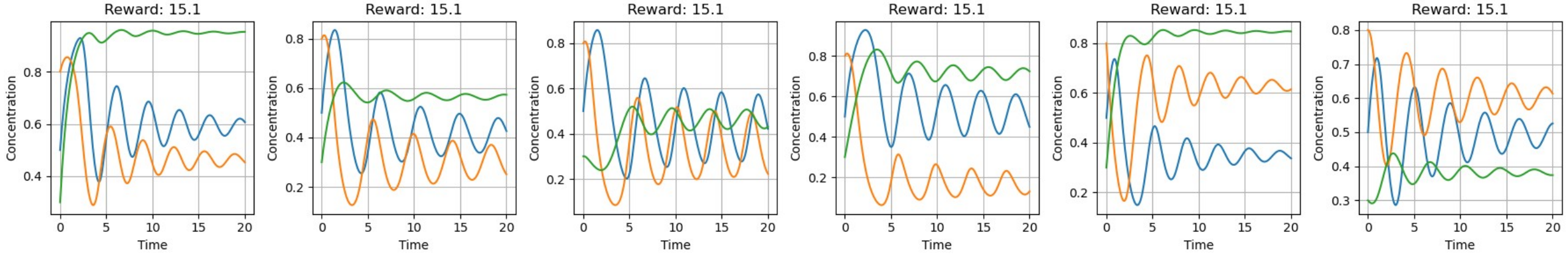
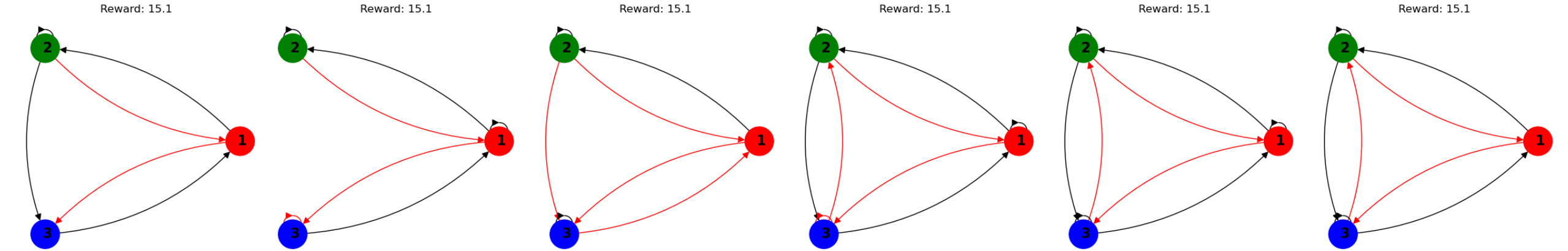
Discover Novel Genetic Circuit by GFN (EvoGFN) or (GRN-GFN)

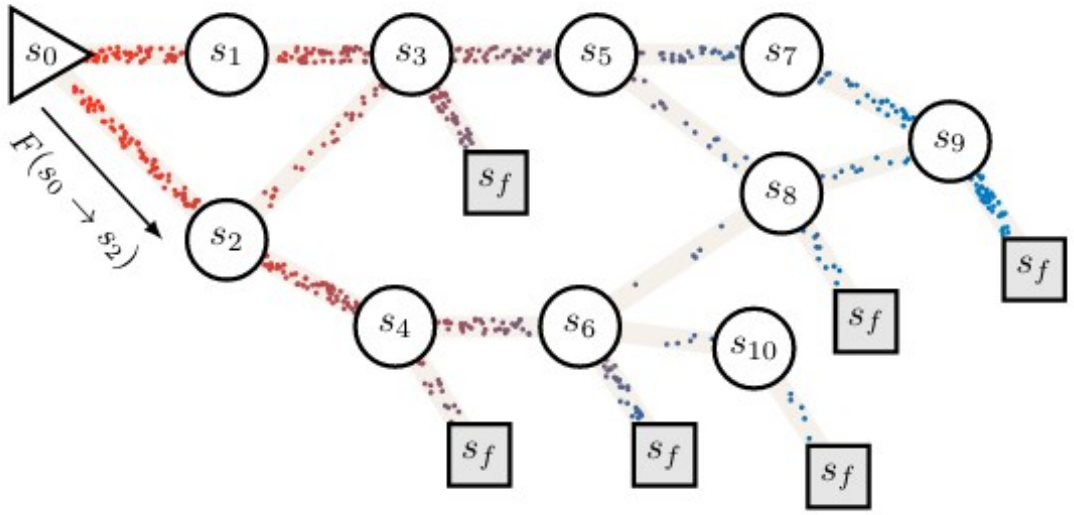


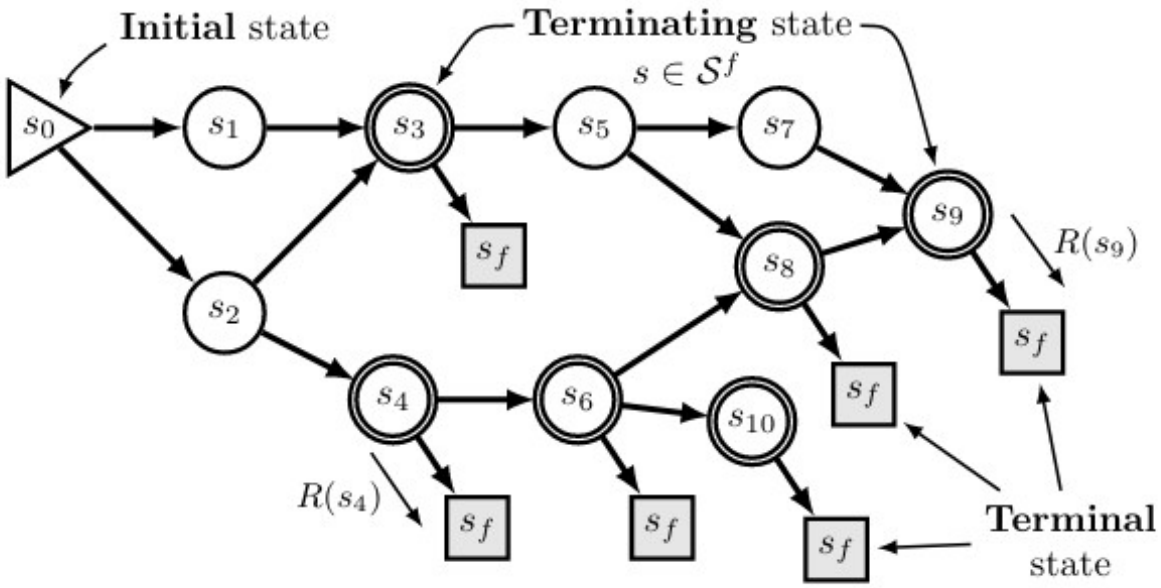


GFN in short

GFN is a sequential generative model based on flow networks, treating the root as a source and all possible generated objects as sinks, which generates objects proportionally to their reward.

$$p(x) \propto R(x)$$





Flow Conservation !!!

Motivation

Sample from Gibbs distribution:

$$P^*(x) = \frac{\exp(-\mathcal{E}(x)/\alpha)}{Z}$$

But partition function Z is intractable:

$$Z = \sum_{x' \in \mathcal{X}} \exp\left(-\mathcal{E}(x')/\alpha\right)$$

So instead we Sample from the distribution:

$$P^*(x) \propto \exp(-\mathcal{E}(x)/\alpha)$$

Motivation

Need a better alternative sampling method than MCMC

- MCMC methods are typically slower to converge
- produce samples that are highly correlated
- lots of random-walk, modes (local minima) are not diverse enough.

$$\pi(x) = \frac{n(x)R(x)}{\sum_{x' \in \mathcal{X}} n(x')R(x')}$$

The goal is to have:

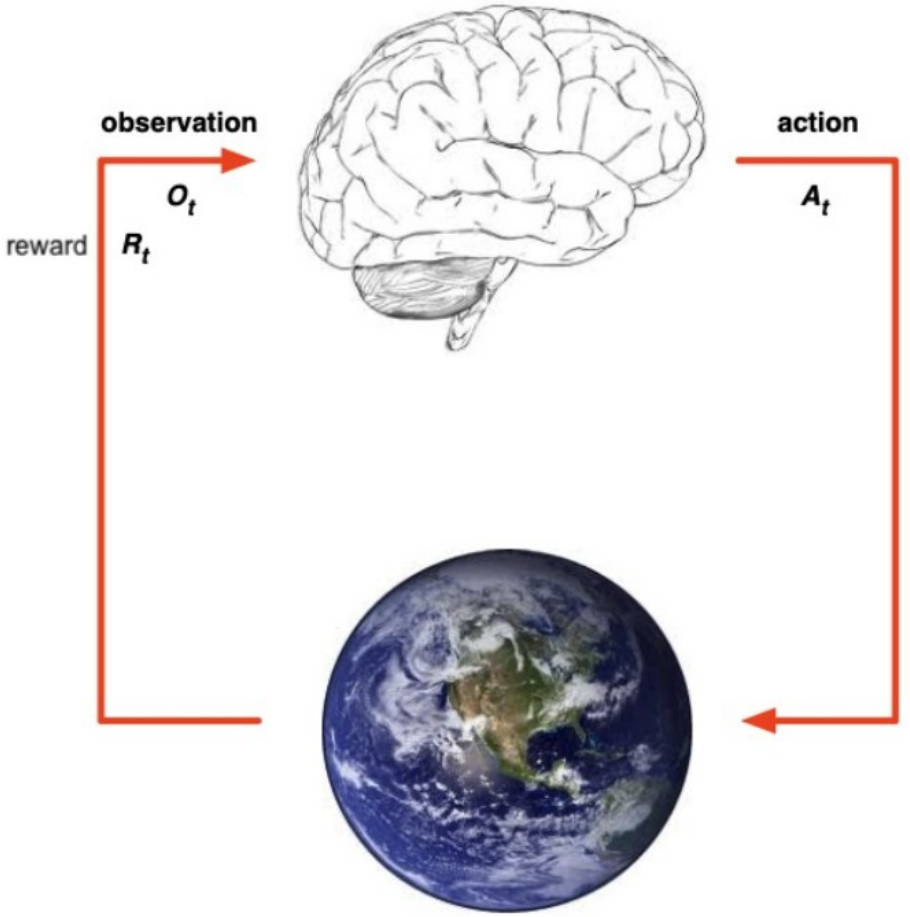
$$\pi(x) = \frac{R(x)}{\sum_{x' \in \mathcal{X}} R(x')}$$

$$p(x) \propto R(x)$$

Reinforcement Learning

Goal:

$$\pi^*_{RL} = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{T} r(s_t, s_{t+1}) \right]$$



Reward is defined by:

$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(s_T)$$

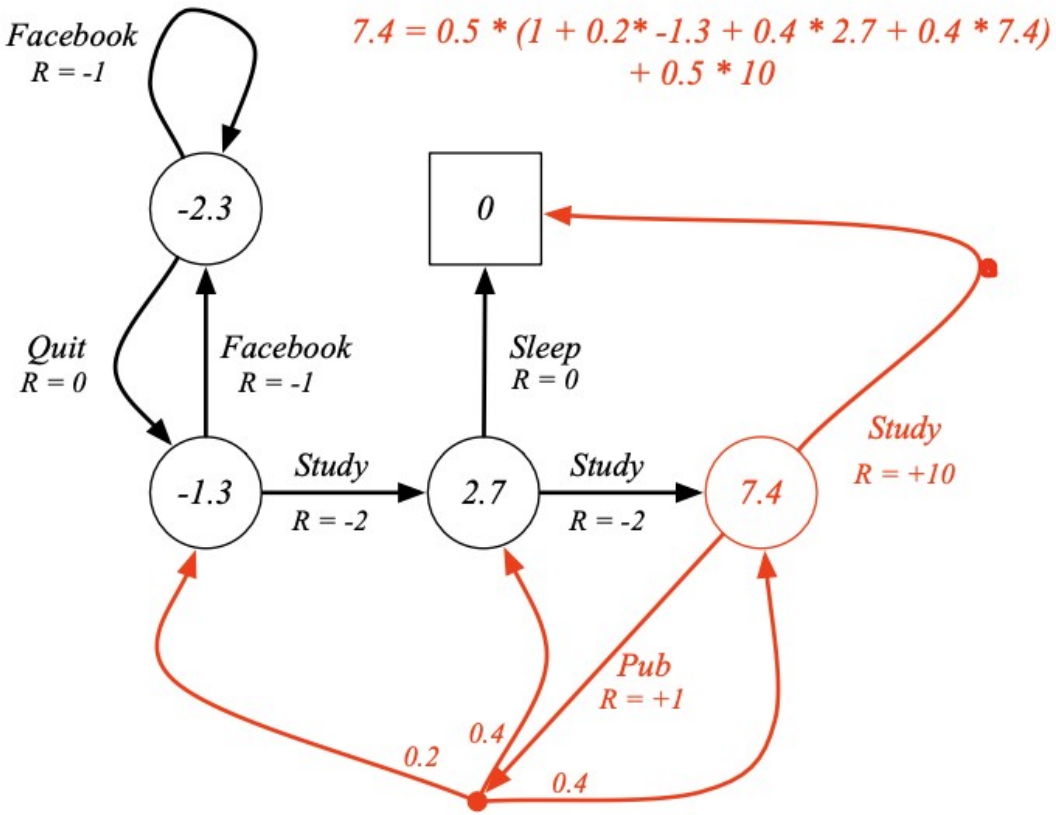
A policy is a distribution over actions given states:

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$$

Reinforcement Learning

Goal:

$$\pi^*_{RL} = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{T} r(s_t, s_{t+1}) \right]$$



Reward is defined by:

$$\sum_{t=0}^{T} r(s_t, s_{t+1}) = -\mathcal{E}(s_T)$$

A policy is a distribution over actions given states:

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$$

GFN vs. RL

- The objective of RL policies is to find trajectories that maximize return,

- GFlowNets are trained so they sample terminal states that match some desired unnormalized probability function. (which is the reward function for GFlowNet)

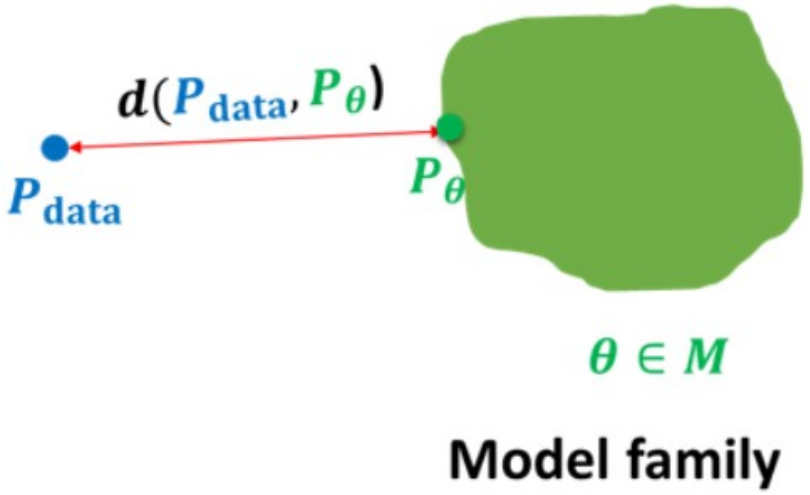
- The value function in RL estimates an expected sum of downstream rewards

- The flow function of GFlowNets at state s estimates the fraction of the sum of all downstream rewards

Learning a Generative model

We are given a training set of examples. eg. images of dogs

$$x_i \sim P_{\text{data}}$$
$$i = 1, 2, \ldots, n$$



p(x) should be high if x looks like a dog, and low otherwise

If we sample x_new ~ p(x), x_new should look like a dog

GFN vs. Generative model

- A GFlowNet is a generative model because during (and after) training we can sample from it,

- but its training objective is about matching a reward function R rather than fitting a finite dataset (the usual objective for generative models).

Review of ML:

The goal is to find f such that $y = f(x)$

f can be parametrized by ... $y = f_\theta(x)$

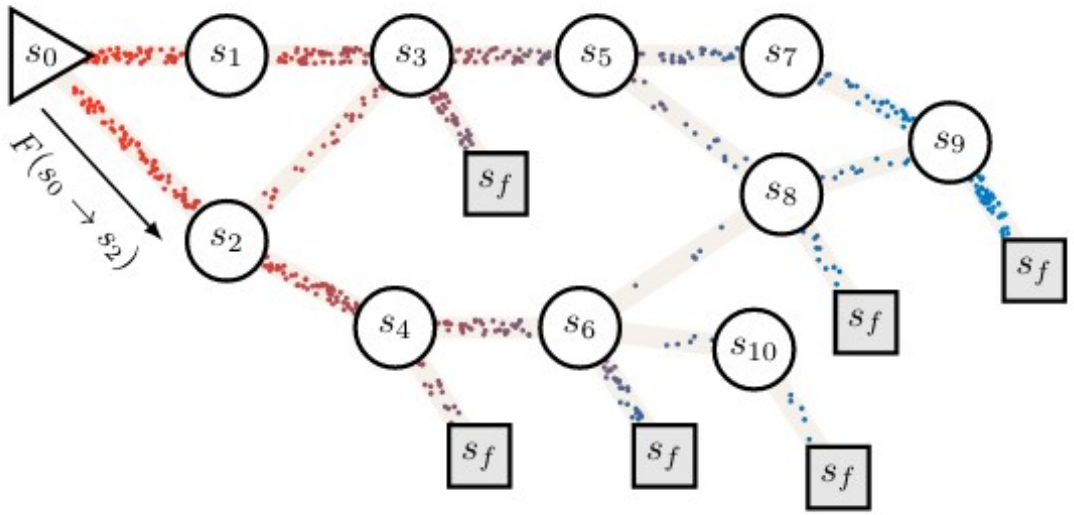
$Loss = y_0 - f(x_0, \theta)$ —GD—> θ_0

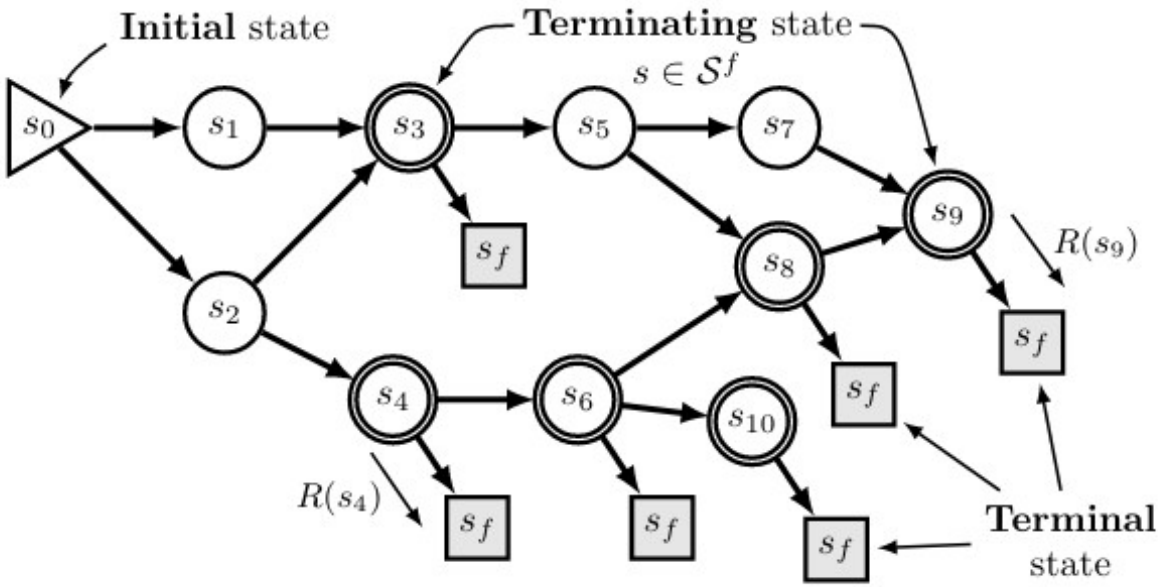
$y = f_{\theta_0}(x) = f(x)$

Back to GFN

GFN is a sequential generative model based on flow networks, treating the root as a source and all possible generated objects as sinks, which generates objects proportionally to their reward.

$$p(x) \propto R(x)$$



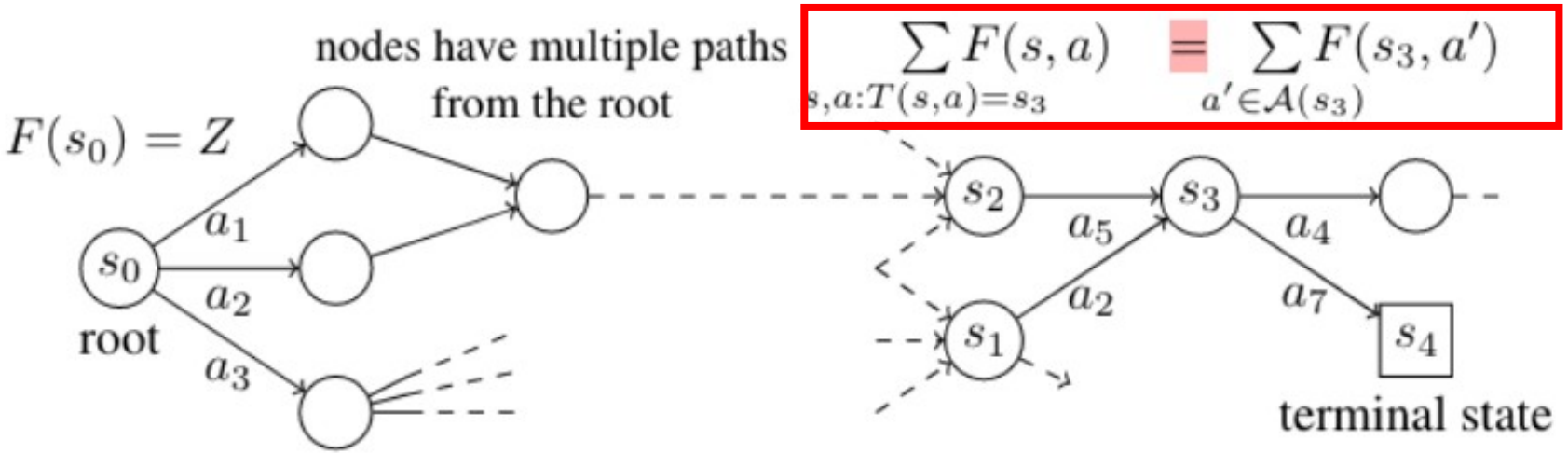


Flow Conservation !!!

Flow Conservation

in-flow = out-flow (flow conservation property)
meaning that the amount of **probability mass** flowing into a state equals the amount flowing out, which is crucial for generating valid samples from the target distribution.

s_0: a single source of the water, the root
x_i: the sinks, terminal states. We'll assign to each sink x an out-flow R(x).
$\sum_x R(x) = Z$: the flow out of the unique source s_0, the partition function.



Flow Conservation

in-flow = out-flow :

$$\sum_{s,a:T(s,a)=s'} F(s,a) = F(s') = \sum_{a' \in \mathcal{A}(s')} F(s',a')$$

$R(s) = 0$ for interior nodes

=>

$$\sum_{s,a:T(s,a)=s'} F(s,a) = R(s') + \sum_{a' \in \mathcal{A}(s')} F(s',a')$$

A bit technical Def of F'

Definition of Flow:

$$\pi(a|s) = \frac{F(s,a)}{F(s)}$$

$$= \pi(s \to s' = T(s,a)|s) = P_F(s'|s) = \frac{F(s \to s')}{\sum_{s''} F(s \to s'')}$$

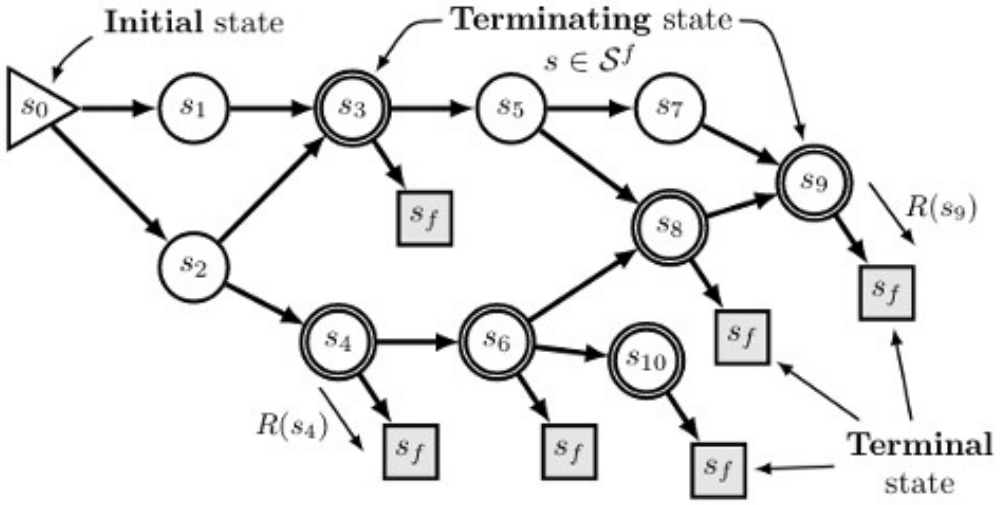
$$F(s_0) = Z = \sum_x R(x)$$

=>

$$\pi(s) = \frac{F(s)}{F(s_0)}$$

$$p(x) \propto R(x)$$

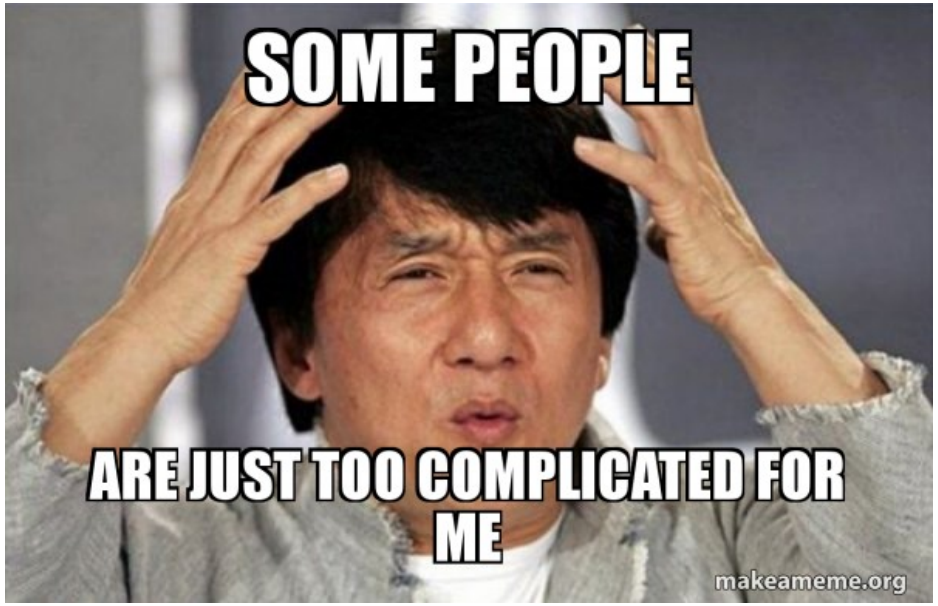
$$\pi(x) = \frac{R(x)}{\sum_{x' \in \mathcal{X}} R(x')}$$

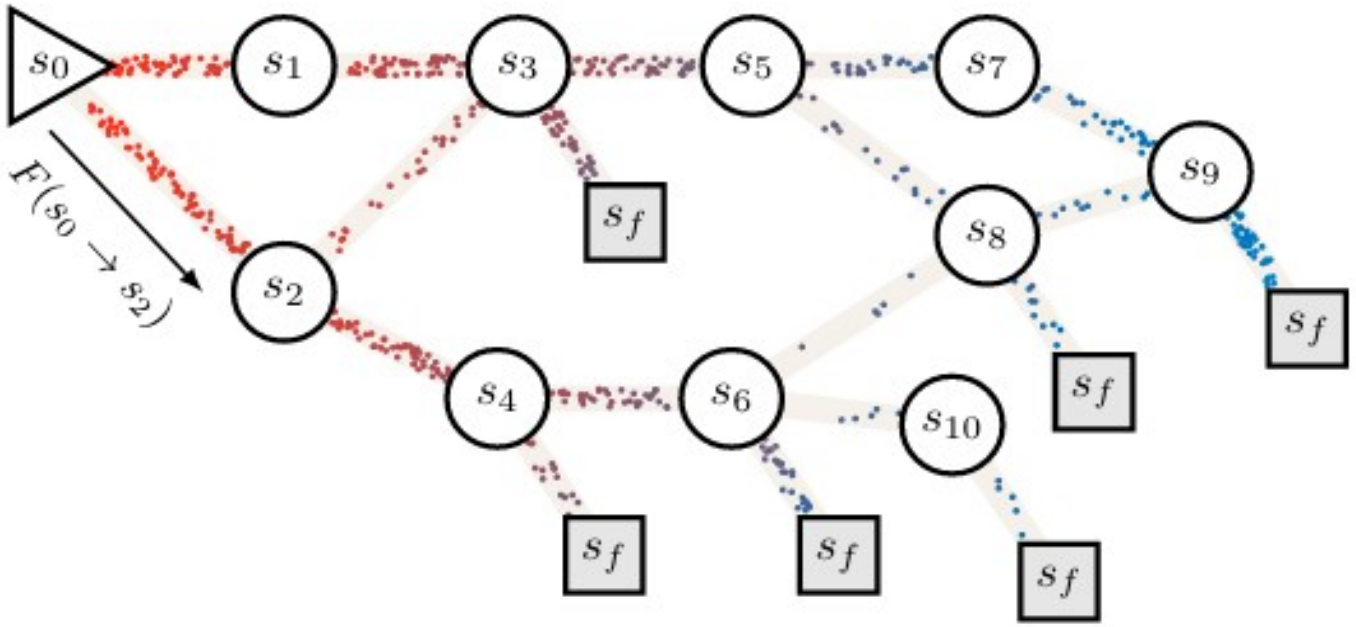


Intuition:

Intuition: $\pi(s) \sim F(s) \sim R(s_0)$

$\pi(s) \sim F(s) \sim R(s_0)$





Training of GFN

Review of ML:

The goal is to find f such that $y = f(x)$

f can be parametrized by ... $y = f_\theta(x)$

$Loss = y_0 - f(x_0, \theta)$ —GD—> θ_0

$y = f_{\theta_0}(x) = f(x)$

GFN:

The goal is to find $F(s, a)$ such that $p_{mass} = F(s, a)$

$p_{mass} = F_\theta(s, a)$

Loss = ...

Loss Func

Objective Functions for GFlowNet

$$\tilde{L}_\theta(\tau) \equiv \sum_{s' \in \tau \neq s_0} \left(\sum_{s,a:T(s,a)=s'} F_\theta(s,a) - R(s') - \sum_{a' \in \mathcal{A}(s')} F_\theta(s',a') \right)^2 .$$

Flow Predictor $F_\theta(s,a)$:

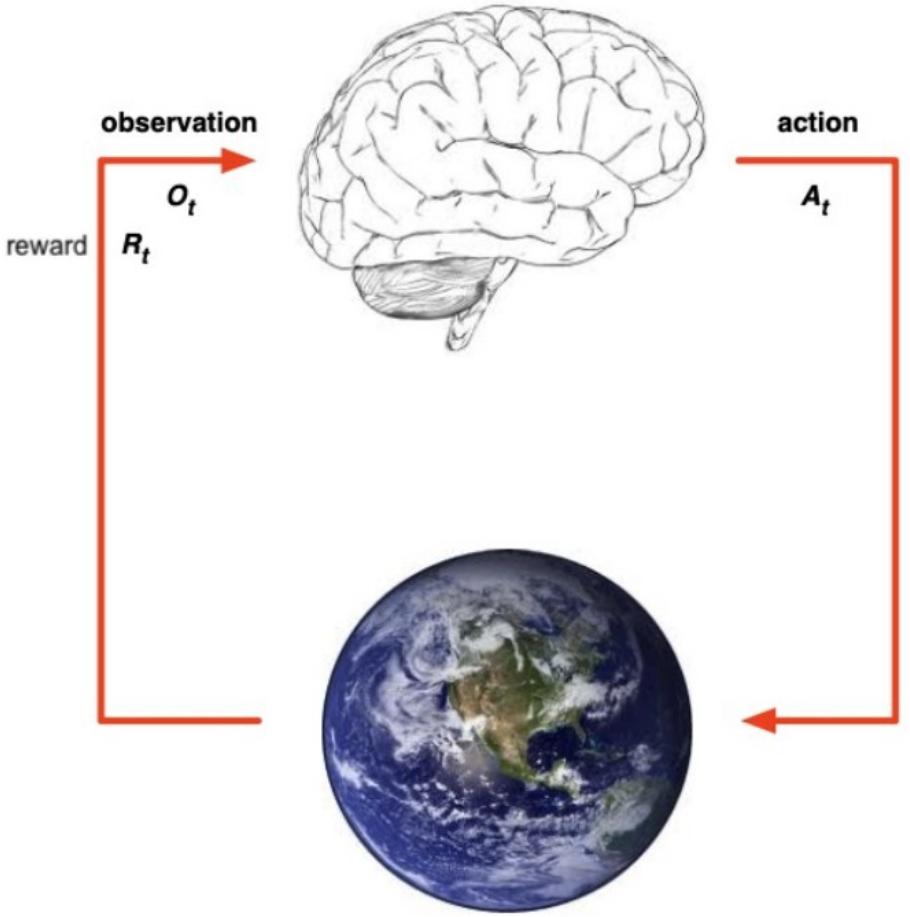
- F_θ is the neural network parameterized by θ that predicts these flows.
- For a given state s and action a, the flow predictor $F_\theta(s,a)$ predicts the probability of taking action a in state s to transition to the next state $s\prime$.
- The flow predictor is trained to ensure that the flow conservation property is satisfied.

GFN in practice

S: State space

R: Reward

A: Actions

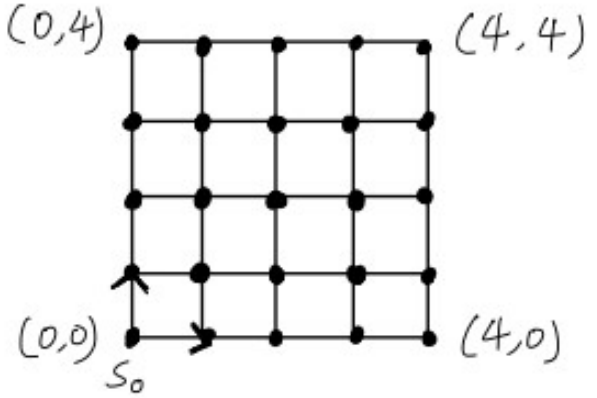


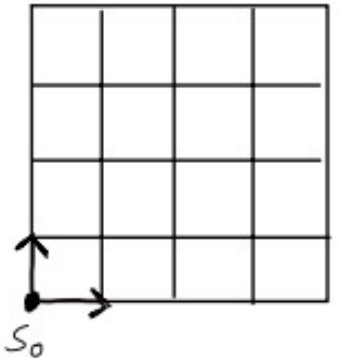
2d grid example

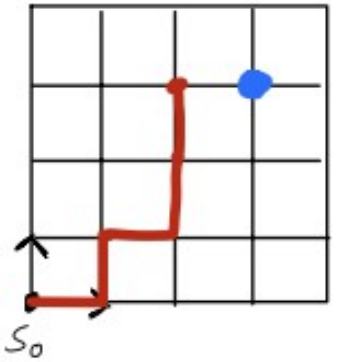
S: State space

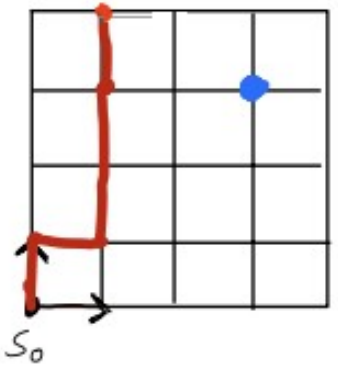
R: Reward

A: Actions (up, right, stop)





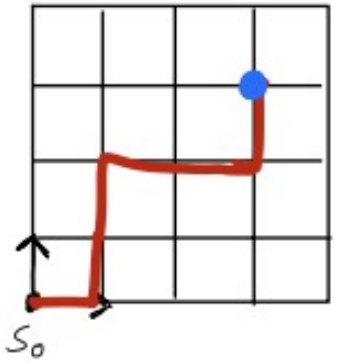




Terminate the trajectory when

- chose “stop” action by chance 主动 stop
- hit the boundary 被动 stop

都有 reward

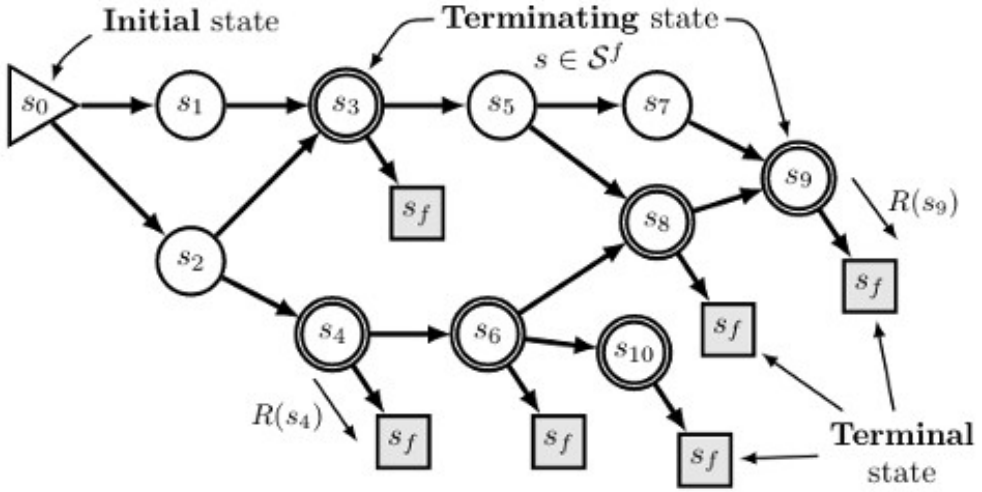



```
done = s.max() >= self.horizon - 1 or a == self.ndim
```

3-node sys w sogmoid:

State:

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}\right) - \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}$$

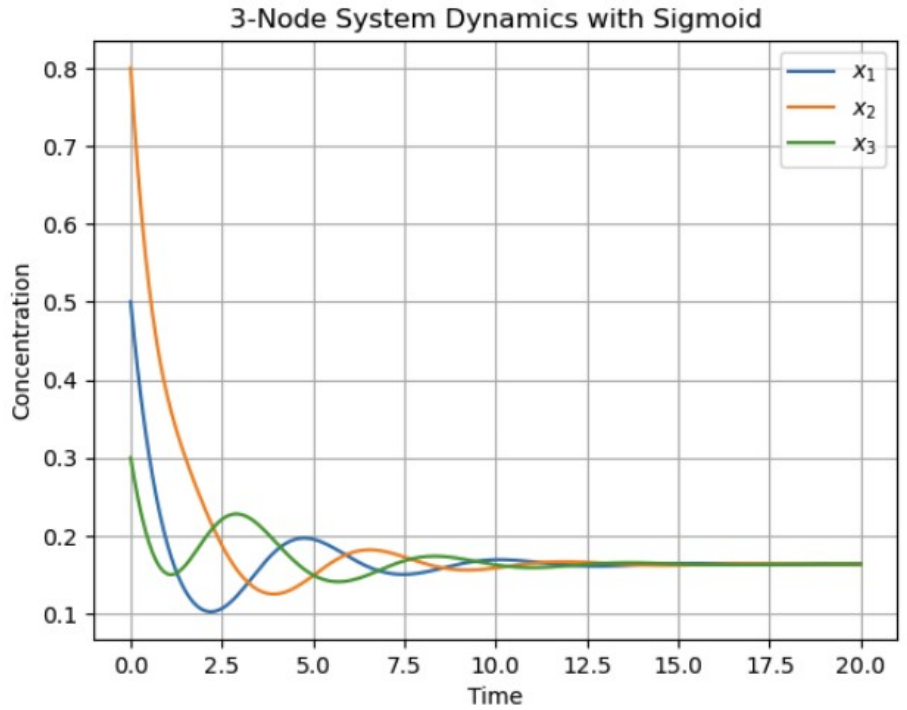


Action: tweaking weights

(*problem: can only increase in each direction)

Reward:

$$R = \sum_{i=1}^{3} \begin{cases} 1 & \text{if } (\max(x_i(\text{peaks}_i)) - \min(x_i(\text{peaks}_i))) > \delta \\ 0 & \text{otherwise} \end{cases}$$



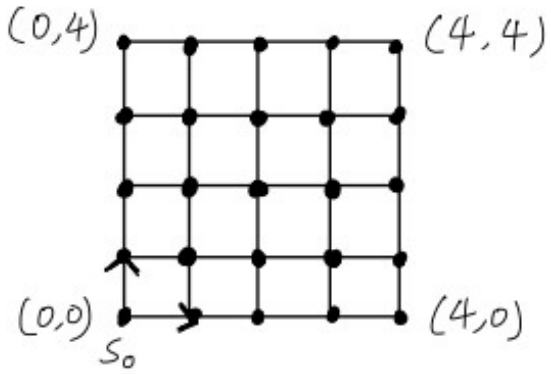
Reward: 3

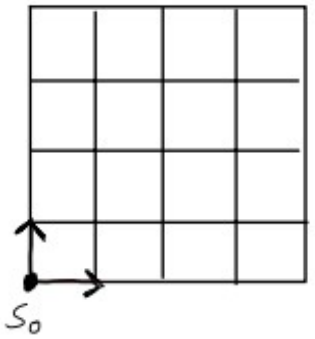
2d grid example

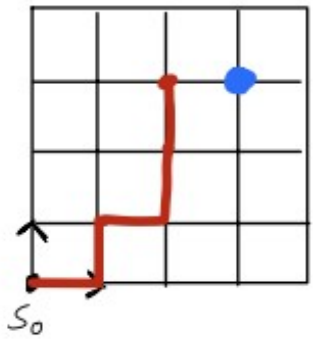
S: State space

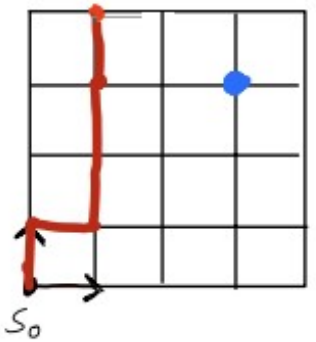
R: Reward

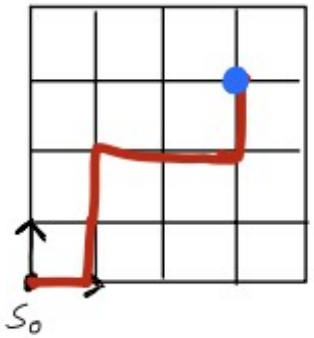
A: Actions
(up, right, stop)



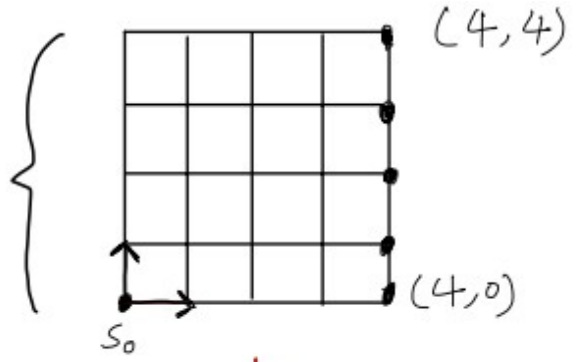


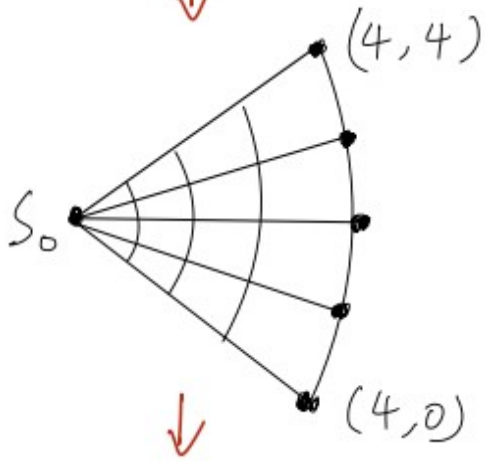


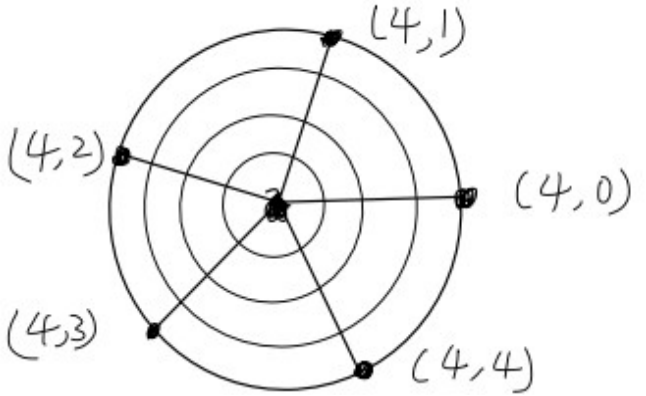




shrink







same coordinate, but different interpretation

General Formula for N-dimensional Hyperspherical Coordinates

For an N-dimensional space, the conversion from hyperspherical coordinates to Cartesian coordinates follows this general pattern:

- $x_1 = r * \cos(\varphi_1)$
- $x_2 = r * \sin(\varphi_1) * \cos(\varphi_2)$
- $x_3 = r * \sin(\varphi_1) * \sin(\varphi_2) * \cos(\varphi_3)$
- ...
- $x_{n-1} = r * \sin(\varphi_1) * ... * \sin(\varphi_{n-2}) * \cos(\varphi_{n-1})$
- $x_n = r * \sin(\varphi_1) * ... * \sin(\varphi_{n-2}) * \sin(\varphi_{n-1})$

Where:

- r is the radial distance from the origin
- φ_i are the angular coordinates (i = 1, 2, ..., n-1)
- x_i are the Cartesian coordinates (i = 1, 2, ..., n)

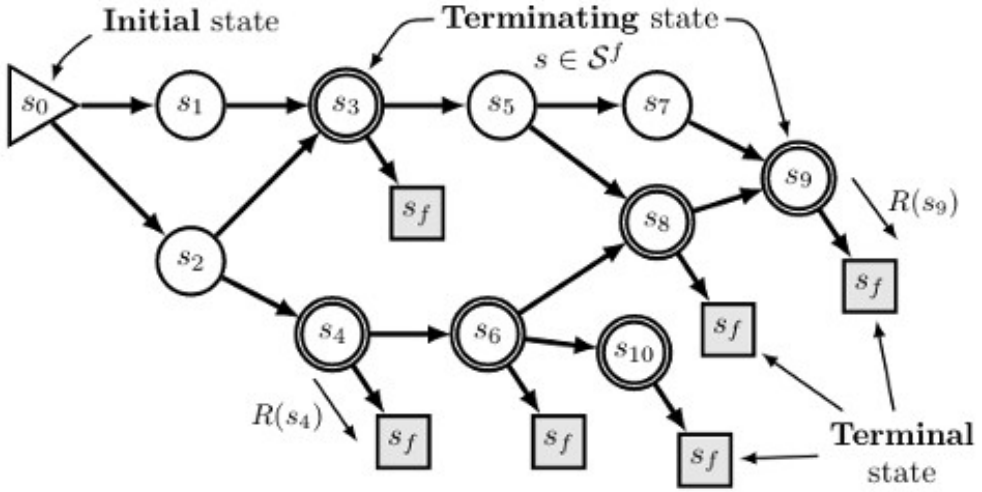
The **spherinder** can be seen as the volume between two parallel and equal solid 2-spheres (3-balls) in 4-dimensional space, here stereographically projected into 3D.

$$x = r \cos \varphi \sin \theta$$
$$y = r \sin \varphi \sin \theta$$
$$z = r \cos \theta$$
$$w = w$$

3-node sys w sogmoid:

State: (9-d grid.)

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}\right) - \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}$$



Action: tweaking weights (up, right, stop)

~~(*problem:~~ can only increase in each dimension)

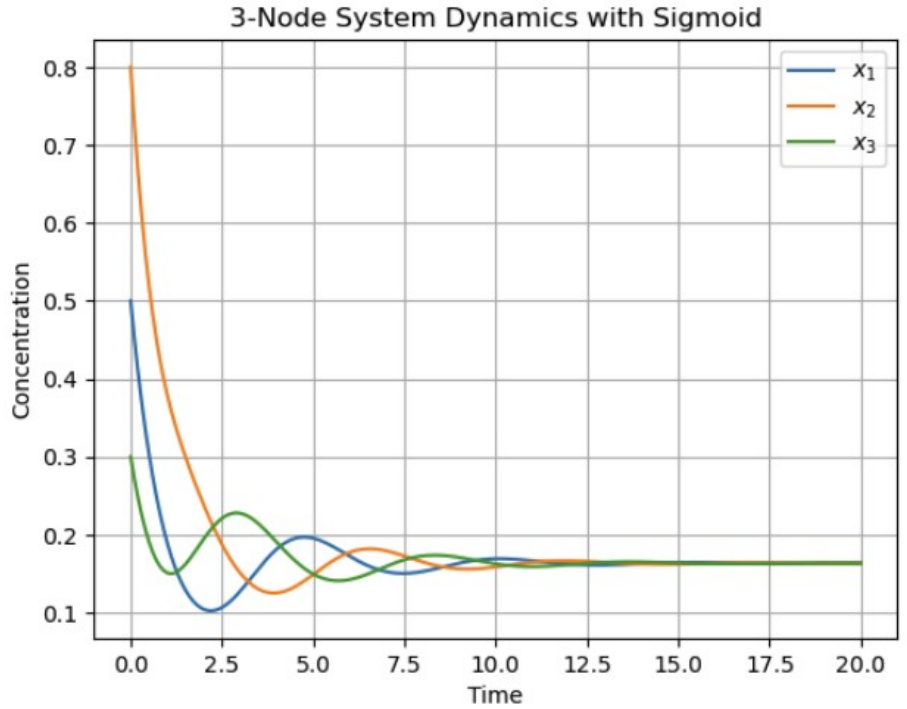
Reward:

$$R = \sum_{i=1}^{3} \begin{cases} 1 & \text{if } (\max(x_i(\text{peaks}_i)) - \min(x_i(\text{peaks}_i))) > \delta \\ 0 & \text{otherwise} \end{cases}$$

Interpret the grid of indices (state s) to sunflower (or clock)

→ treat as polar-line coord

→ transform to the cartesian coord

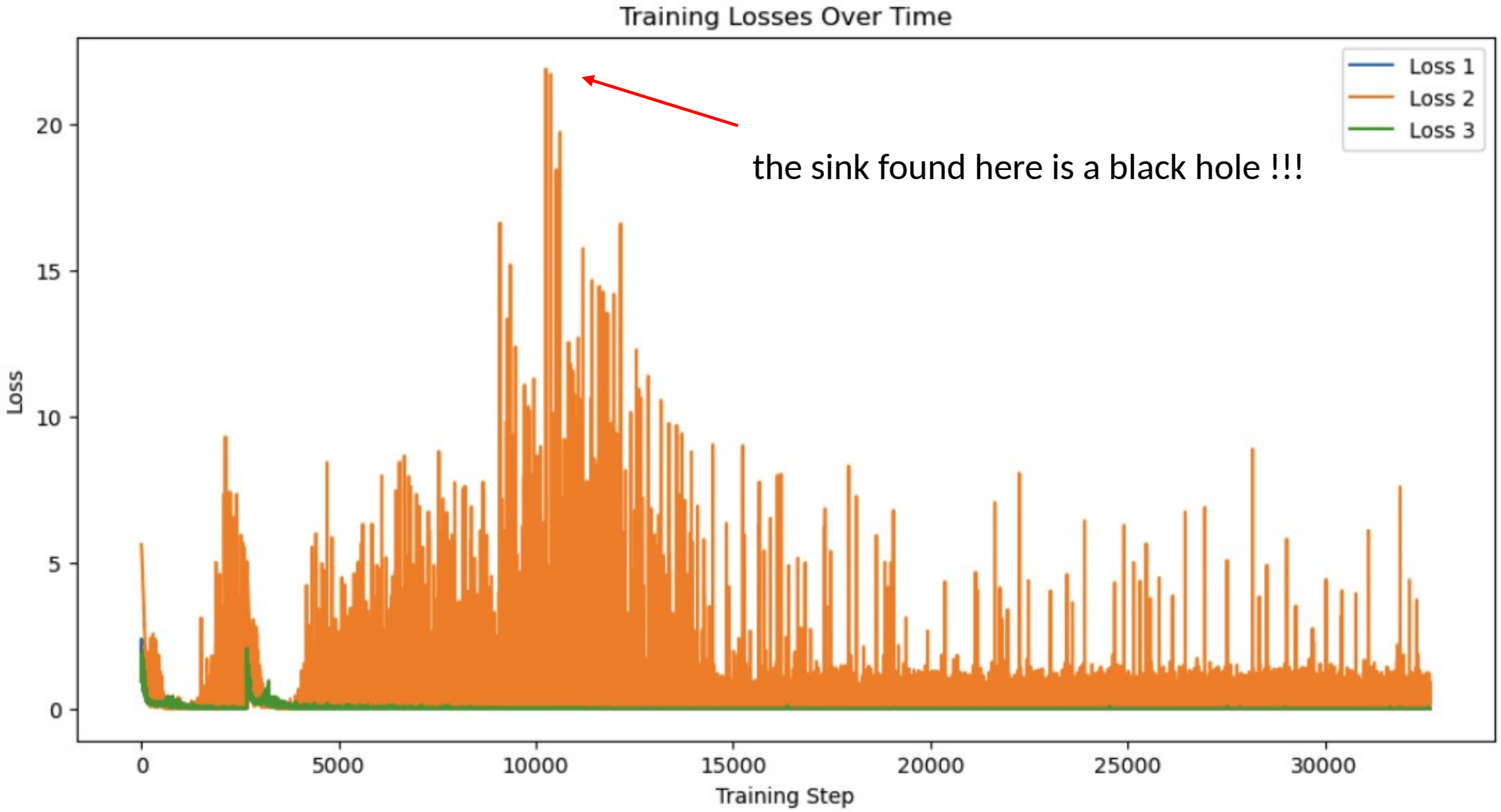



```
Reward: 3
```

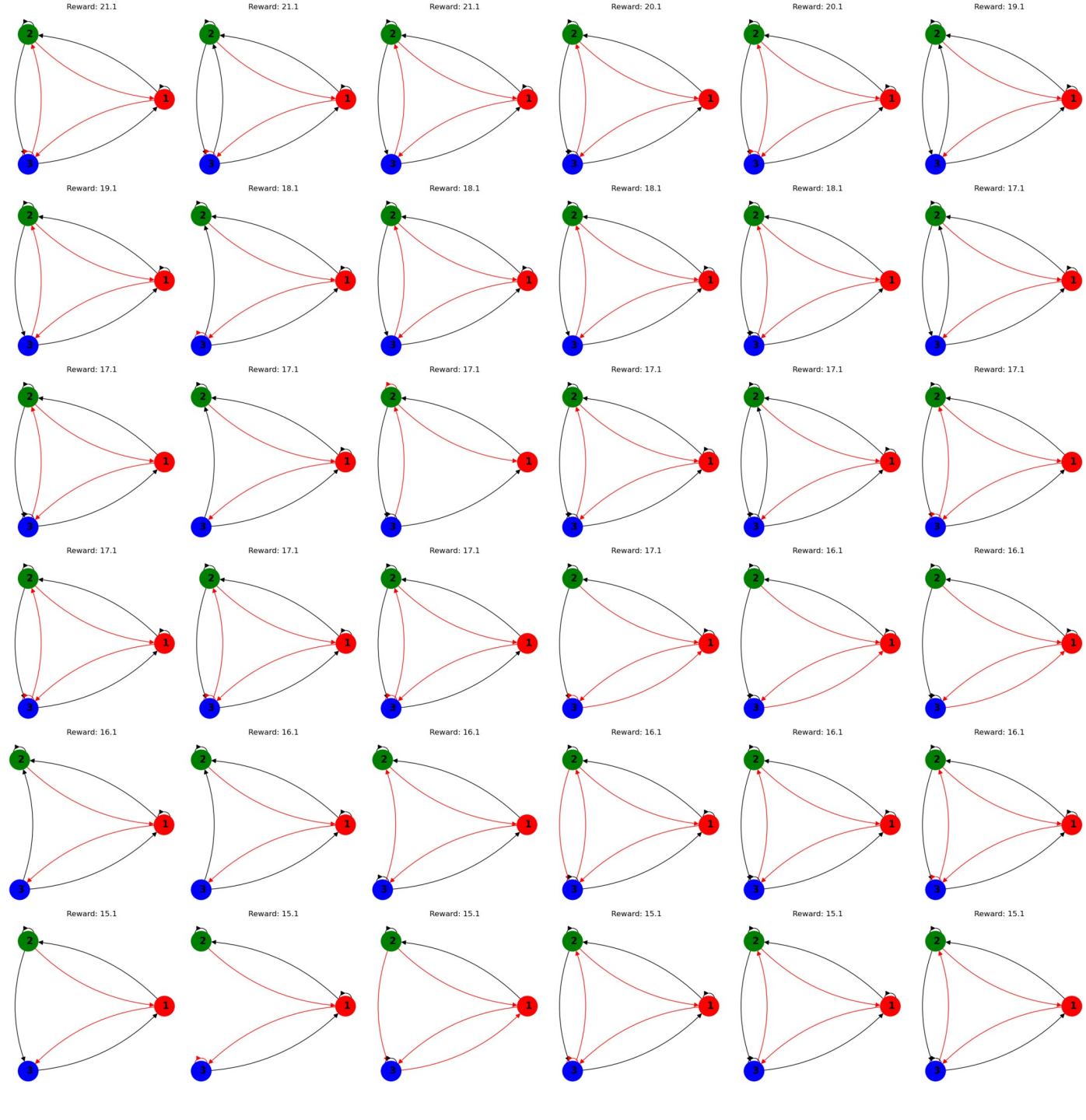
```
Empirical Reward Distribution: {0.1: 0.9951261467889908, 1.1: 0.0047305045871559636, 2.1: 0.00014334862385321102}
  0%|                                                  | 201/90001 [00:51<10:43:05,  2.33it/s]
Empirical Reward Distribution: {0.1: 0.9951026119402985, 1.1: 0.004780783582089552, 2.1: 0.0001166044776119403}
  0%|                                                  | 301/90001 [01:37<10:14:40,  2.43it/s]
Empirical Reward Distribution: {0.1: 0.9932193396226415, 1.1: 0.0066823899371069185, 2.1: 9.827044025157233e-05}
  0%|                                                  | 401/90001 [02:20<11:09:15,  2.23it/s]
Empirical Reward Distribution: {0.1: 0.991593070652174, 1.1: 0.008237092391304348, 2.1: 0.00016983695652173913}
  1%|▍                                                 | 502/90001 [02:51<5:02:04,  4.94it/s]
Empirical Reward Distribution: {0.1: 0.9916267942583732, 1.1: 0.008223684210526315, 2.1: 0.00014952153110047846}
  1%|▍                                                 | 602/90001 [03:05<2:30:18,  9.91it/s]
Empirical Reward Distribution: {0.1: 0.9921207264957265, 1.1: 0.007745726495726496, 2.1: 0.00013354700854700856}
```

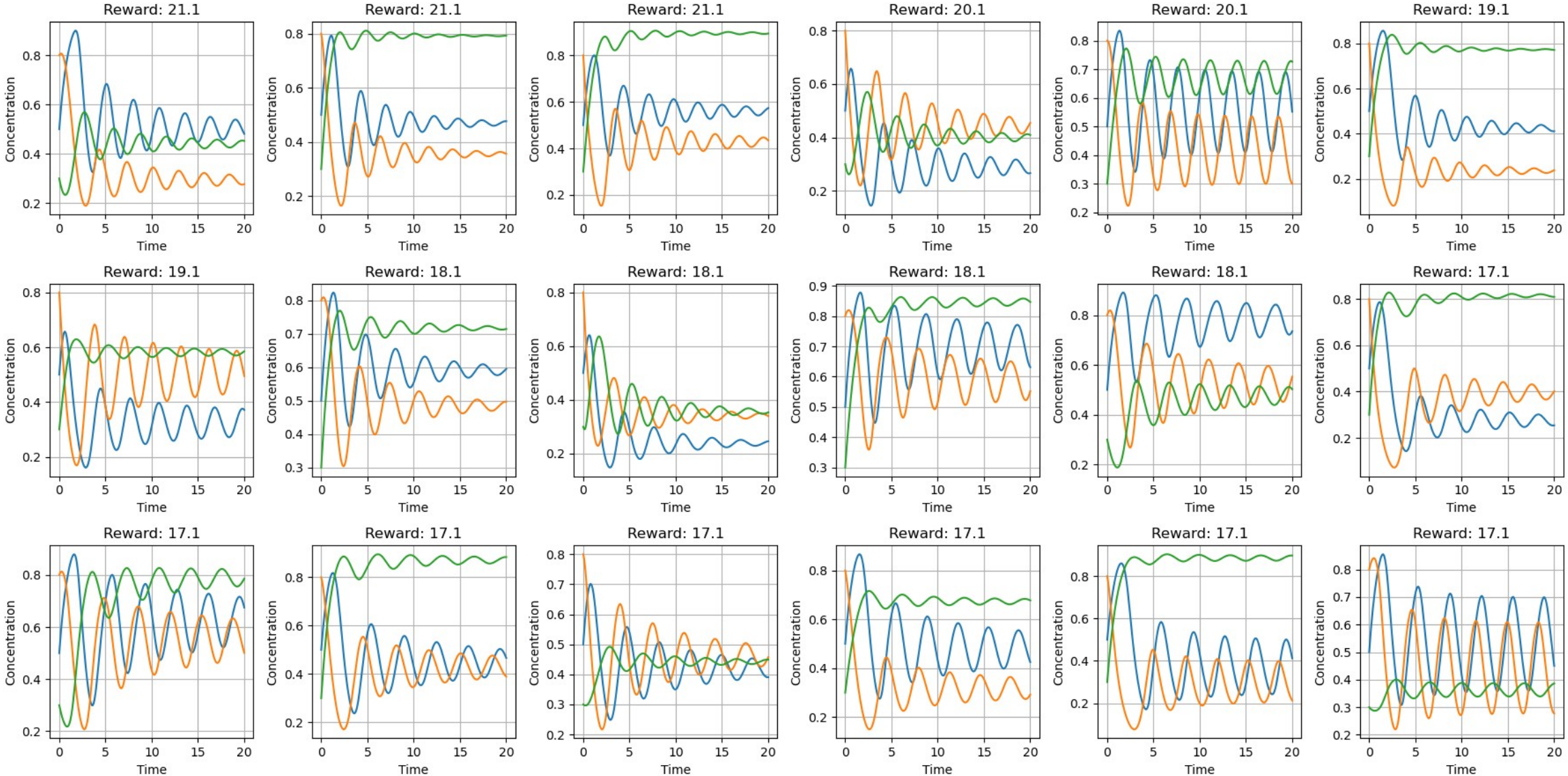
```
  4%|██                                                | 3601/90001 [12:29<5:31:29,  4.34it/s]
Empirical Reward Distribution: {0.1: 0.8612487296747967, 1.1: 0.12006161077235772, 2.1: 0.018610264227642278, 4.1: 1.5879065040650408e-05, 8.1: 1.5879065040650408e-05, 3.1: 4.763719512195122e-05}
  4%|██                                                | 3701/90001 [12:52<5:37:35,  4.26it/s]
Empirical Reward Distribution: {0.1: 0.8565721010901883, 1.1: 0.12442703171456888, 2.1: 0.018907953419226957, 4.1: 1.5485629335976214e-05, 8.1: 1.5485629335976214e-05, 3.1: 6.194251734390486e-05}
  4%|██                                                | 3802/90001 [13:14<4:59:08,  4.80it/s]
Empirical Reward Distribution: {0.1: 0.8525598404255319, 1.1: 0.12835469052224371, 2.1: 0.01899480174081238, 4.1: 1.5111218568665377e-05, 8.1: 1.5111218568665377e-05, 3.1: 6.044487427466151e-05}
  4%|██                                                | 3901/90001 [13:37<6:08:20,  3.90it/s]
Empirical Reward Distribution: {0.1: 0.8482648725212465, 1.1: 0.13252478753541078, 2.1: 0.019121813031161474, 4.1: 1.475448536355052e-05, 8.1: 1.475448536355052e-05, 3.1: 5.901794145420208e-05}
  4%|██                                                | 4001/90001 [13:59<5:09:41,  4.63it/s]
```

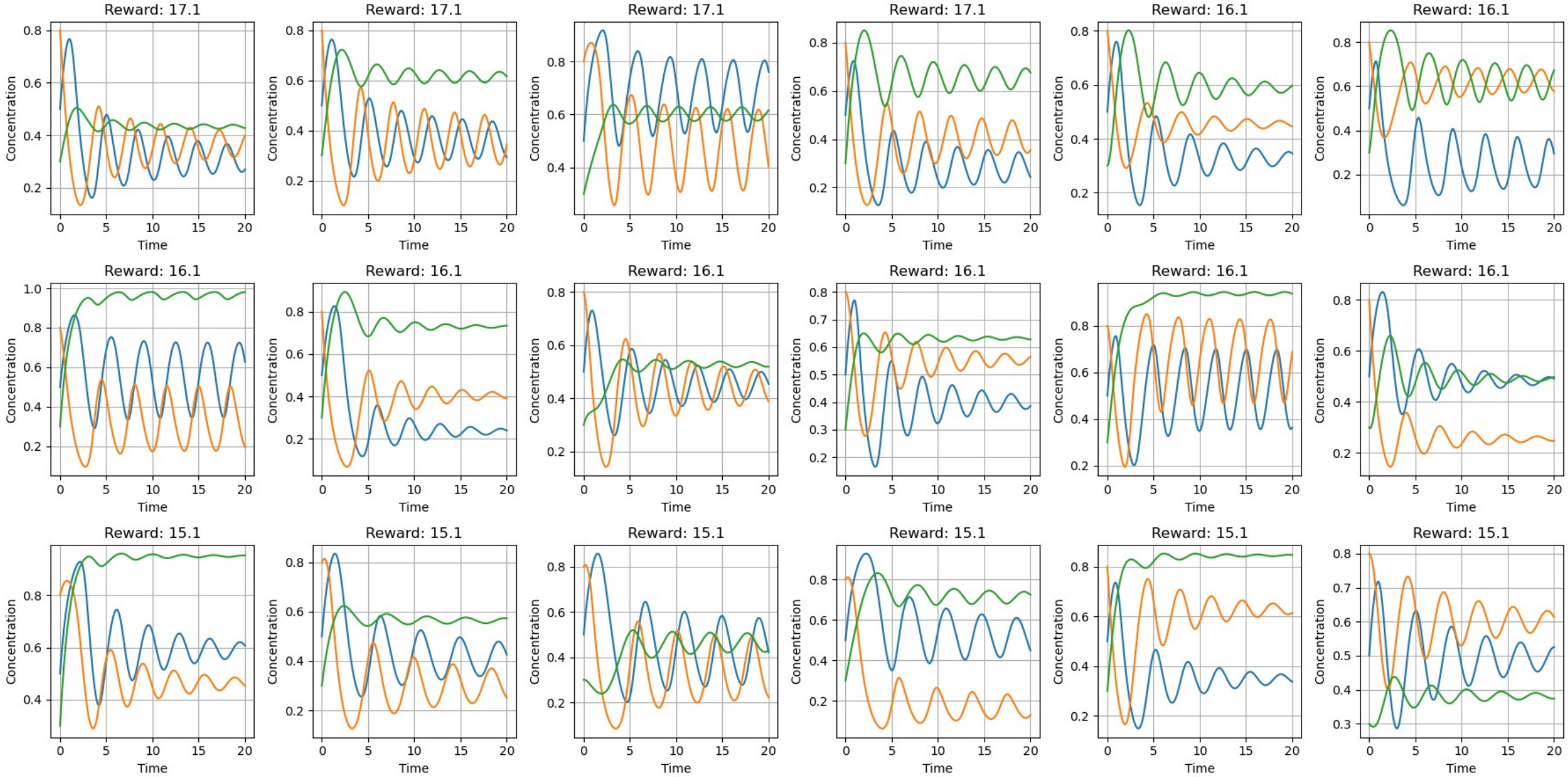
```
 40%|████████████████████                              | 36301/90001 [7:36:44<6:08:52,  2.43it/s]
Empirical Reward Distribution: {1.1: 0.28209, 0.1: 0.66285, 2.1: 0.04266, 3.1: 0.00655, 4.1: 0.00292, 10.1: 0.00013, 6.1: 0.00055, 12.1: 8e-05, 5.1: 0.00085, 7.1: 0.0005, 14.1: 4e-05, 8.1: 0.00019, 11.1
17, 9.1: 0.00027, 15.1: 3e-05, 16.1: 3e-05, 13.1: 6e-05, 20.1: 1e-05, 19.1: 1e-05, 17.1: 1e-05}
 40%|████████████████████                              | 36401/90001 [7:41:49<6:28:37,  2.30it/s]
Empirical Reward Distribution: {0.1: 0.66401, 1.1: 0.28091, 2.1: 0.04269, 3.1: 0.00657, 6.1: 0.00055, 5.1: 0.00086, 7.1: 0.00051, 10.1: 0.00012, 4.1: 0.00289, 14.1: 4e-05, 8.1: 0.00019, 11.1: 0.00017, 9
0027, 12.1: 7e-05, 15.1: 3e-05, 16.1: 3e-05, 13.1: 6e-05, 20.1: 1e-05, 19.1: 1e-05, 17.1: 1e-05}
 41%|████████████████████                              | 36501/90001 [7:42:55<6:35:19,  2.26it/s]
Empirical Reward Distribution: {0.1: 0.66566, 1.1: 0.2796, 2.1: 0.0424, 4.1: 0.0029, 14.1: 4e-05, 3.1: 0.00655, 7.1: 0.0005, 8.1: 0.0002, 6.1: 0.00054, 5.1: 0.00084, 11.1: 0.00017, 10.1: 0.00011, 9.1:
12.1: 7e-05, 15.1: 3e-05, 16.1: 3e-05, 13.1: 6e-05, 20.1: 1e-05, 19.1: 1e-05, 17.1: 1e-05}
```

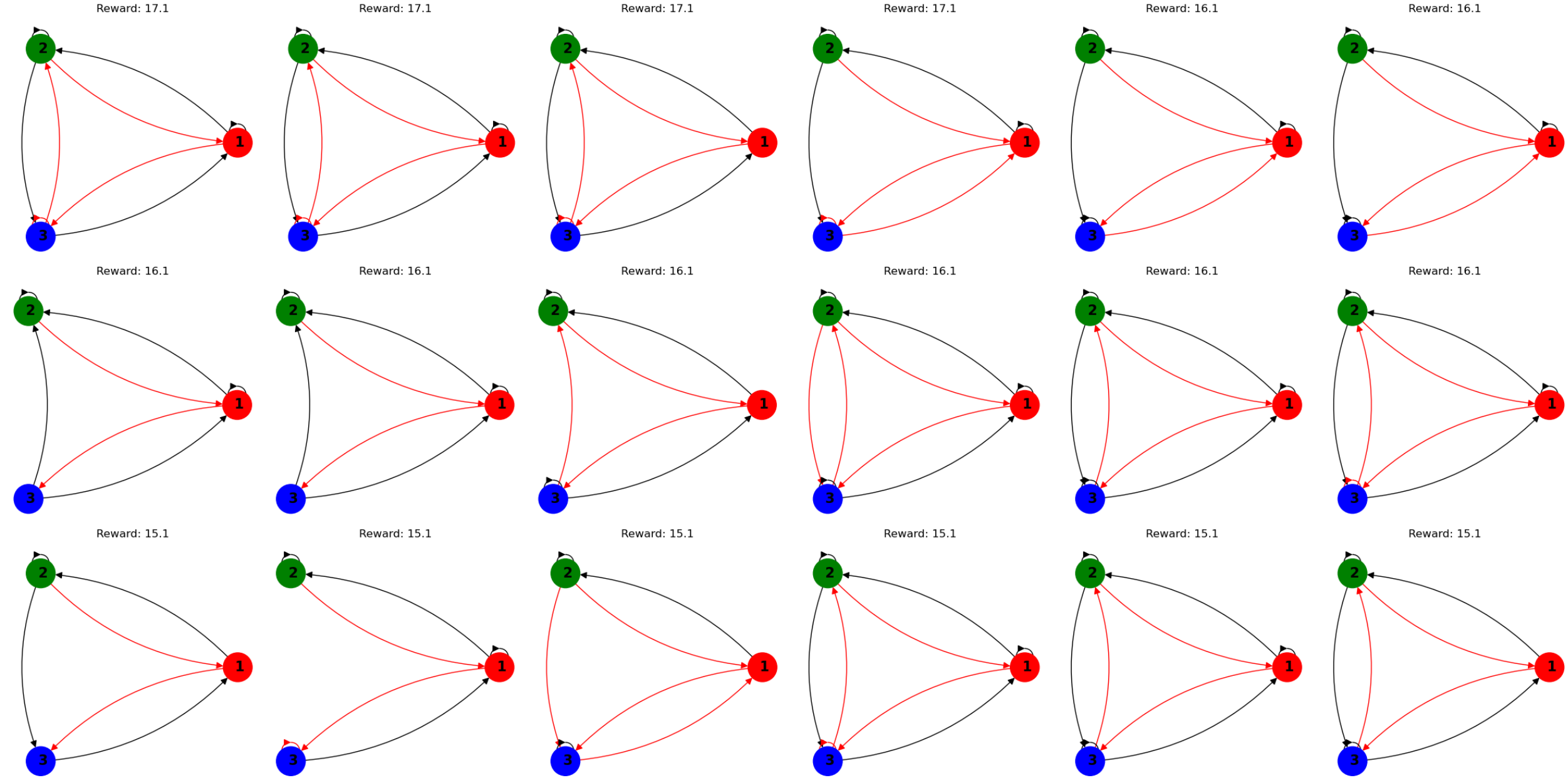

Training Losses Over Time
Loss 1
Loss 2
Loss 3
the sink found here is a black hole !!!
Loss
Training Step
0
5
10
15
20
0
5000
10000
15000
20000
25000
30000

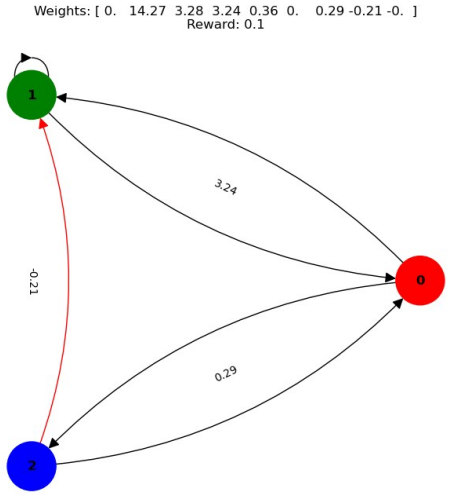
Top 10 States with Highest Rewards:
State 1: (18, 1, 16, 5, 16, 5, 1, 16, 1), Reward: 21.1
State 2: (17, 1, 16, 3, 14, 6, 4, 11, 7), Reward: 21.1
State 3: (18, 2, 17, 3, 14, 2, 5, 8, 10), Reward: 21.1
State 4: (18, 0, 17, 5, 14, 3, 3, 1, 8), Reward: 20.1
State 5: (18, 2, 17, 4, 15, 5, 3, 7, 13), Reward: 20.1
State 6: (19, 2, 15, 4, 13, 4, 5, 11, 10), Reward: 19.1
State 7: (18, 1, 18, 3, 13, 7, 5, 12, 0), Reward: 19.1
State 8: (16, 1, 17, 3, 15, 6, 3, 13, 5), Reward: 18.1
State 9: (16, 2, 19, 4, 15, 1, 2, 15, 0), Reward: 18.1
State 10: (17, 1, 16, 4, 15, 7, 4, 9, 10), Reward: 18.1
State 11: (17, 0, 16, 4, 16, 4, 5, 1, 8), Reward: 18.1
State 12: (17, 2, 17, 3, 13, 6, 2, 6, 0), Reward: 17.1
State 13: (16, 0, 16, 3, 16, 5, 1, 2, 8), Reward: 17.1
State 14: (14, 2, 16, 4, 15, 5, 1, 10, 5), Reward: 17.1
State 15: (15, 0, 17, 4, 14, 7, 5, 0, 8), Reward: 17.1
State 16: (17, 2, 15, 6, 14, 7, 4, 8, 8), Reward: 17.1
State 17: (17, 2, 15, 4, 14, 7, 1, 14, 14), Reward: 17.1
State 18: (19, 0, 17, 4, 15, 8, 5, 17, 2), Reward: 17.1
State 19: (19, 2, 17, 6, 13, 6, 5, 15, 4), Reward: 17.1
State 20: (17, 2, 18, 3, 13, 5, 6, 12, 1), Reward: 17.1
State 21: (19, 0, 17, 2, 15, 7, 7, 6, 11), Reward: 17.1
State 22: (16, 2, 18, 3, 15, 4, 0, 12, 1), Reward: 17.1
State 23: (13, 2, 17, 5, 17, 2, 0, 5, 9), Reward: 16.1
State 24: (17, 2, 17, 6, 17, 2, 0, 15, 18), Reward: 16.1
State 25: (18, 2, 16, 3, 15, 4, 2, 10, 9), Reward: 16.1
State 26: (17, 3, 17, 4, 13, 3, 2, 10, 0), Reward: 16.1
State 27: (16, 0, 17, 3, 14, 7, 14, 5, 15), Reward: 16.1
State 28: (17, 1, 16, 6, 15, 12, 17, 9, 9), Reward: 16.1
State 29: (19, 1, 17, 3, 14, 7, 4, 3, 6), Reward: 16.1
State 30: (16, 3, 18, 5, 15, 1, 4, 5, 12), Reward: 16.1
State 31: (17, 0, 15, 3, 16, 7, 0, 6, 0), Reward: 15.1
State 32: (14, 1, 17, 3, 15, 9, 0, 8, 15), Reward: 15.1
State 33: (17, 0, 16, 4, 15, 12, 0, 5, 16), Reward: 15.1
State 34: (16, 2, 16, 3, 15, 7, 3, 13, 1), Reward: 15.1
State 35: (15, 2, 17, 4, 13, 5, 4, 6, 6), Reward: 15.1
State 36: (16, 0, 17, 6, 15, 7, 1, 19, 18), Reward: 15.1

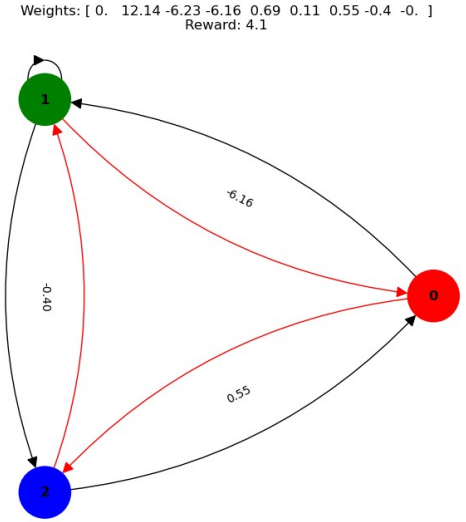


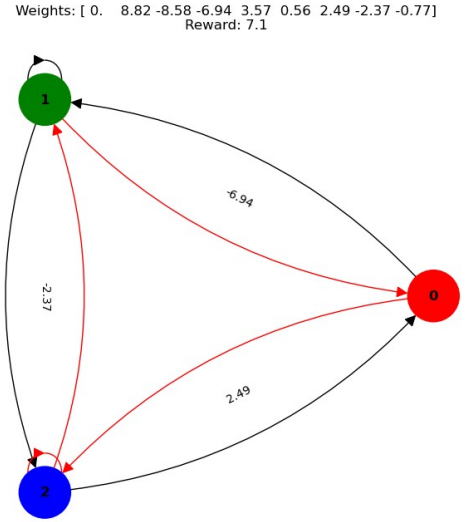

Reward: 21.1
Reward: 21.1
Reward: 21.1
Reward: 20.1
Reward: 20.1
Reward: 19.1
Reward: 19.1
Reward: 18.1
Reward: 18.1
Reward: 18.1
Reward: 18.1
Reward: 17.1
Reward: 17.1
Reward: 17.1
Reward: 17.1
Reward: 17.1
Reward: 17.1
Concentration
Time

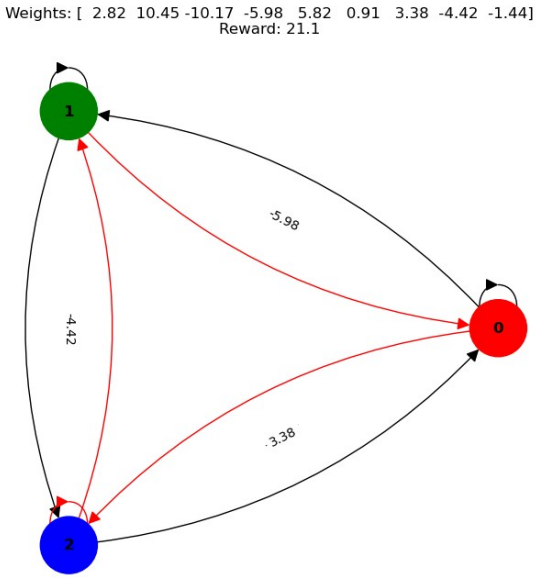

Reward: 17.1
Reward: 17.1
Reward: 17.1
Reward: 17.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 15.1
Reward: 15.1
Reward: 15.1
Reward: 15.1
Reward: 15.1
Reward: 15.1
Concentration
Time


Reward: 17.1
Reward: 17.1
Reward: 17.1
Reward: 17.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 16.1
Reward: 15.1
Reward: 15.1
Reward: 15.1
Reward: 15.1
Reward: 15.1
Reward: 15.1
1
2
3









State (15, 0, 8, 5, 9, 5, 0, 14, 0),
Weights [0. 14.27 3.28 3.24 0.36 0. 0.29 -0.21
-0.],
Reward 0.1

+6 +2 +1
State (15, 0, 14, 5, 11, 5, 1, 14, 0),
Weights [0. 12.14 -6.23 -6.16 0.69 0.11 0.55 -0.4
-0.],
Reward 4.1

+2 +3 +1
State (15, 0, 16, 5, 14, 5, 1, 15, 1),
Weights [0. 8.82 -8.58 -6.94 3.57 0.56 2.49 -2.37
-0.77],
Reward 7.1

+3 +1. +2 +1
Plotting Motif at Depth 2: State (18, 1, 16, 5, 16, 5, 1, 16,
1),
Weights [2.82 10.45 -10.17 -5.98 5.82 0.91 3.38 -4.42
-1.44],
Reward 21.1

**Extension of GFN:

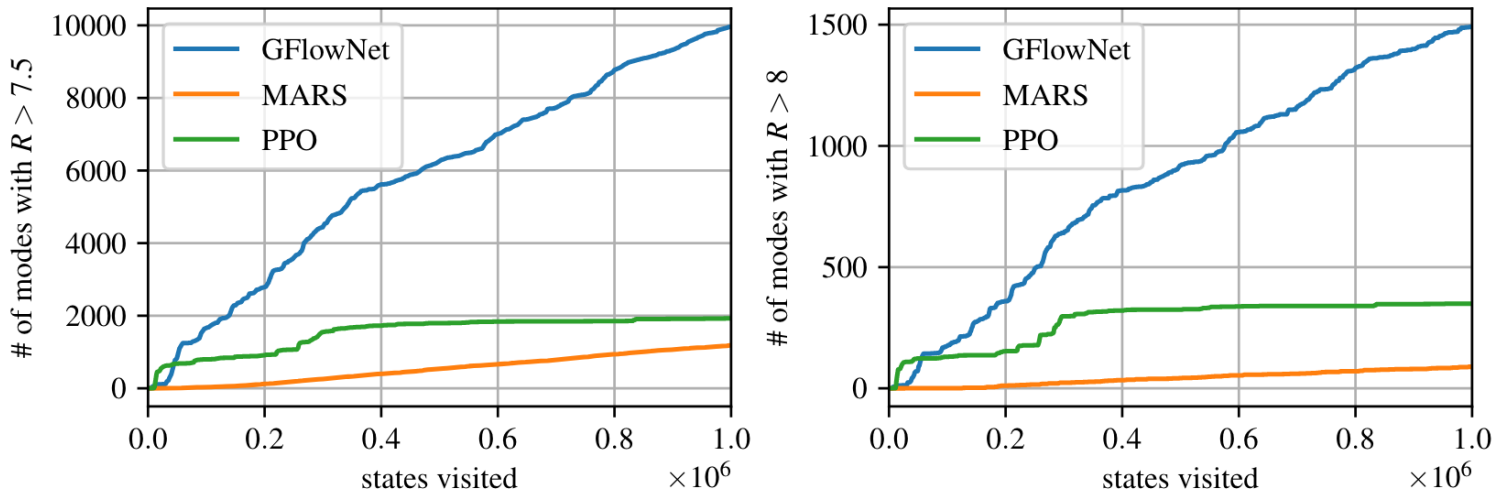


Figure 5: Number of diverse Bemis-Murcko scaffolds found above reward threshold T as a function of the number of molecules seen. Left, $T = 7.5$. Right, $T = 8$.

**Extension of GFN:

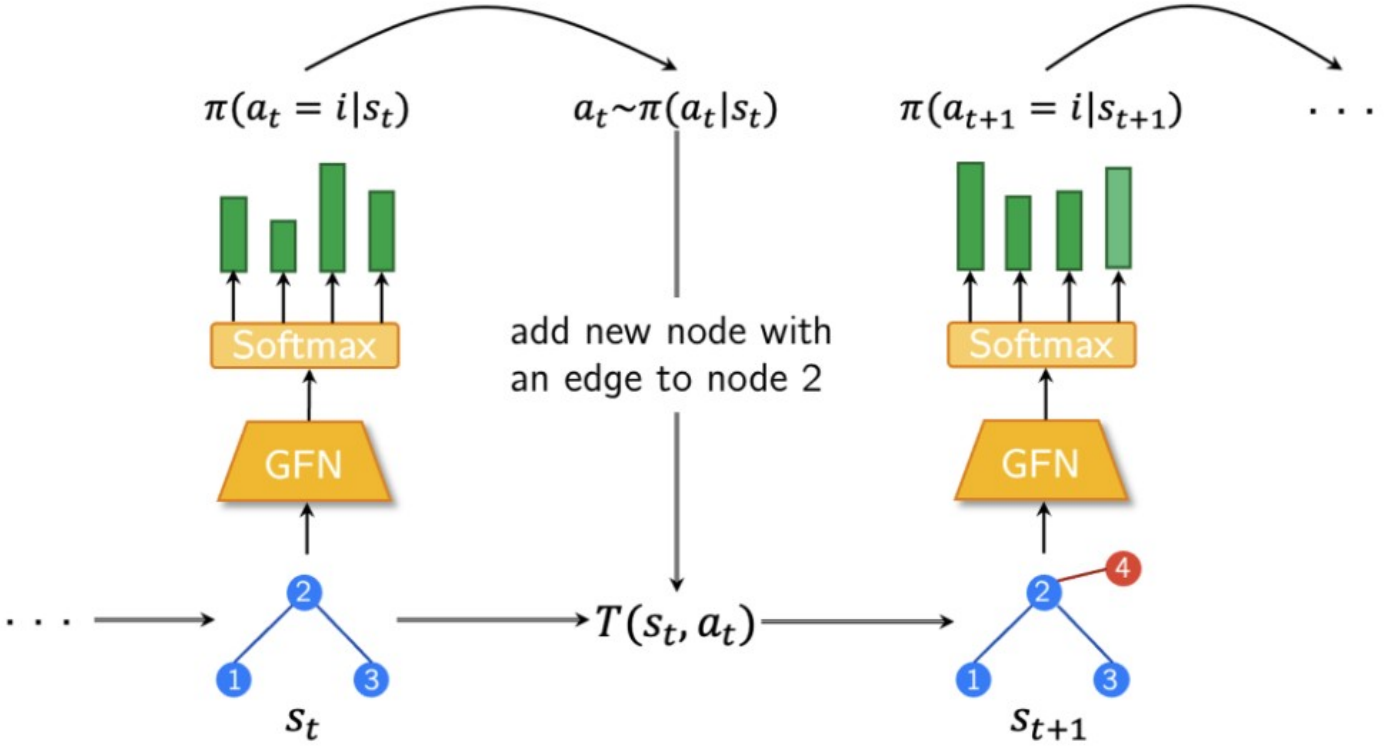


Figure 2: the most basic component of a GFlowNet is a neural network that defines its constructive policy, i.e., how to sample the next state s_{t+1} given the previous state s_t, through the choice of an action a_t. The new state s_{t+1} becomes the input for the next application of the GFlowNet policy denoted P_F or π, until a special "exit" action signals the end of the sequence, that an object $x = s_n$ has been generated, and a reward $R(x)$ is provided to encourage or discourage the policy from generating such a solution. The training objective optimizes the parameters of P_F towards making x be generated with probability proportional to $R(x)$.

**Extension of GFN:

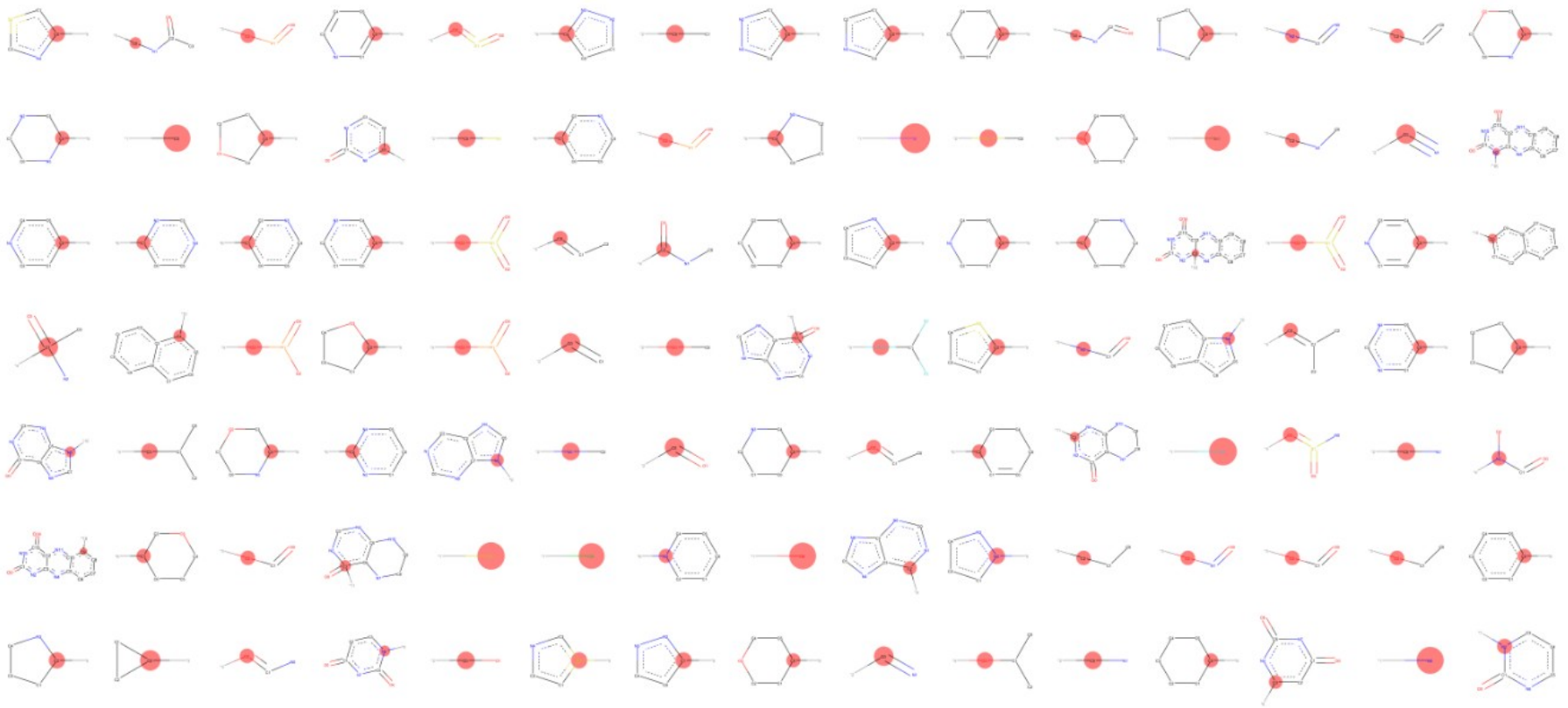


Figure 8: The list of building blocks used in molecule design. The stem, the atom which connects the block to the rest of the molecule, is highlighted.

**Extension of GFN:

(and more ……..x1000)