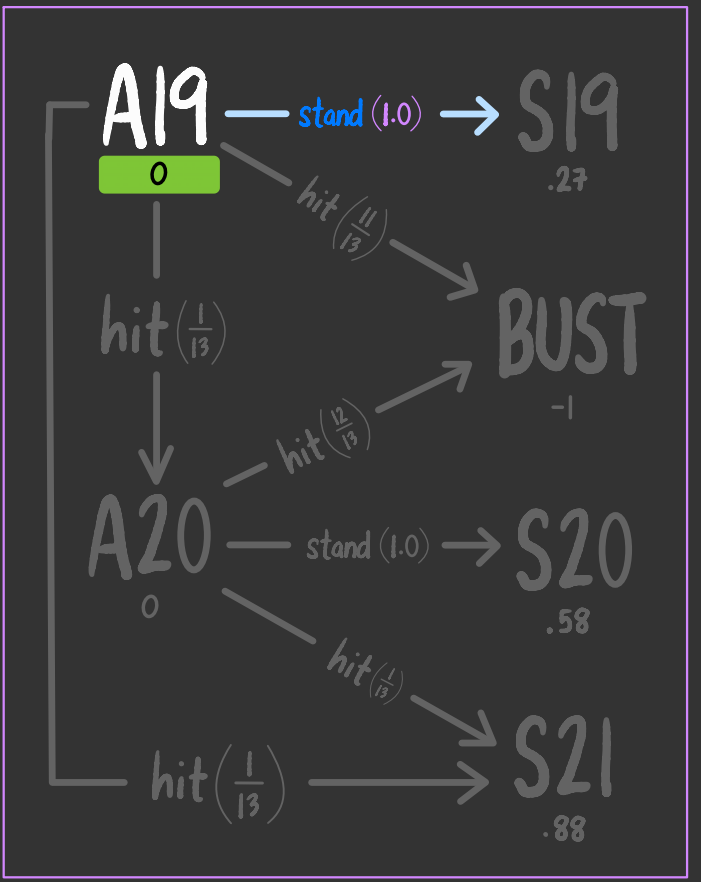


the transitions
originating from the same state,
labeled with the same action...

... their weights must sum to **one**

- ω assigns a positive weight to each transition such that for every state q and action σ

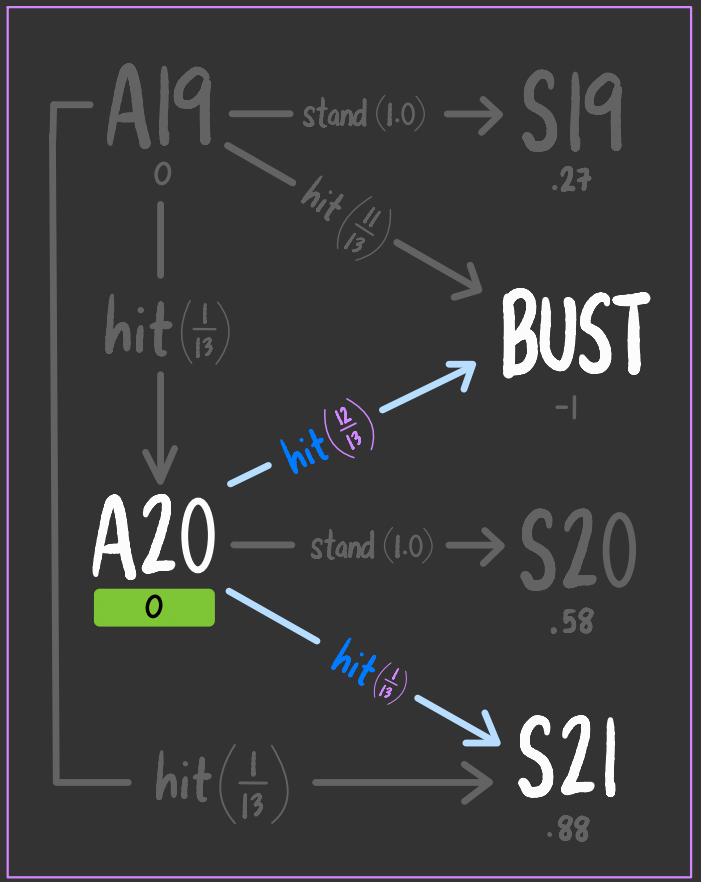
$$\sum_{\delta \in \Delta : \delta = (q, \sigma, q')} \omega(\delta) = 1$$



this is so that we can interpret the weights as conditional probabilities

$$P(\text{BUST} \mid \text{A20}, \text{hit}) = \frac{12}{13}$$

$$P(\text{S21} \mid \text{A20}, \text{hit}) = \frac{1}{13}$$



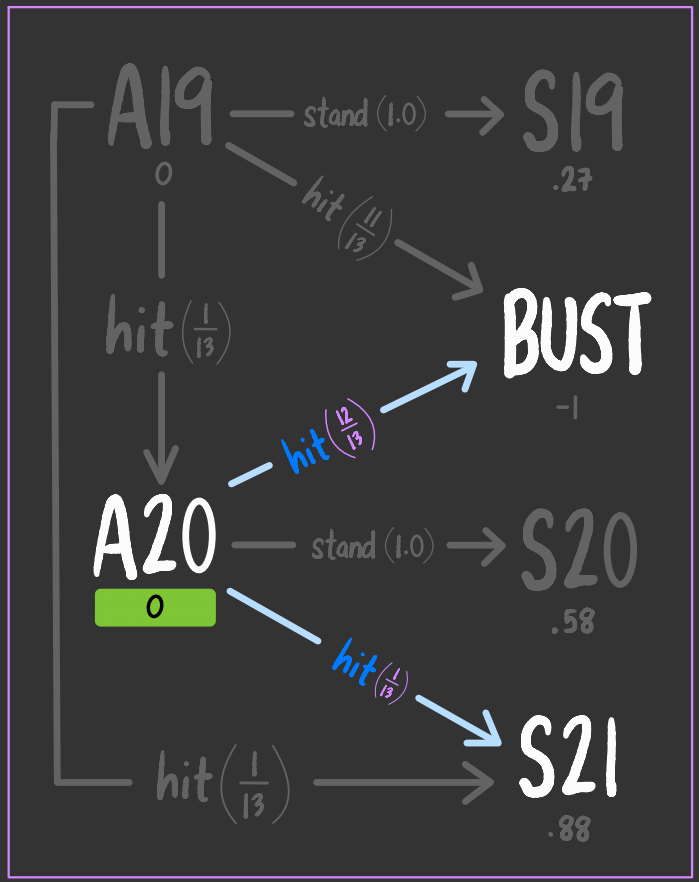
therefore

we will often write

$$P(q' \mid q, \sigma)$$

instead of

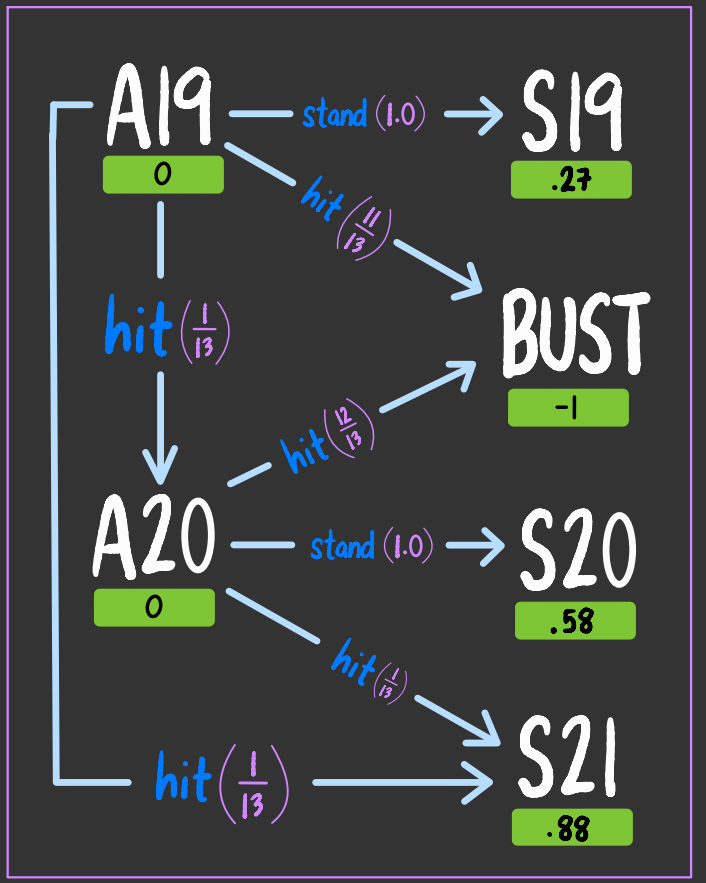
$$\omega(q, \sigma, q')$$



a markov decision process is a triple (M, R, ω) where:

- $M = (Q, \Sigma, \Delta, q_0, F)$ is a state machine
- $R : Q \to \mathbb{R}$ is a reward function
- ω assigns a positive weight to each transition such that for every state q and action σ

$$\sum_{\delta \in \Delta : \delta = (q, \sigma, q')} \omega(\delta) = 1$$



rewards accrue over time

a mark decision process is a t here:

- M = is a state machine
- $R : Q \to \mathbb{R}$ is a reward function
- w assigns a positive weight to each transition such that for every state q and action σ

$$\sum_{\delta \in \Delta : \delta = (q, \sigma, q')} w(\delta) = 1$$

given $\gamma \in [0, 1]$, define the discounted reward:

$$R_{0,\gamma}(q) = R(q)$$

$$R_{t,\gamma}(q) = \gamma R_{t-1,\gamma}(q)$$

γ is called the discount factor

a mark decision process is a t here:

- $M =$ is a state machine
- $R : Q \to \mathbb{R}$ is a reward function
- w assigns a positive weight to each transition such that for every state q and action σ

$$\sum_{\delta \in \Delta : \delta = (q, \sigma, q')} w(\delta) = 1$$

given $\gamma \in [0, 1]$, define the discounted reward:

$$R_{0,\gamma}(q) = R(q)$$

$$R_{t,\gamma}(q) = \gamma R_{t-1,\gamma}(q)$$

to avoid notational clutter, we usually omit the γ subscript

a mark decision process is a t here:

- $M =$ is a state machine
- $R : Q \to \mathbb{R}$ is a reward function
- w assigns a positive weight to each transition such that for every state q and action σ

$$\sum_{\delta \in \Delta : \delta = (q, \sigma, q')} w(\delta) = 1$$

given $\gamma \in [0, 1]$, define the discounted reward:

$$R_0(q) = R(q)$$

$$R_t(q) = \gamma R_{t-1}(q)$$

to avoid notational cluster, we usually omit the γ subscript

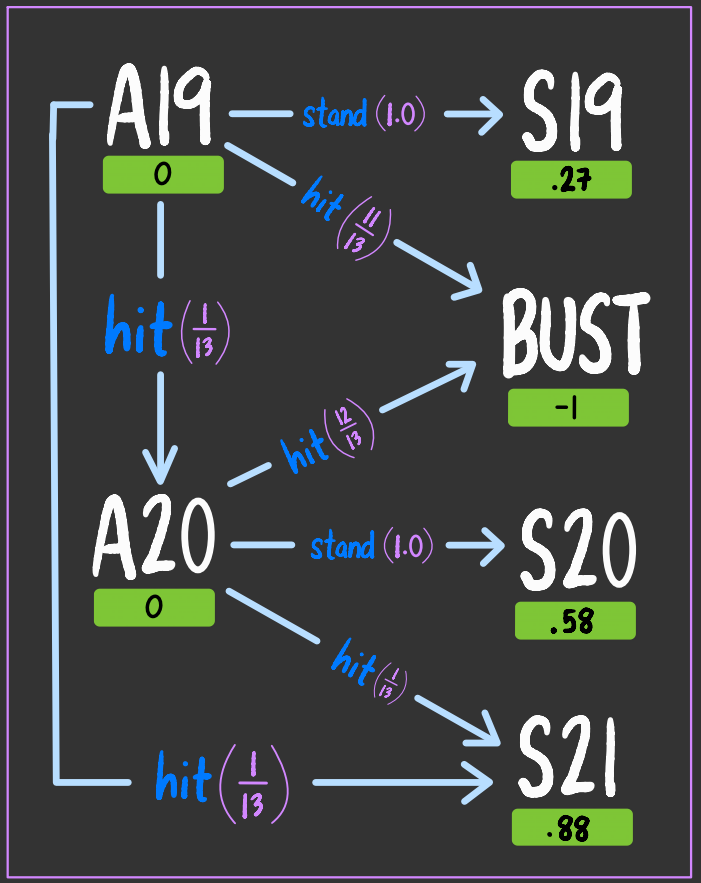
given $\gamma \in [0, 1]$, define the discounted reward:

$$R_0(q) = R(q)$$

$$R_t(q) = \gamma R_{t-1}(q)$$

what is $R_2(S21)$ if $\gamma = 0.5$?

?



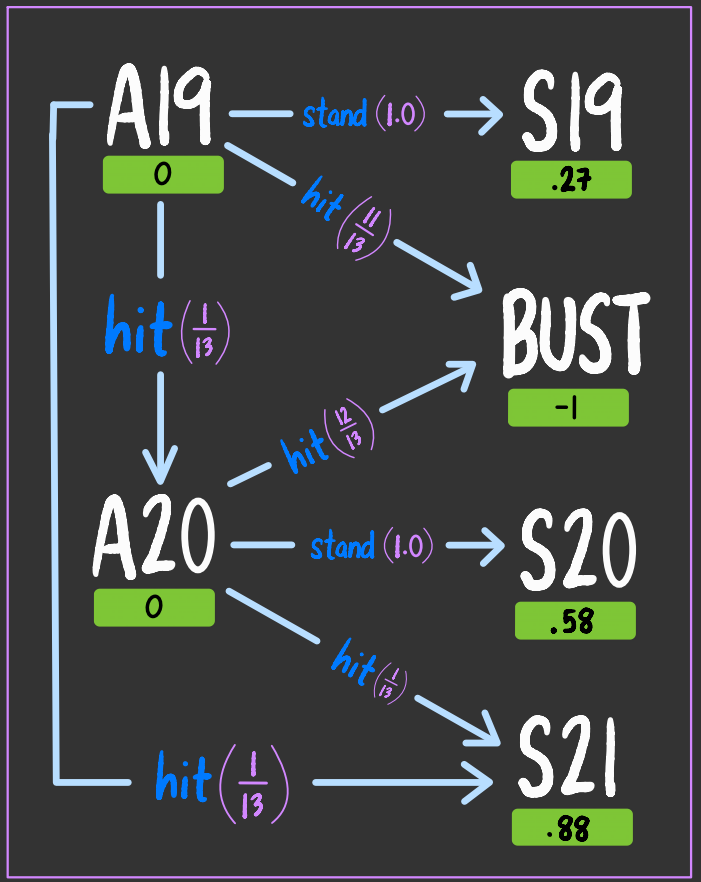
given $\gamma \in [0, 1]$, define the discounted reward:

$$R_0(q) = R(q)$$

$$R_t(q) = \gamma R_{t-1}(q)$$

what is $R_2(S21)$ if $\gamma = 0.5$?

.22



given $\gamma \in [0, 1]$, define the discounted reward:

$$R_0(q) = R(q)$$

$$R_t(q) = \gamma R_{t-1}(q) = \gamma^t R(q)$$

what is $R_2(S21)$ if $\gamma = 0.5$?

.22