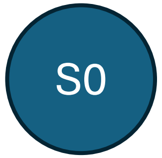
Reinforce Explanation for LUTEnv

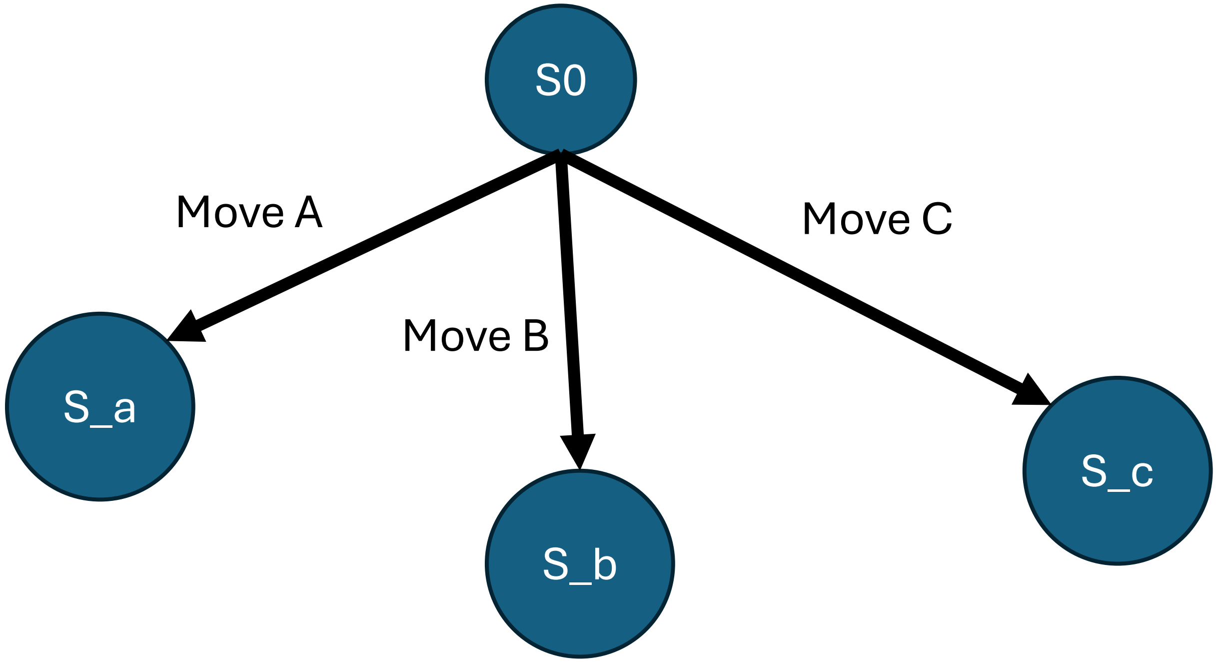
Step 1: You start at the initial netlist state



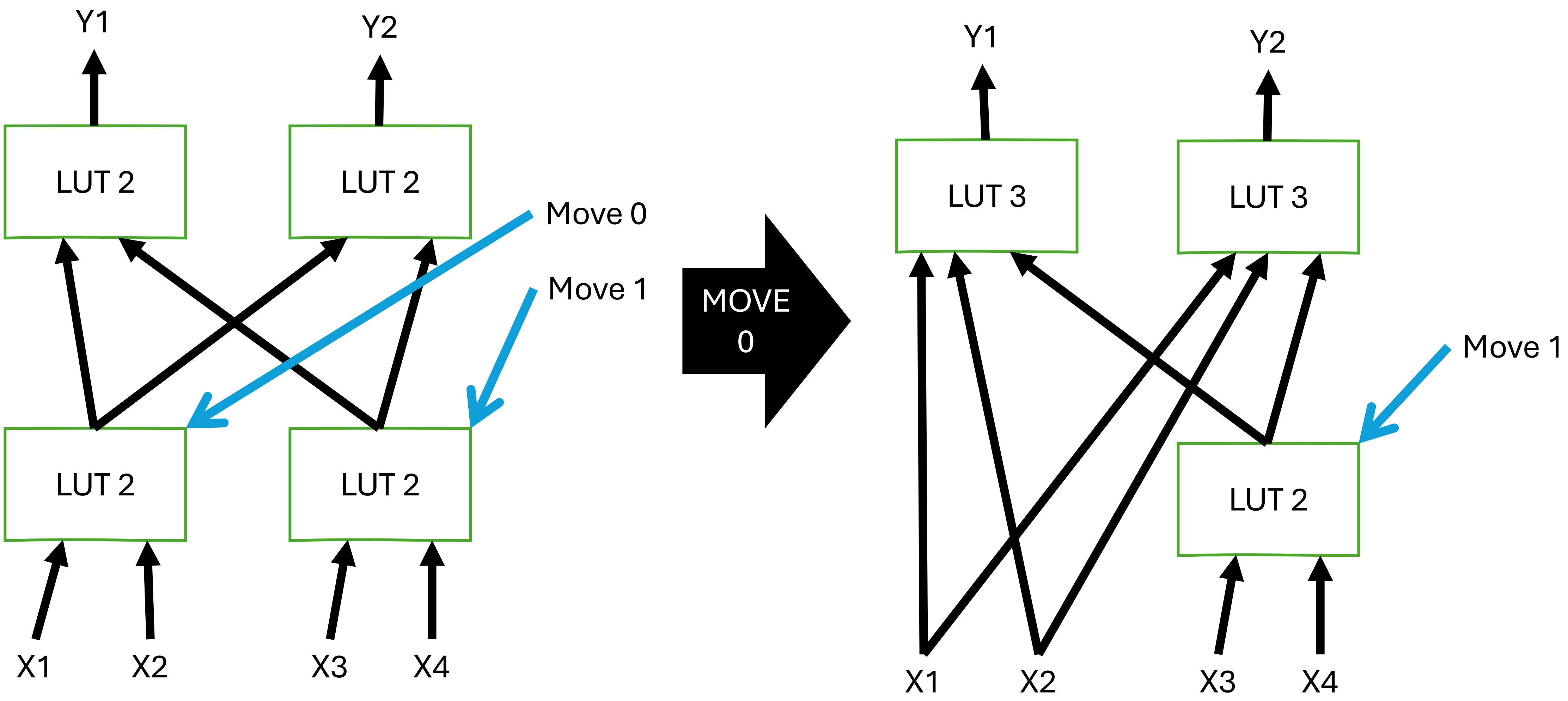
The netlist as read from the .BLIF files

Step 2: Each different move puts you in a different netlist state

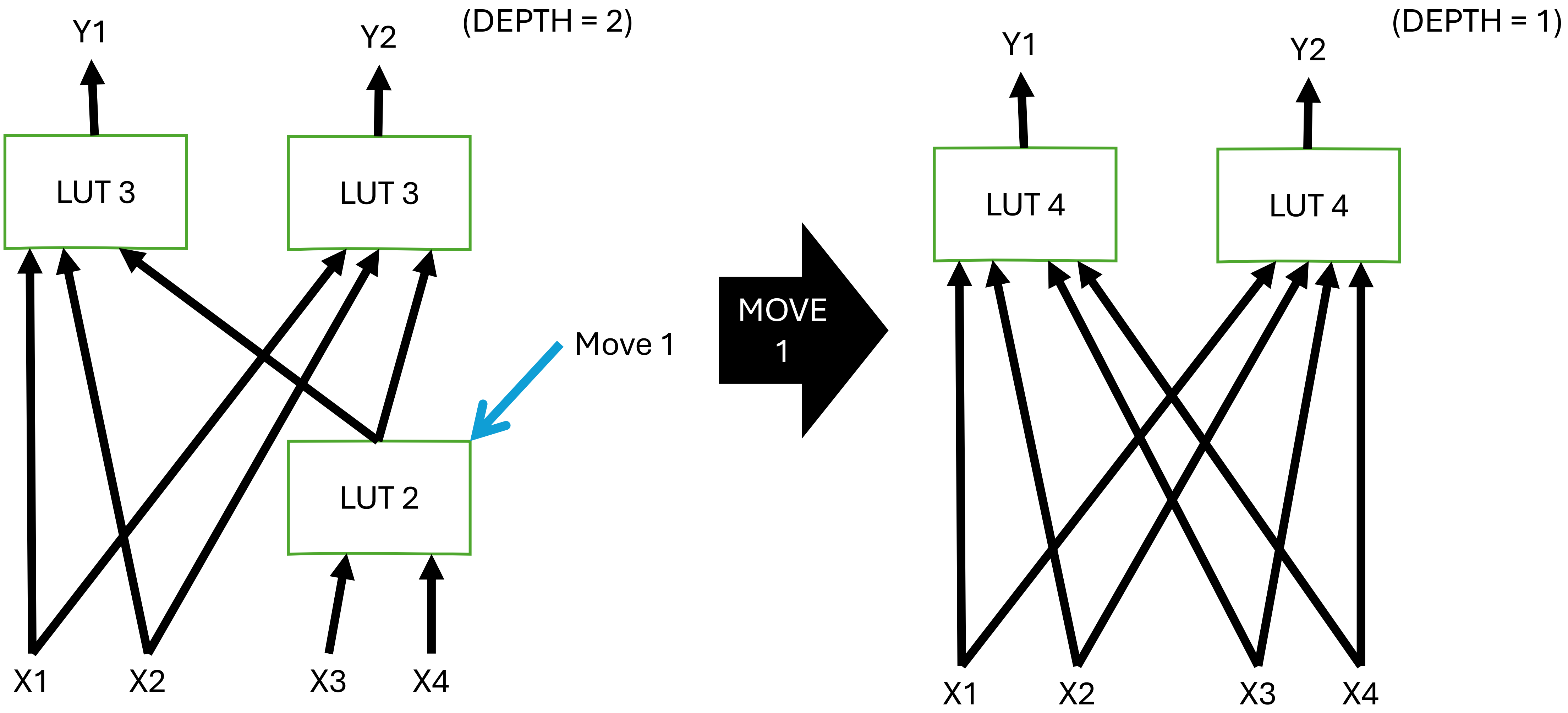
Reminder:
Move = LUT Merges
Which change the netlist
(new state)



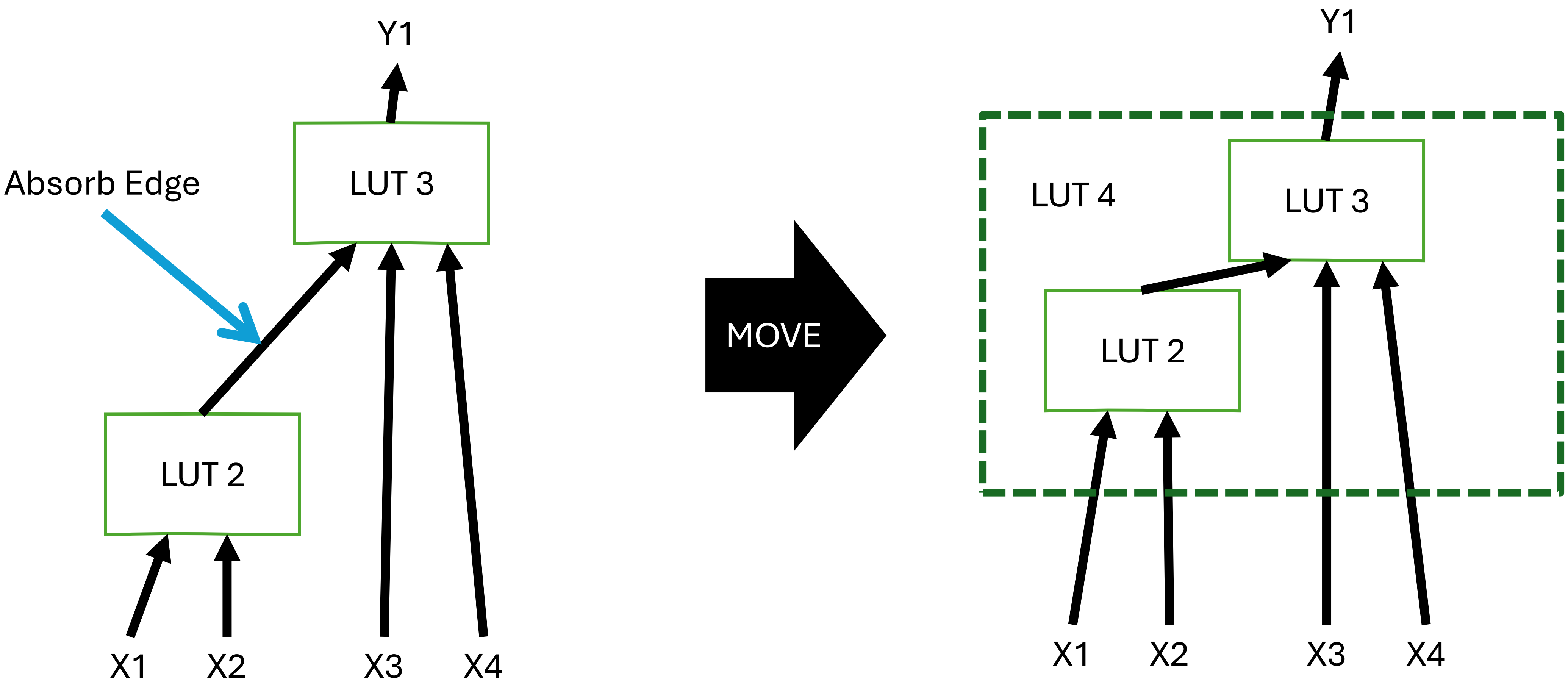
Concrete Representation Of Moves



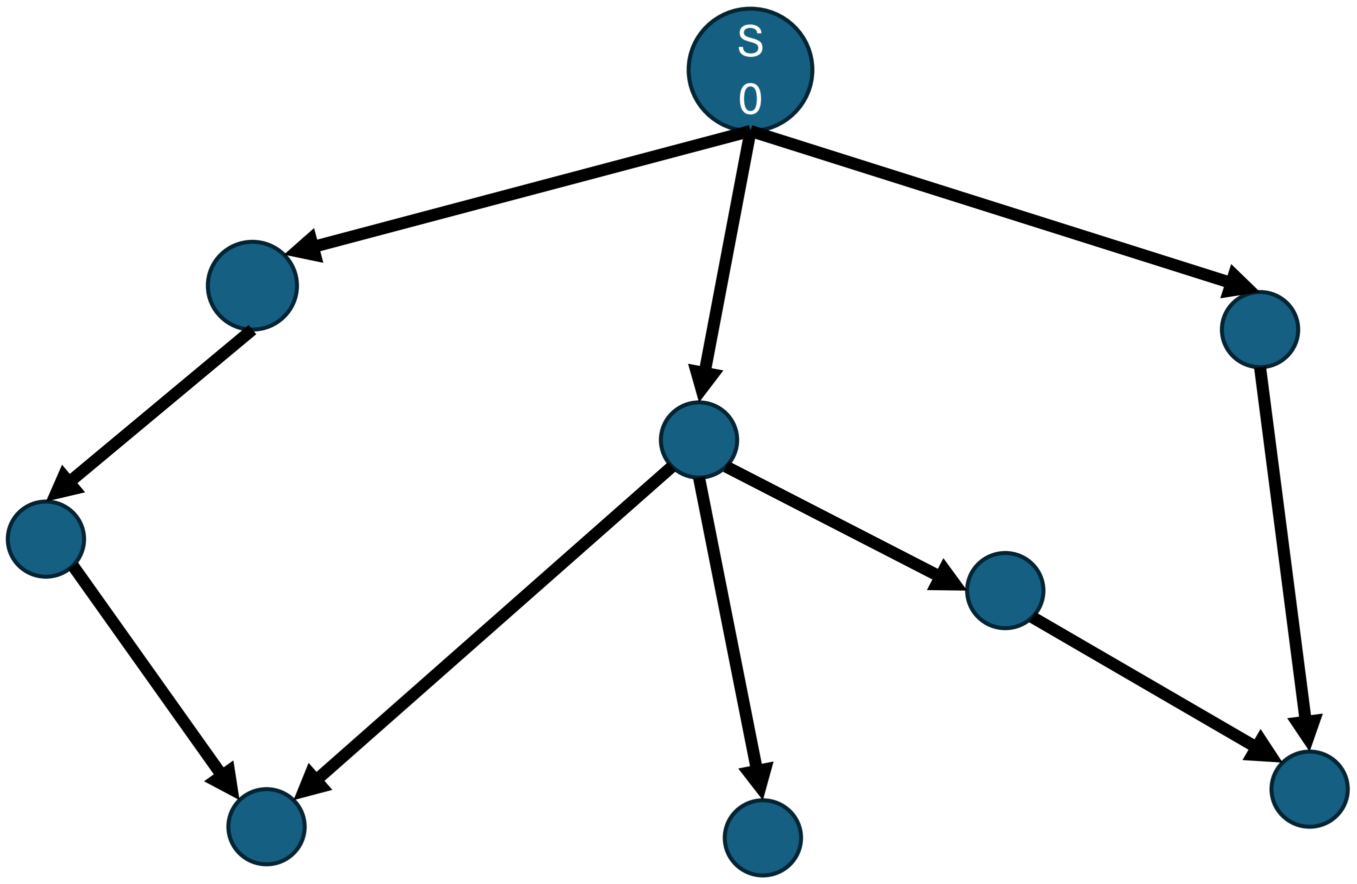
Concrete Representation Of Moves



How Functional Equivalence is Maintained



In general, this forms a Directed Acyclic Graph



Directed because Undo moves aren't provided

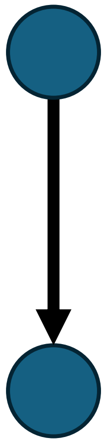
Graph and not tree because a state can be reached in multiple move orders

Some states have no moves

All remaining moves (LUT merges) create LUTs with
K > 6 and violate constraints.

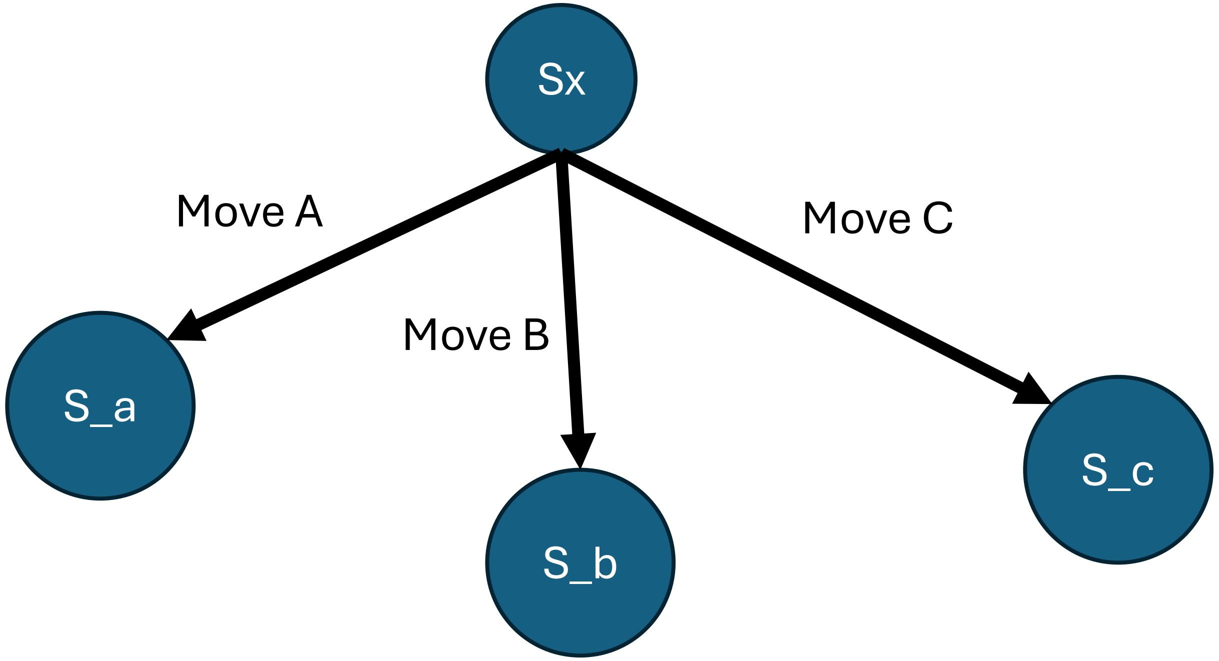
Therefore, the state is terminal

Terminal States have some given reward / value
Value = -Depth * #LUTs



A policy function will suggest moves for a state (Vector of probabilities)

$$\pi(s_x, \theta) = [p_A, p_B, p_c]$$



Policy gradients is an update to the weights

$$\nabla_\theta = \nabla_\theta^w \log(\pi(s_x, \theta))$$

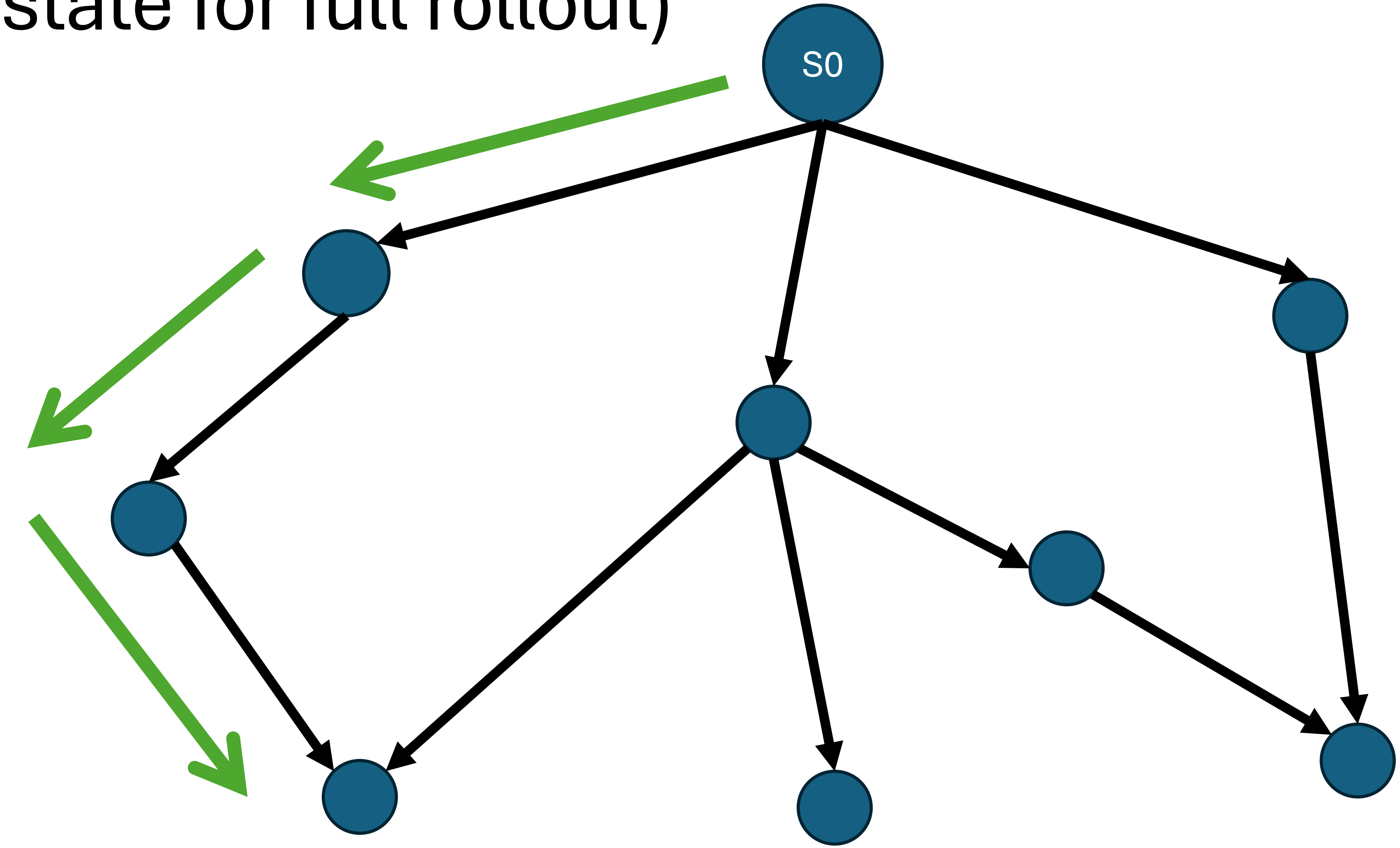
Is an update to the weights that makes

p_w (probability move is move w) more likely (in state s_x)
probability move is everything else less likely (in state s_x)
When added to θ

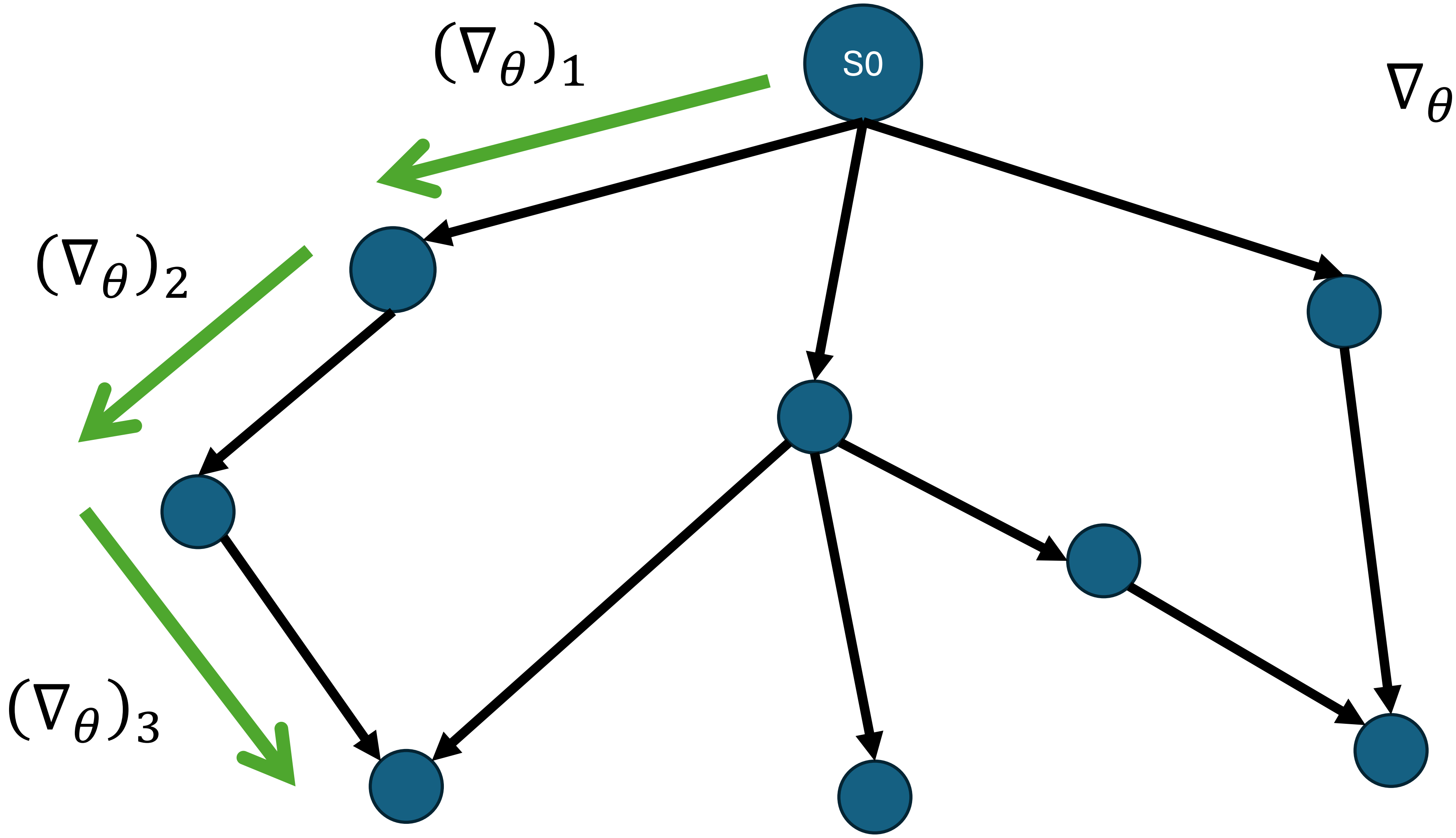
The gradient is taking w.r.t log probability for mathematical reasons.

$\theta_{after} = \theta_{\mathrm{before}} + \nabla_\theta$: Move w is made more likely in state s_x

During an episode you take some fixed number of random steps (or until you reach some terminal state for full rollout)



Accumulate the gradient over those steps



$$\nabla_\theta = \sum_i (\nabla_\theta)_i$$

Consider the rewards of that episode

R= Rewards earned in episode (- Final LUTs * Final depth)

B = Baseline rewards earned by policy (Assume you have some way of computing how much reward is “typically earned”)

A = Advantage = $R - B$

$A > 0$: The moves were better than expected

$A < 0$: The moves were worse than expected

Update the policy based on advantage

$$\theta_{final} = \theta_{initial} + A * \nabla_{\theta}$$

If $A > 0$: All moves made are updated to be made more likely.

If $A < 0$: All move made are updated to be made less likely.

You might use ADAMW or other optimizers, this is simple gradient ascent.

Baselines

Baseline can be done in many ways:

- Training and managing a value function to estimate baseline
- Population of episodes and computing mean and std deviation of rewards in the population.
- (In this lab) exponential moving averages of rewards earned in previous episodes.

Policy Gradient Methods are All variants of this Formulation

Variants can be quite sophisticated:

- Dynamic policy updates during an episode
- Off – policy existing reward data (Difficult to do in practice)
- Managing step sizes and proximity to original policy
- Managing reward credit assignment to particular moves
- …