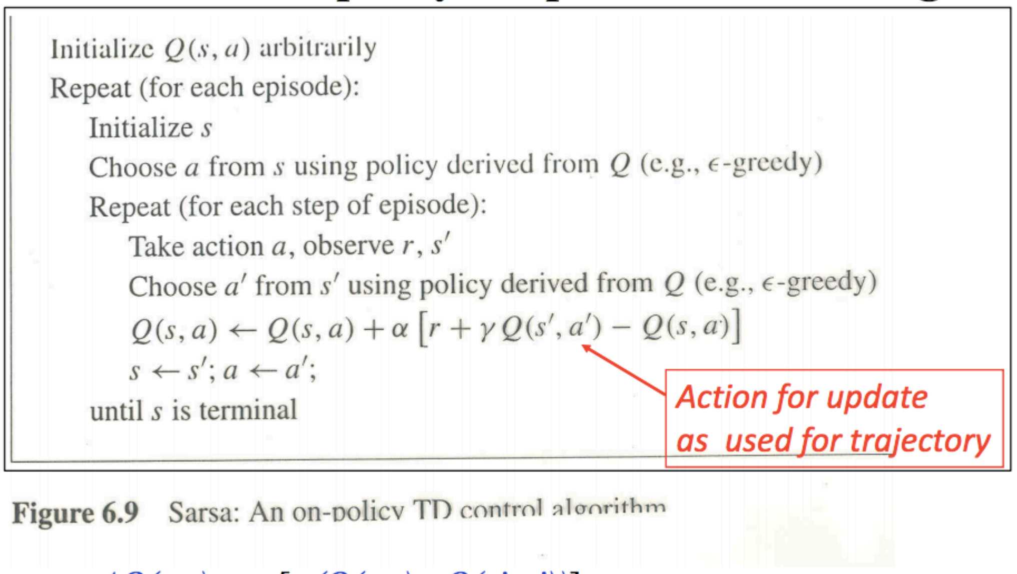
Sarsa – on policy temporal difference algo



```
Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
    Initialize $s$
    Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
    Repeat (for each step of episode):
        Take action $a$, observe $r$, $s'$
        Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$
        $s \leftarrow s'$; $a \leftarrow a'$;
    until $s$ is terminal
```

Action for update as used for trajectory

Figure 6.9 Sarsa: An on-policy TD control algorithm

$\Delta Q(s,a)=\eta\ [r-(Q(s,a)-\gamma Q(s',a'))]$

From: Reinforcement Learning, Sutton and Barto 1998

$$\Delta Q(s,a)=\eta\ [r-(Q(s,a)-\gamma Q(s',a'))]$$

SARSA algo

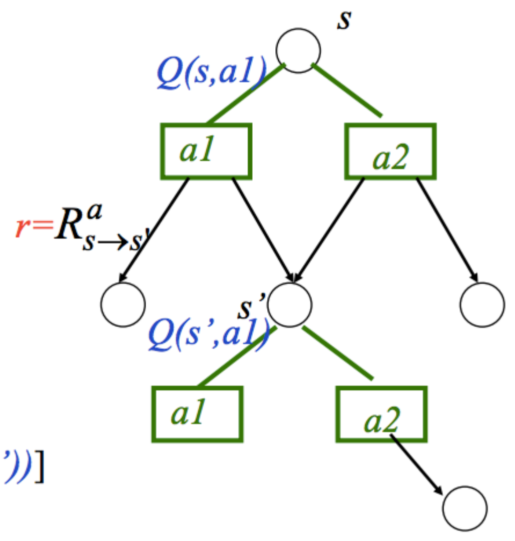
SARSA algo

Initialise Q values
Start from initial state ***s***

1) Being in state ***s*** choose action ***a*** according to policy $\boldsymbol{\pi}$
2) Observe reward ***r*** and next state ***s'***
3) Choose action **a'** in state s' according to policy $\boldsymbol{\pi}$
4) Update

   $\Delta Q(s,a)=\eta\ [r-(Q(s,a)-\gamma Q(s',a'))]$

5) s' $\rightarrow$ s; a' $\rightarrow$ a
6) Goto 1)



$\Delta Q(s,a)=\eta\ [r-(Q(s,a)-\gamma Q(s',a'))]$