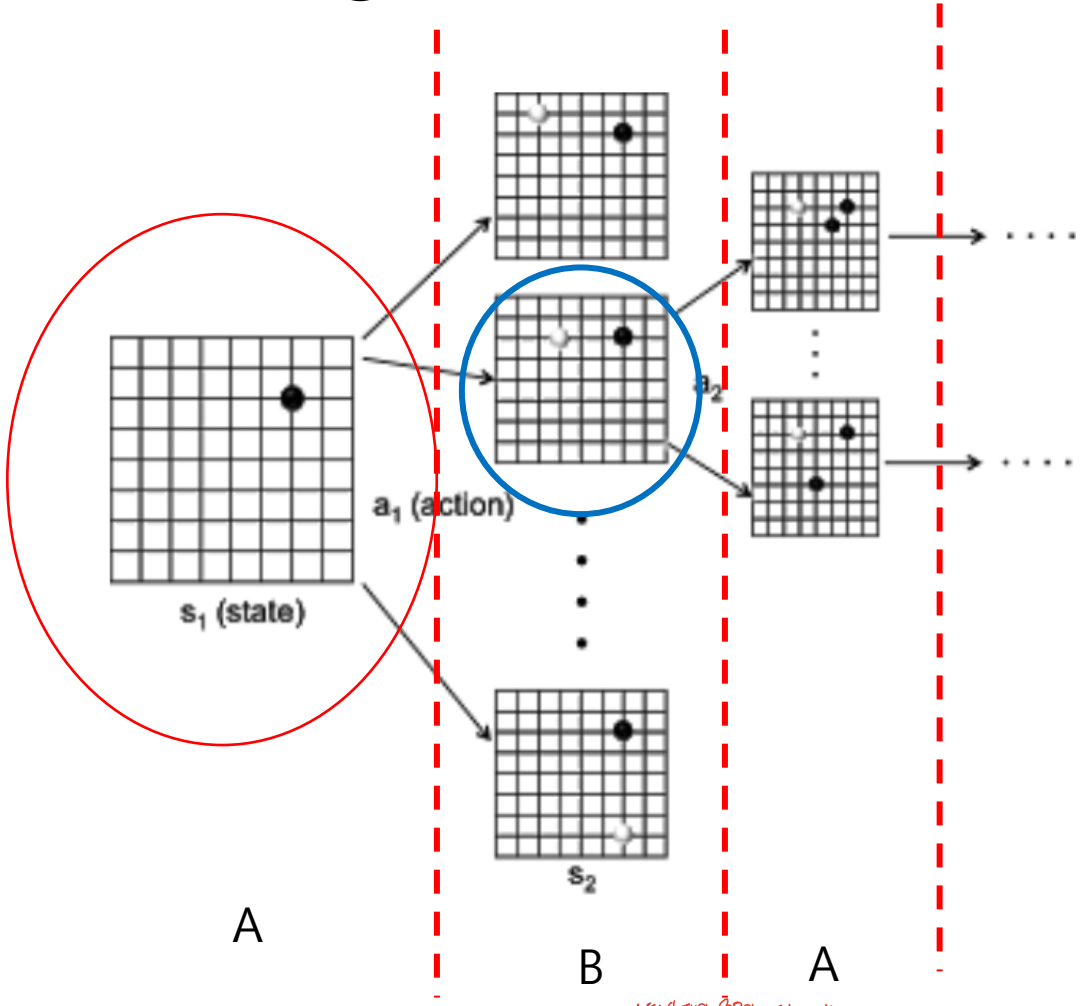
Alpha GO

(외부자료)

“자신이 없어요. 질 자신이요”

edge에 방향이 없다고 보고, 시작 node를 root 노드로 보아서 ➔ Tree 라고도 함.

Game Tree: a directed graph (or tree) whose nodes are positions in a game and whose edges are moves.



A : white stone 놓을 차례
B : black stone 놓을 차례

Monte Carlo Tree Search: 완벽한 Game Tree를 만들지 않고, fanout과 height가 제한된 game tree를 만들어서 next move를 결정하는 방법 중 하나.

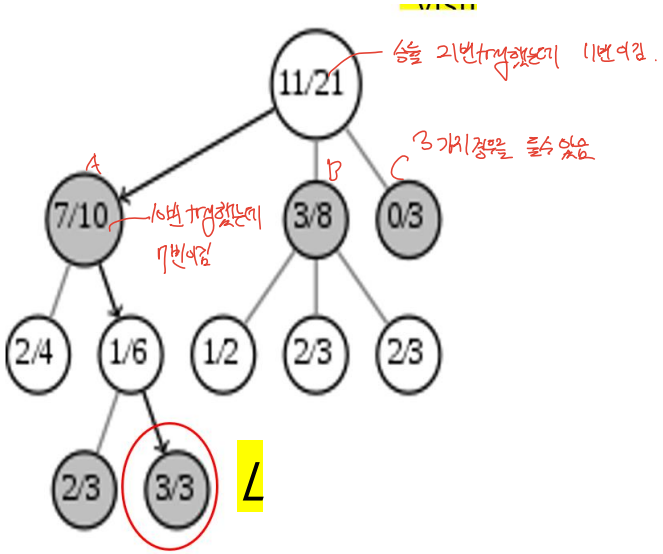
어떤상태가 좋은지 알아내는 방법중 몰테 카르로 방법이 있음

What is Monte Carlo Tree Search (MCTS) ?

- **Monte Carlo Tree Search Basics**
  - Given a **game state, it** chooses **the most promising next move.**
  - In each playout (simulation), the game is played out to the very end by selecting moves at random.

특정 node(게임 state)에서 출발해서, 2명의 player가 어떤 policy (일반적으로 random move를 이용)에 입각해서 번갈아 가면서 연속적으로 (승패가 날 때까지 또는 유리/불리를 판단할 때까지) move를 해 보는 것. 나중에 마지막 게임 state가 어떤 player에게 유리한 지를 판단해 보게 된다 .

Monte Carlo Tree Search (MCTS)

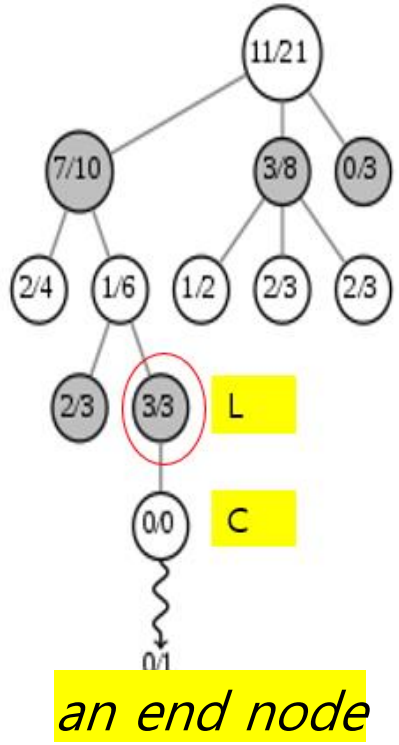


- *Selection*: start from root *R* and select successive child nodes down to a node *L* that is **not fully expanded**.

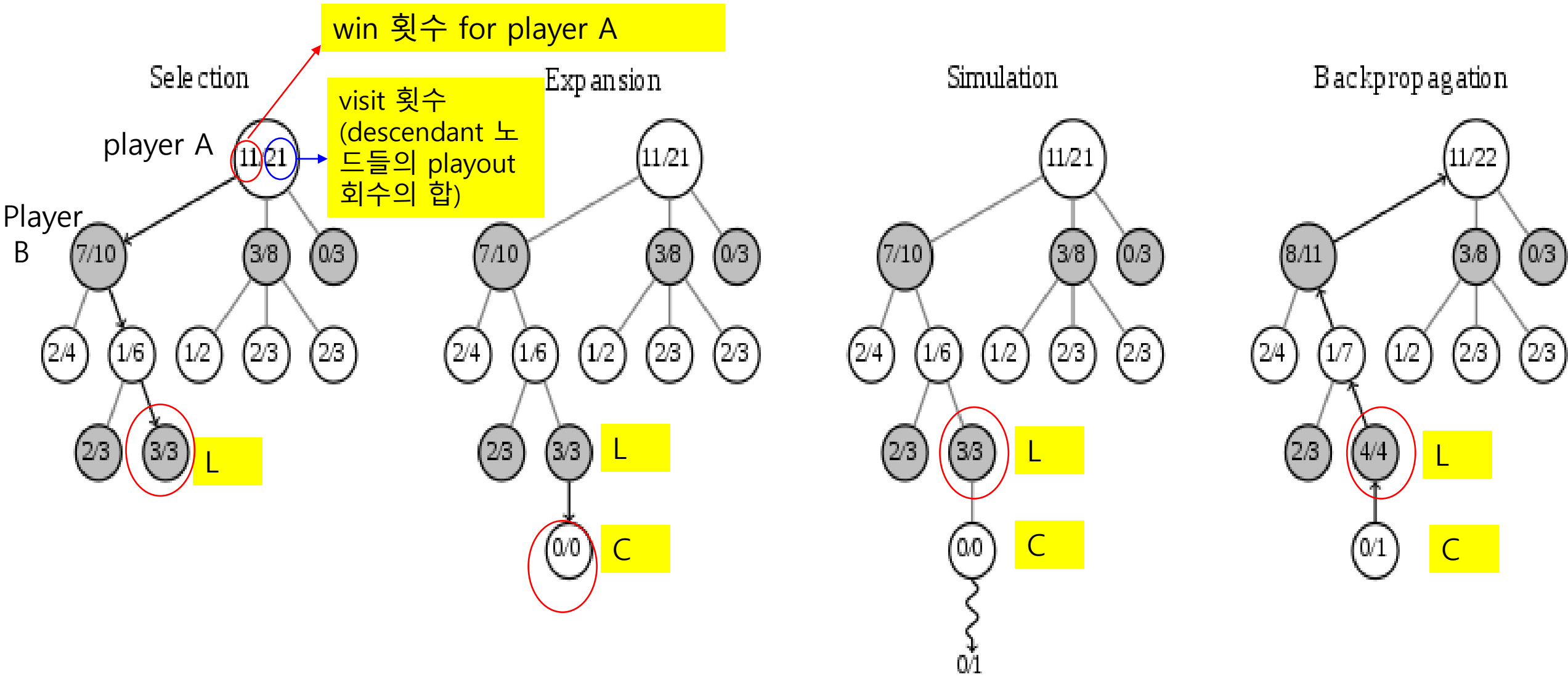
- *Expansion*: unless *L* ends the game with a win/loss for either player, create one child node C and add C to L.

- *Simulation*: play a random playout from node *C down to an end node*.
  This step is sometimes also called *playout* or *rollout*.

- *Backpropagation*: use the result of the playout to update information (visit count, win count, etc) in the nodes on the path from *C* to *R*.



Monte Carlo Tree Search (MCTS)



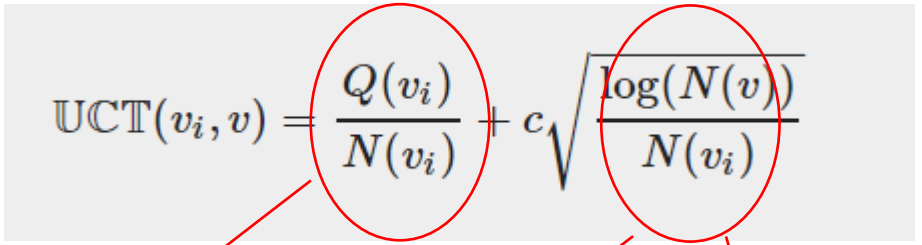
- In Selection step, **node maximizing UCT is the one to follow during MCTS traversal**.

- for a child node vi of a node v,

  Q(vi): win count of node vi

  N(vi): visit count of node vi

  N(v): visit count of node v



win count가 큰 노드에게 유리

This second term is the *exploration term*. This term increases for a given node when it hasn't been visited very much relative to the number of visits to the parent node.

playout 수가 적은 (아직 시도하지 않은 move 를 많이 가진) 노드에게 유리

- During simulation (playout, rollout) the moves are chosen with respect to a function called rollout policy function:

$$\mathrm{RolloutPolicy} : s_i \rightarrow a_i$$

- Default rollout policy function is a **uniform random**.

Alpha Go

알파고 전까지만 설명위

node v에서 child node vi를 선택했을 때 얼마나 좋은 지를 측정하는 방법

UCT in Alpha GO: for a child node vi of a node v

$$\mathrm{UCT}(v_i, v) = Q(v_i) + cP(v, v_i)\sqrt{\frac{N(v)}{1 + N(v_i)}}$$

P(vi, v) is prior probability of the move (transition from v to vi), its value comes from the output of deep neural network called **SL Policy Network**. Policy Network is a function that consumes game state and produce probability distribution over possible moves.

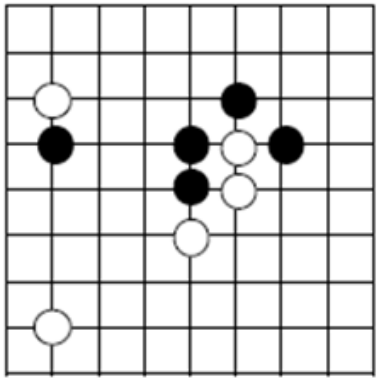
Q(vi) : mean reward for selecting vi ➔ $1/N(v_i)\sum_{s_L \mid v_i \to s_L} V(s_L)$

S_L 노드는 각 시뮬레이션에서 vi 노드로부터 시작해 도달하는 노드
$V(S_L)$은 뒤에 정의함.

P(vi, v)

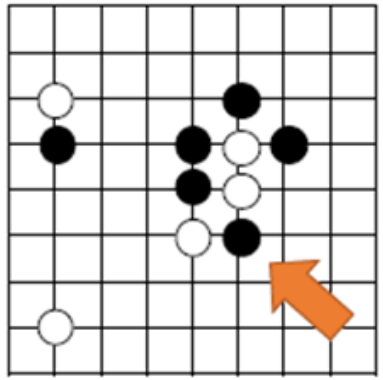
프로 바둑기사 따라하기 (supervised learning)

현재 판



예측 모델

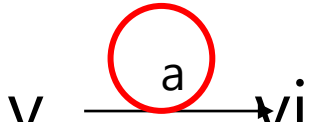
다음 판



Learning: **P (next action | current state)**

= P (a | s)

v를 s 라고 하자



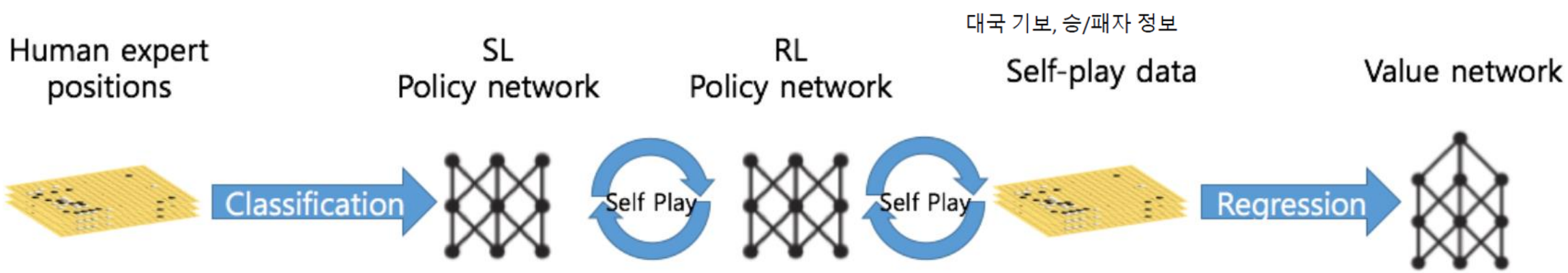
Playout/Simulation is Alpha Go

- Evaluation of node S_L is a weighted sum of two components:

$$V(S_L) = (1-\alpha)v_0(S_L) + \alpha z_L$$

  - zL : standard rollout evaluation with custom fast rollout policy.
  - v0($\mathbf{S_L}$): position evaluation given by 13-layer convolutional neural network v0 called **Value Network**.
  - Value Network는 S_L 의 대한 승률을 예측함

* The **RL Policy Network** is used to generate 30 min positions (self plays) dataset for Value Network training (the one used for game state evaluation)

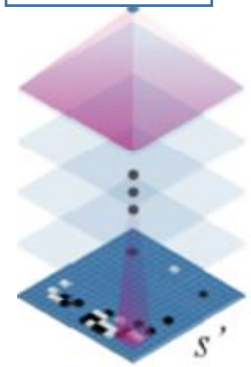


P (a | s)

P (a | s)

RL policy network은 승리하면 +1, 패배하면 -1의 reward score를 부여하는 강화학습을 하는데, P(a|s) 확률을 구하도록 학습

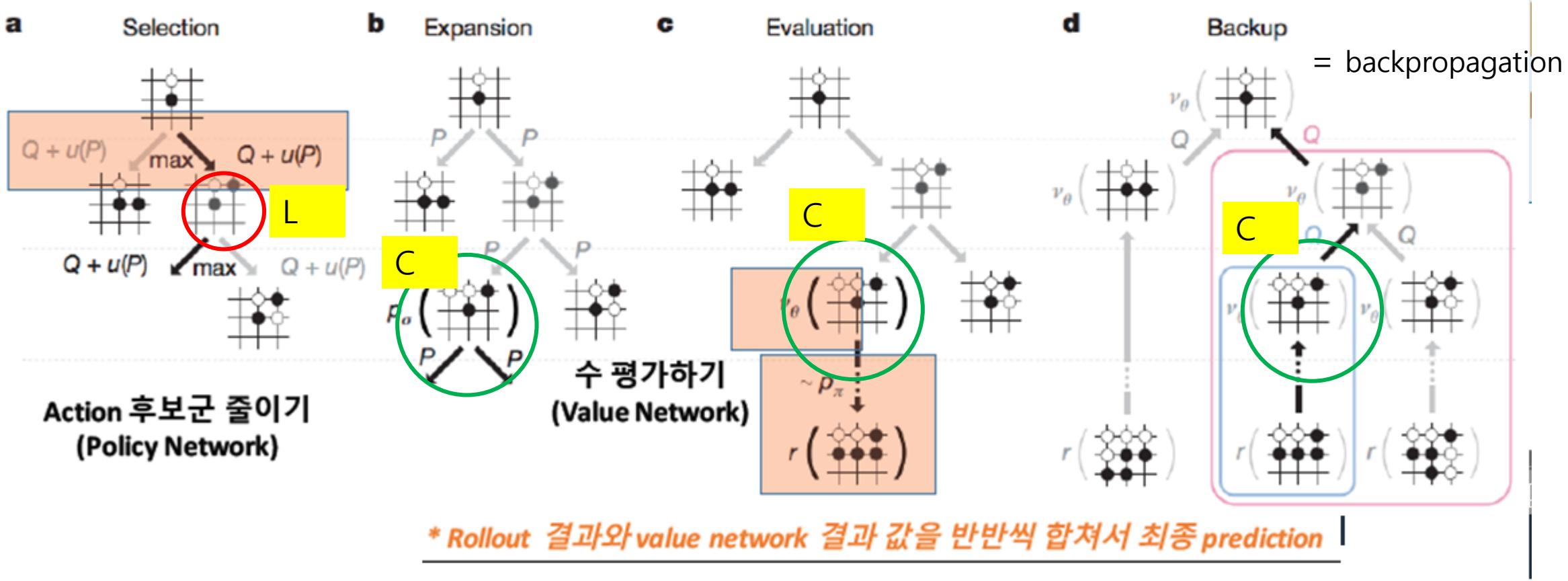
v0($\mathbf{S_L}$):



0~1 사이의 값으로 예측
1에 가까우면 좋은 판세
0에 가까우면 좋지 않은 판세

승/패

수 읽기 (w/ Monte Carlo Search Tree)



O + u(P)는 앞에서 설명한 UCT(vi, v)를 나타냄

음성 설명 없음

Reference

- MCTS: https://int8.io/monte-carlo-tree-search-beginners-guide/

강의 종료