CS 480

Introduction to Artificial Intelligence

September 15, 2022

Announcements / Reminders

- **Please follow the Week 04 To Do List instructions**
- **Quiz #01 due on Sunday (09/18/22) at 11:00 PM CST**
- **Written Assignment #01 due on Tuesday (09/20/22) at 11:00 PM CST**
- **Programming Assignment #1 will be posted within 1.5 - 2 weeks**
- **Midterm Exam (<u>consider fixed</u>):**
  - **October 13th, 2022 during lecture time**

Plan for Today

- **A* Heuristics revisited**
- **Problem Solving: Adversarial Search**

A* Algorithm: Evaluation Function

Calculate / obtain:

$$f(n) = \mathbf{g(State_n)} + \mathbf{h(State_n)}$$

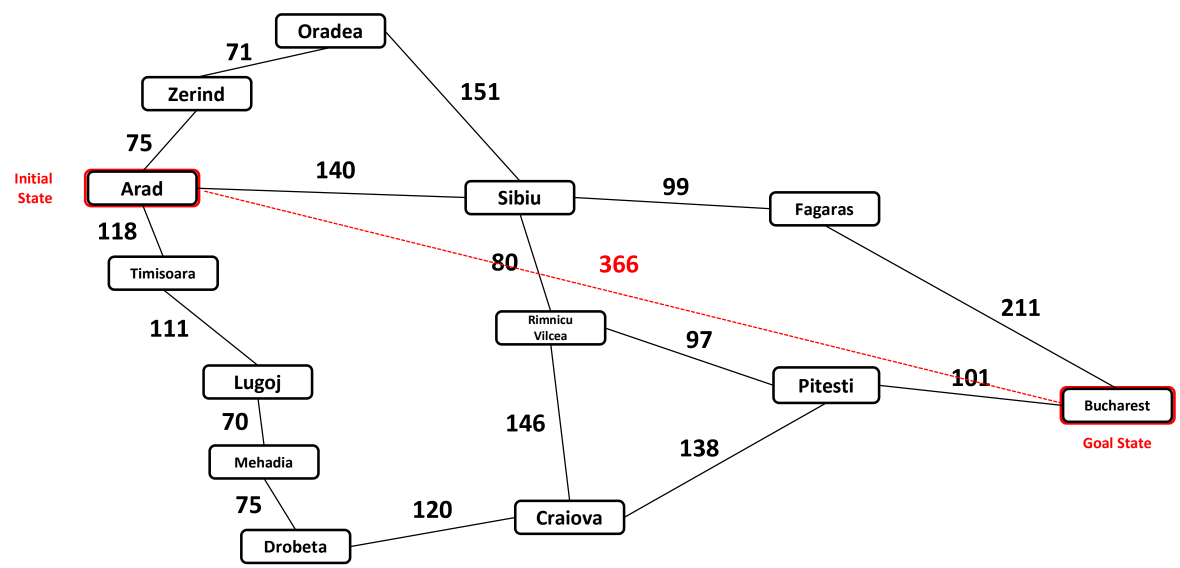
where:

- **g(n) - initial node to node n path cost**
- **h(n) - estimated cost of the best path that continues from node n to a goal node**

A state n with minimum (maximum) f(n) should be chosen for expansion

What Made A* Work Well?

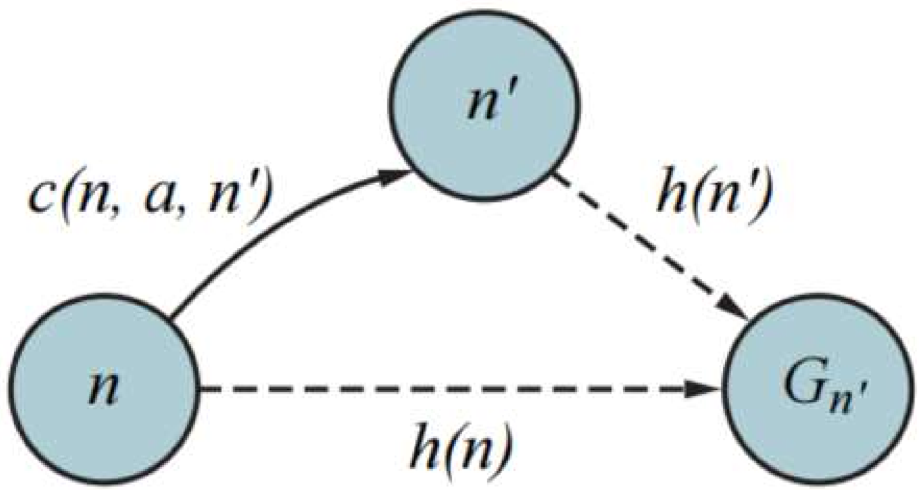
- **Straight-line heuristics is admissible: it never overestimates the cost.**



- **An admissible heuristics is guaranteed to give you the optimal solution**

What Made A* Work Well?

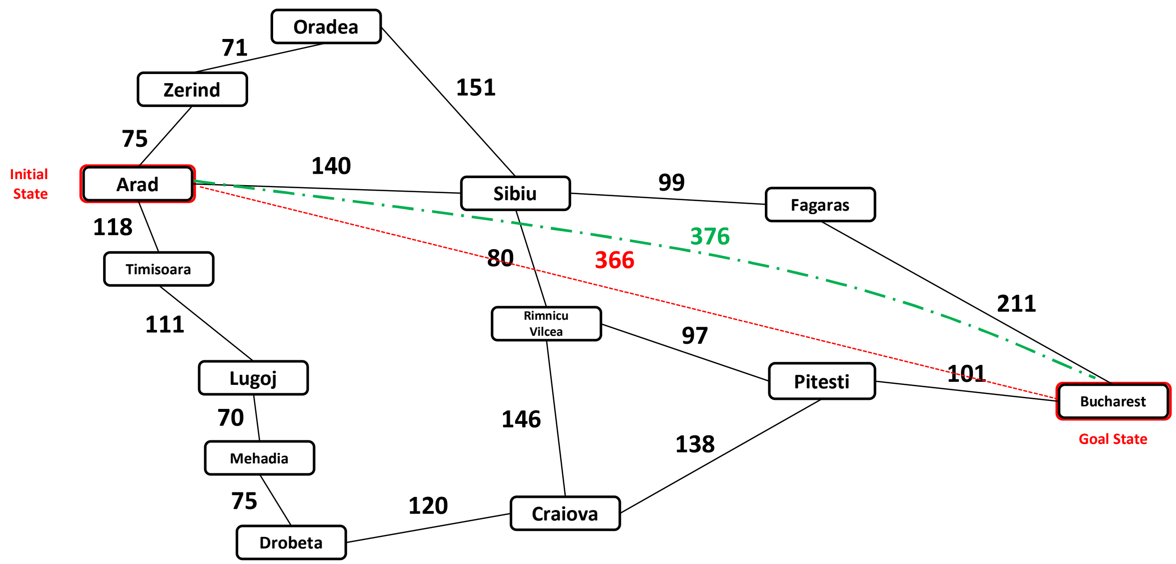
- **Straight-line heuristics is consistent: its estimate is getting better and better as we get closer to the goal**



- **Every consistent heuristics is admissible heuristics, but <u>not the other way around</u>**

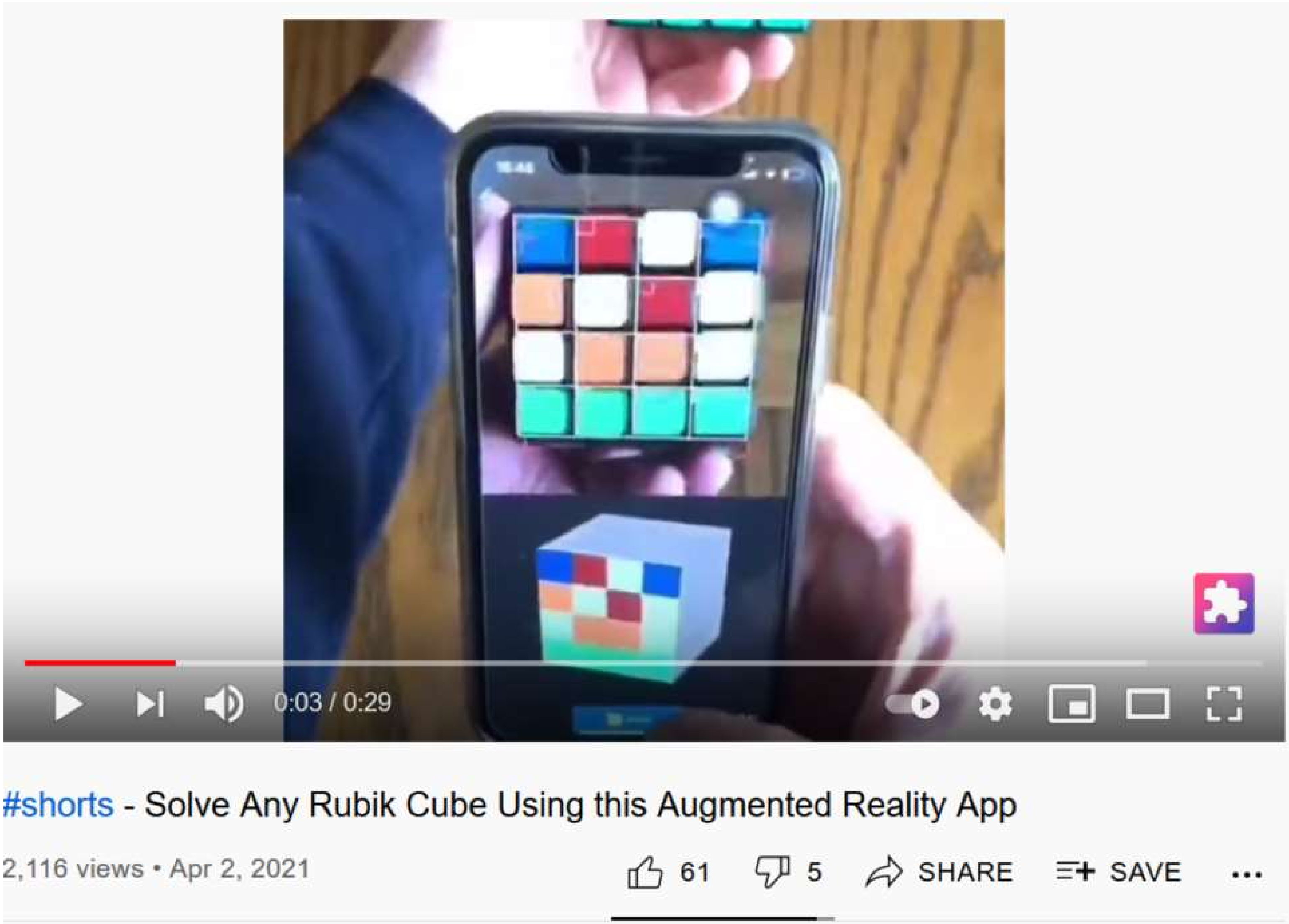
Dominating Heuristics

- We can have more than one available heuristics. For example $h_1(n)$ and $h_2(n)$.



- Heuristics $h_2(n)$ estimate is closer to actual cost than $h_1(n)$. $h_2(n)$ dominates $h_1(n)$. Use $h_2(n)$.

Informed Search: Application Example



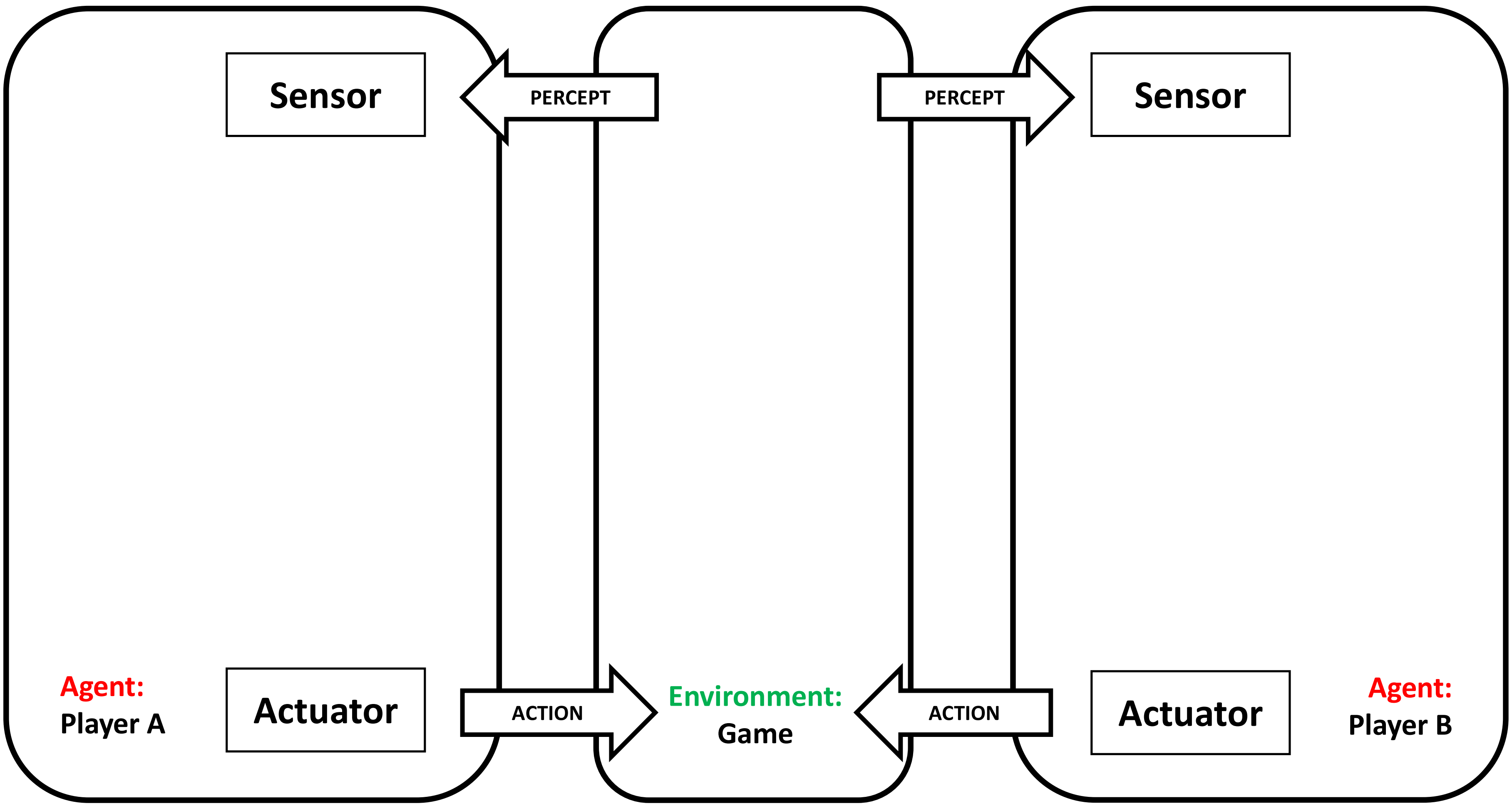
Source: https://www.youtube.com/watch?v=Pxbv2gEhnMk

Selected Searching Algorithms

- Searching Algorithms
  - **Uninformed** Search
    - Breadth First Search
    - Depth First Search
    - Brute Force Search
    - Uniform Cost Search
  - **Informed** Search
    - Greedy Best First Search
    - Branch and Bound
    - A* Algorithm
    - Hill Climbing Algorithm
    - Constraint Satisfaction Search
    - α - β Pruning
    - Min-Max Search

Uninformed search: an agent <u>has no estimate</u> how far it is from the goal
Informed search: an agent <u>can estimate</u> how far it is from the goal

Two-player Games



Perfect Information Zero Sum Games

- Perfect information = fully observable
- Multiagent: number of players is 2 or more
- Multiagent: agents are competitve
- Zero-sum: “winner takes all”
- Examples:
  - Tic Tac Toe
  - Chess

Two Player Games: Env Assumptions

Works with a "Simple Environment":

- Fully observable
- ~~Single agent~~ Mulitagent (competitive!)
- Deterministic
- Static
- Episodic / sequential
- Discrete
- Known to the agent

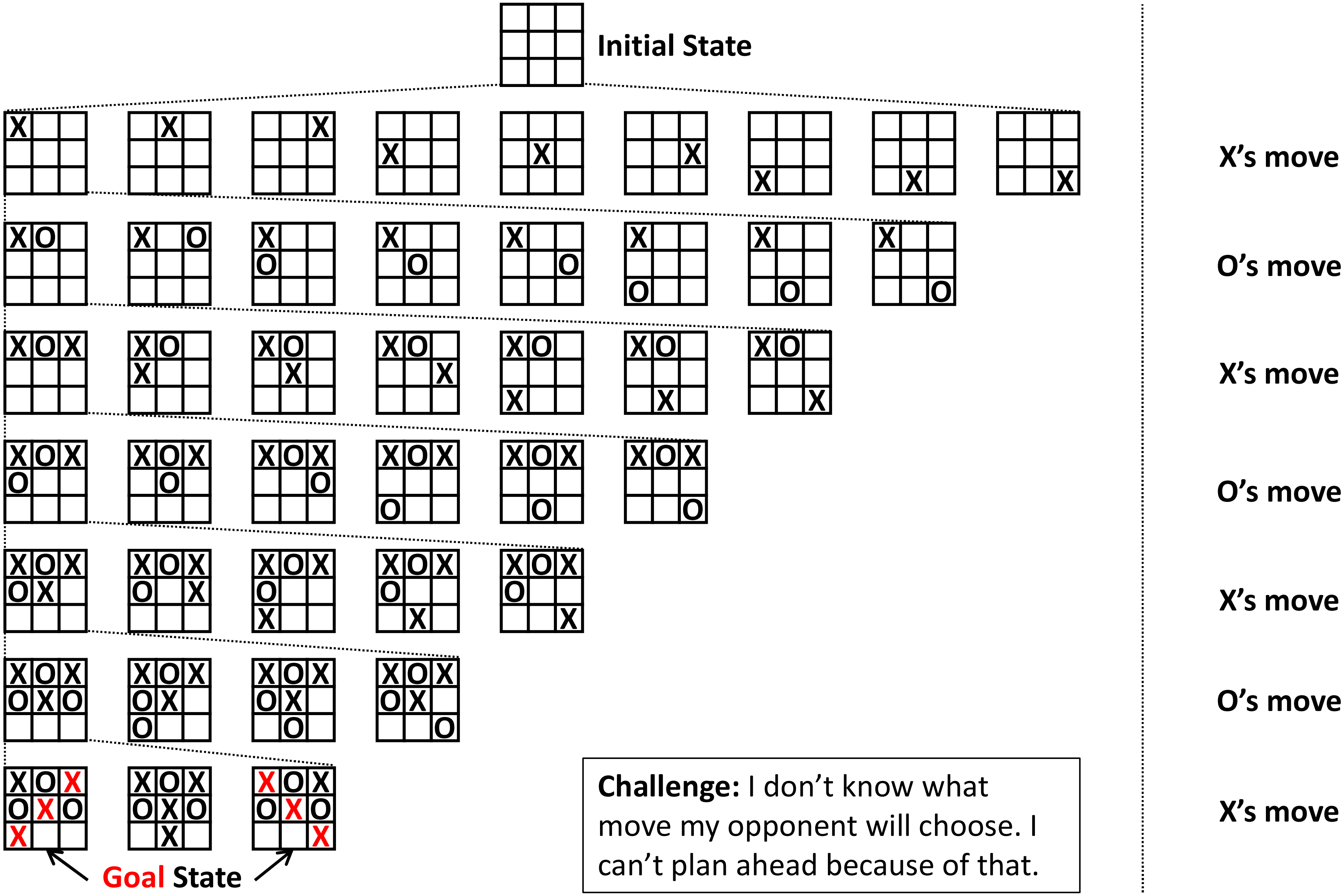
Defining Zero Sum Game Problem

- Define a set of possible states: **State Space**
- Specify how will you track **Whose Move / Turn** it is
- Specify **Initial State**
- Specify **Goal State**(s) (there can be multiple)
- Define a FINITE set of possible **Actions** (legal moves) for EACH state in the State Space
- Come up with a **Transition Model** which describes what each action does
- Come up with a **Terminal Test** that verifies if the game is over
- Specify the **Utility (Payoff / Objective) Function**: a function that defines the final numerical value to player *p* when the game ends in **terminal state** *s*

MinMax Algorithm: the Idea

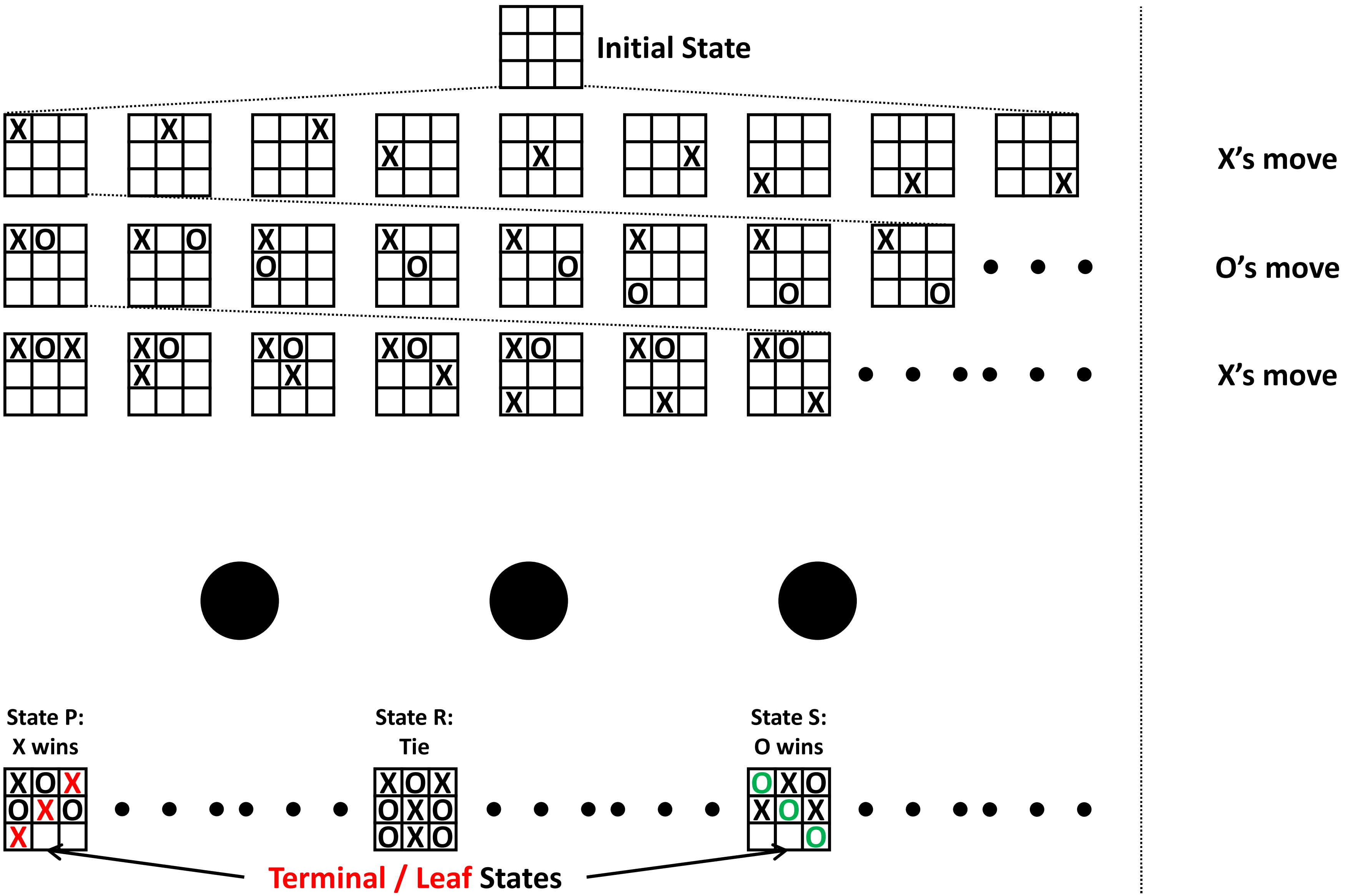
I don't know what move my opponent will choose, but I am going to ASSUME that it is going to be the best / optimal option

Tic Tac Toe: Zero Sum Game (2 Players)

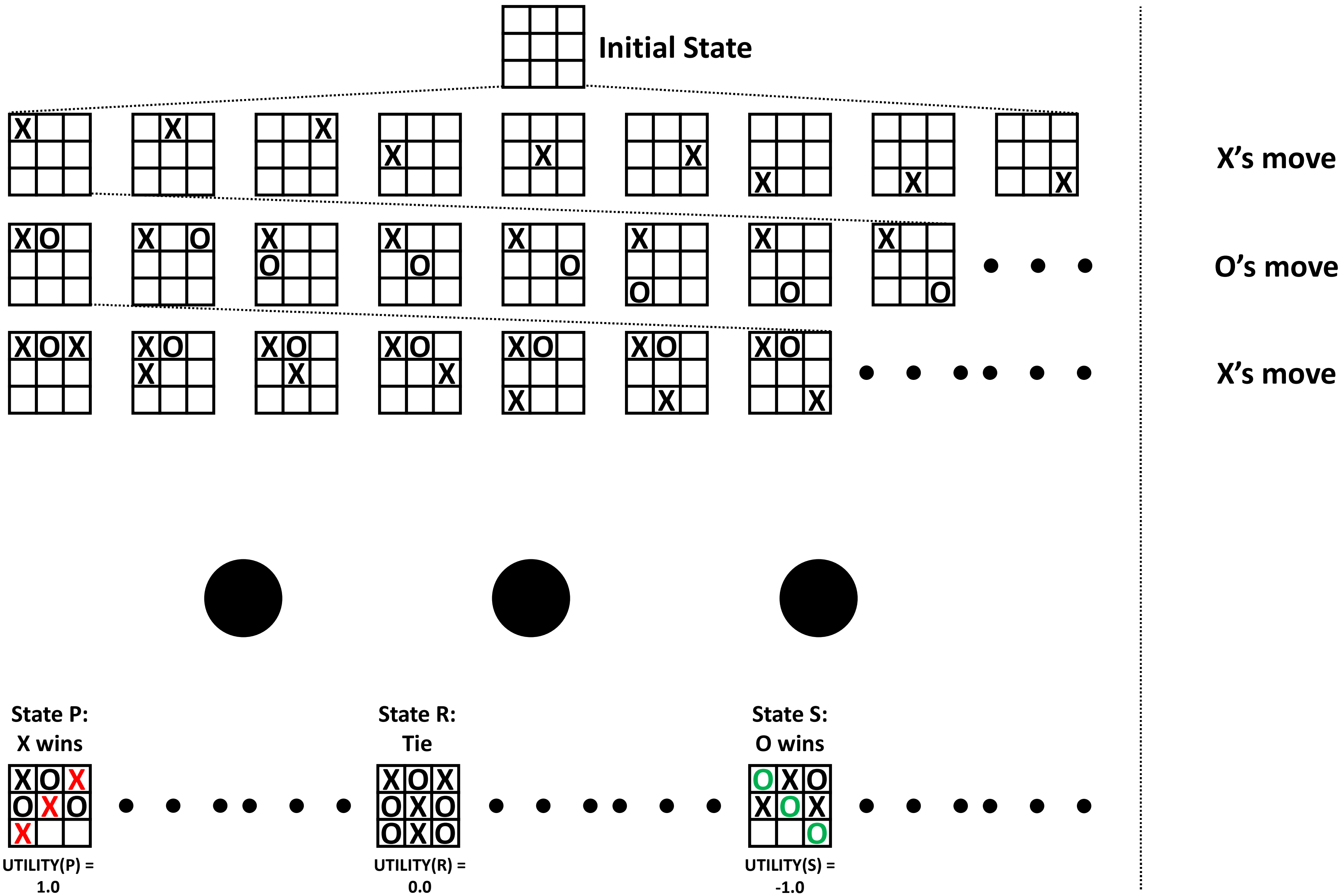


Challenge: I don't know what move my opponent will choose. I can't plan ahead because of that.

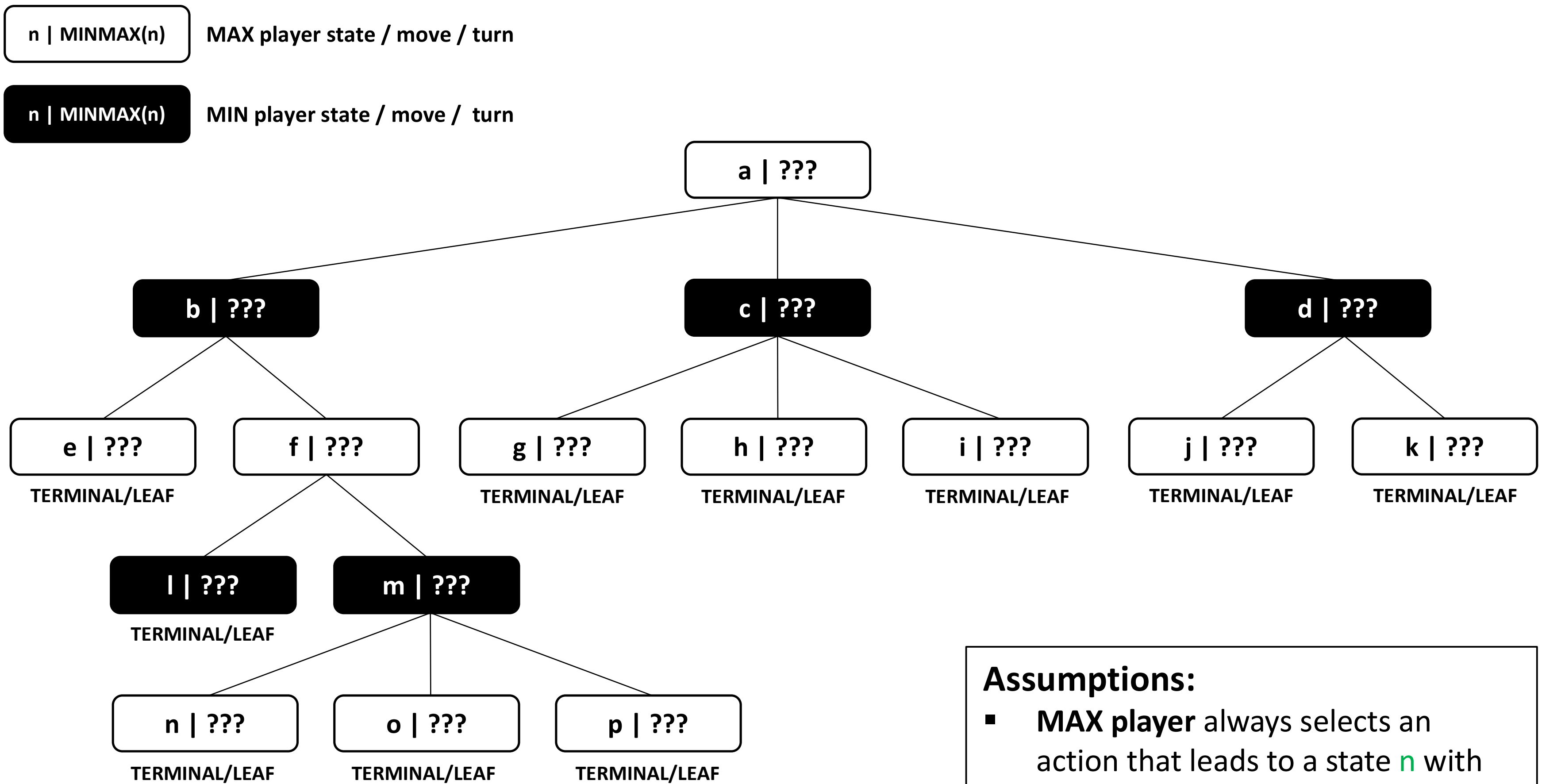
Tic Tac Toe: Zero Sum Game (2 Players)



Tic Tac Toe: Zero Sum Game (2 Players)



Example MinMax Search Tree



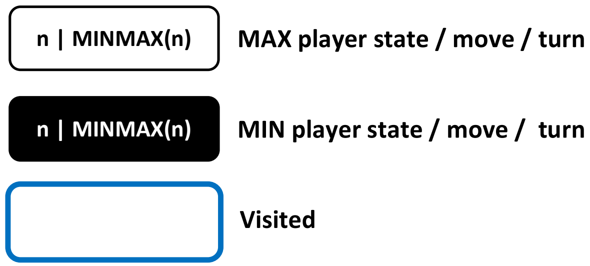
$$MINMAX(n) = MINMAX(State_n)$$

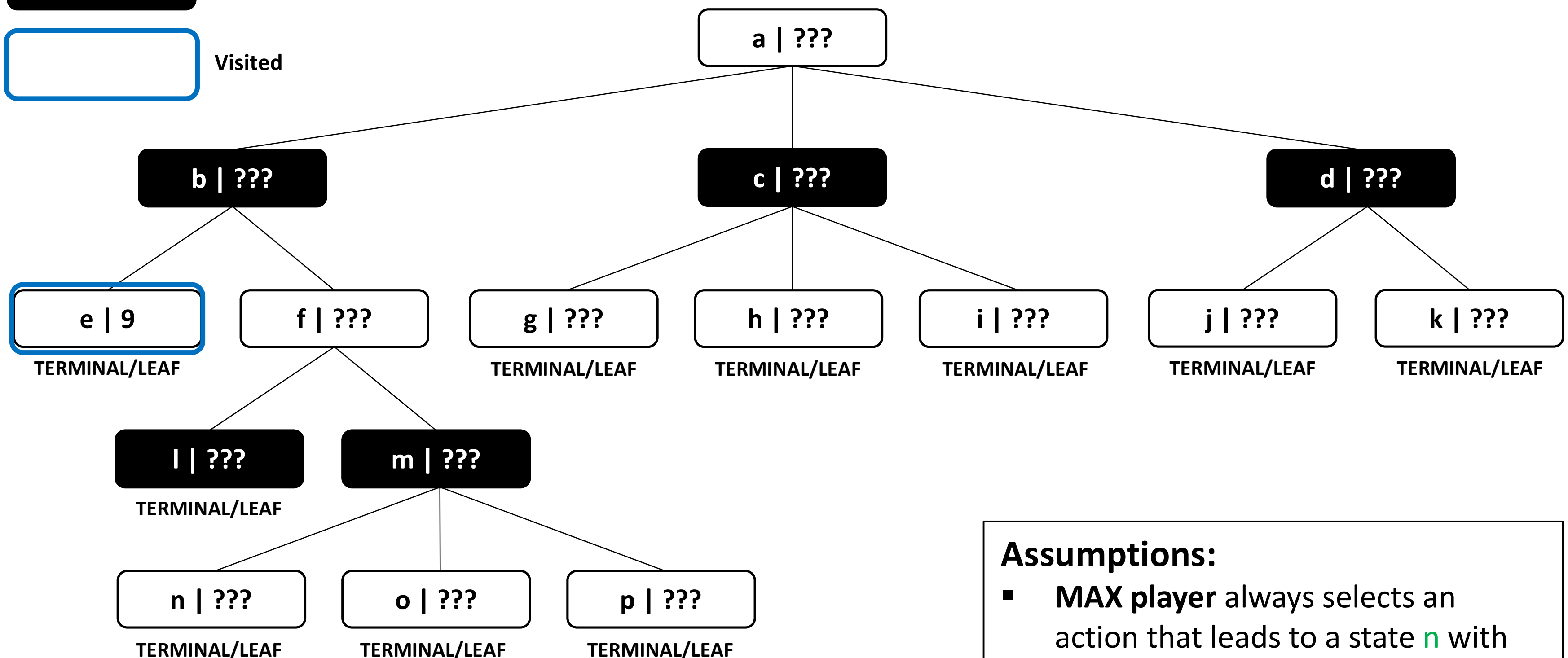
$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with maximum MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with minimum MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree





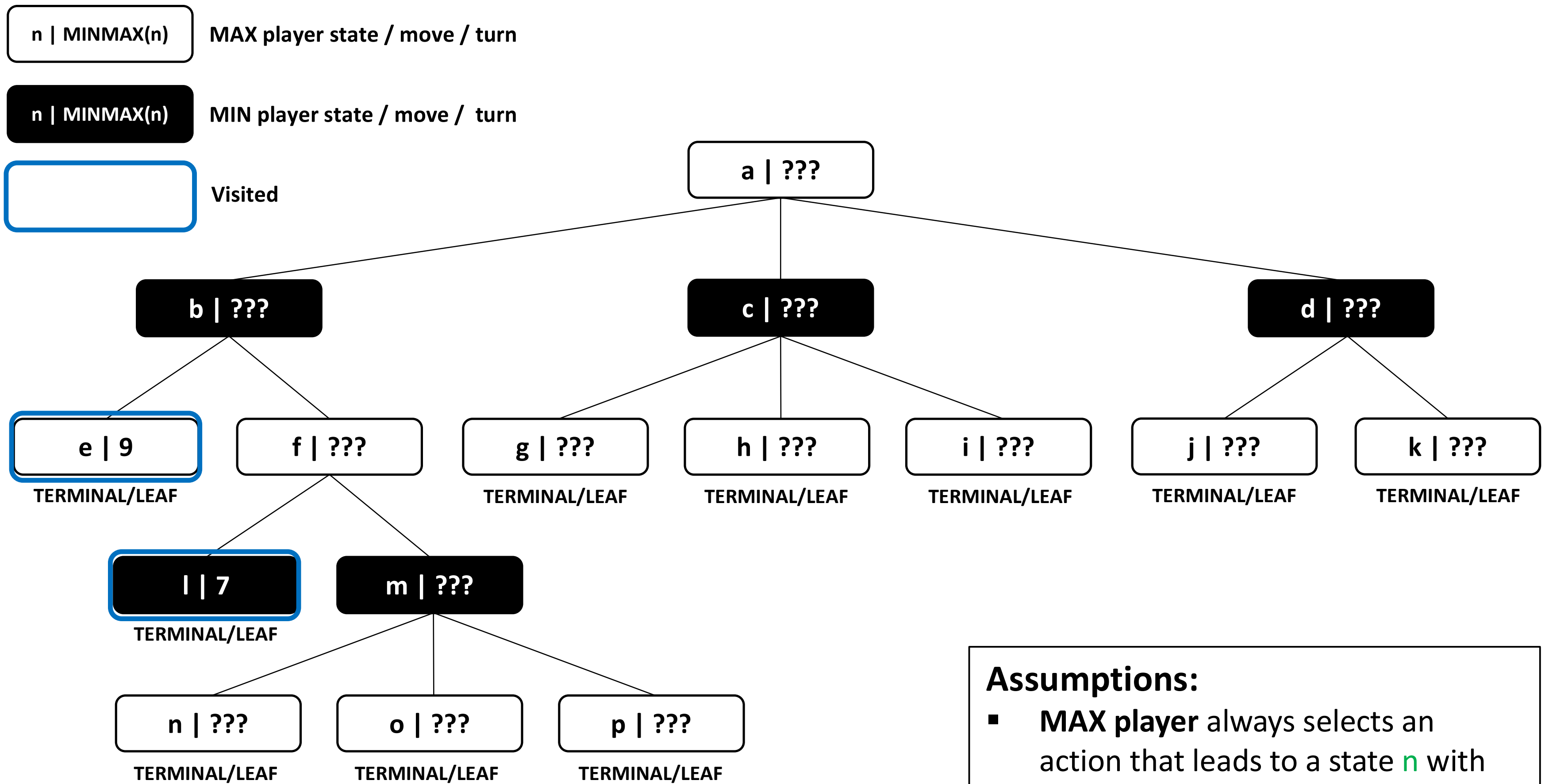
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \; if \; ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \; if \; TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \; if \; TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



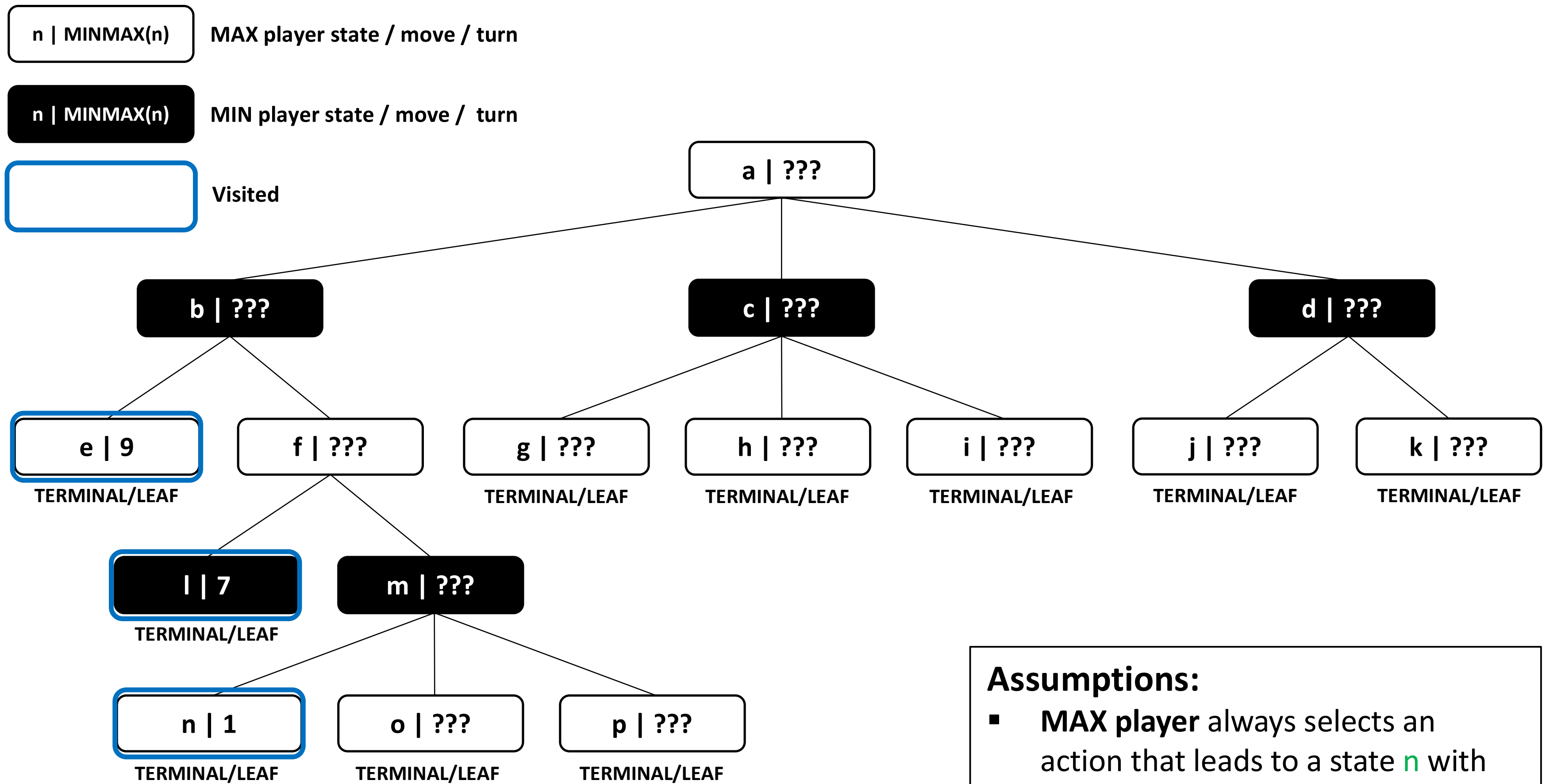
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



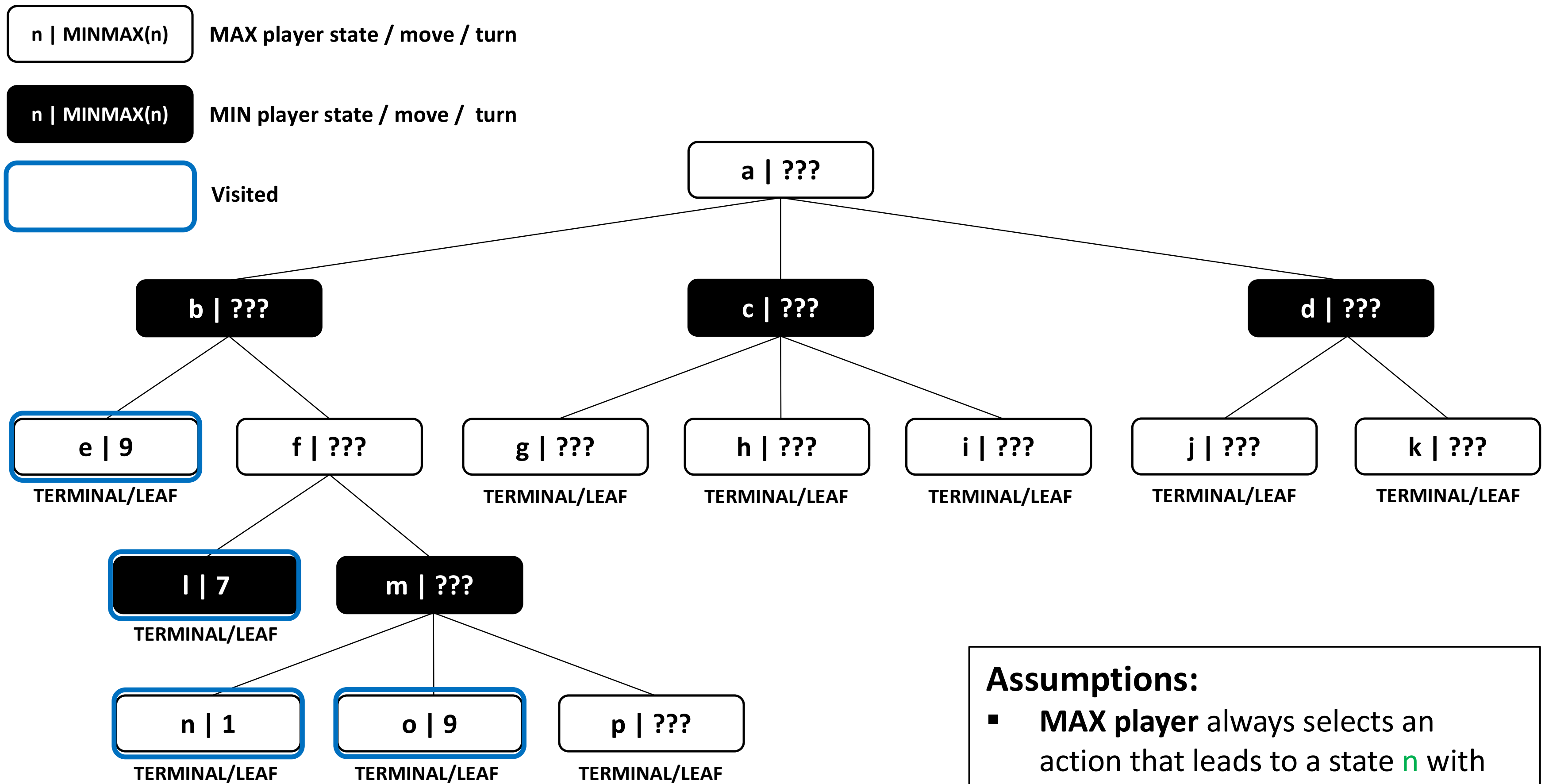
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \ if \ ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \ if \ TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \ if \ TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



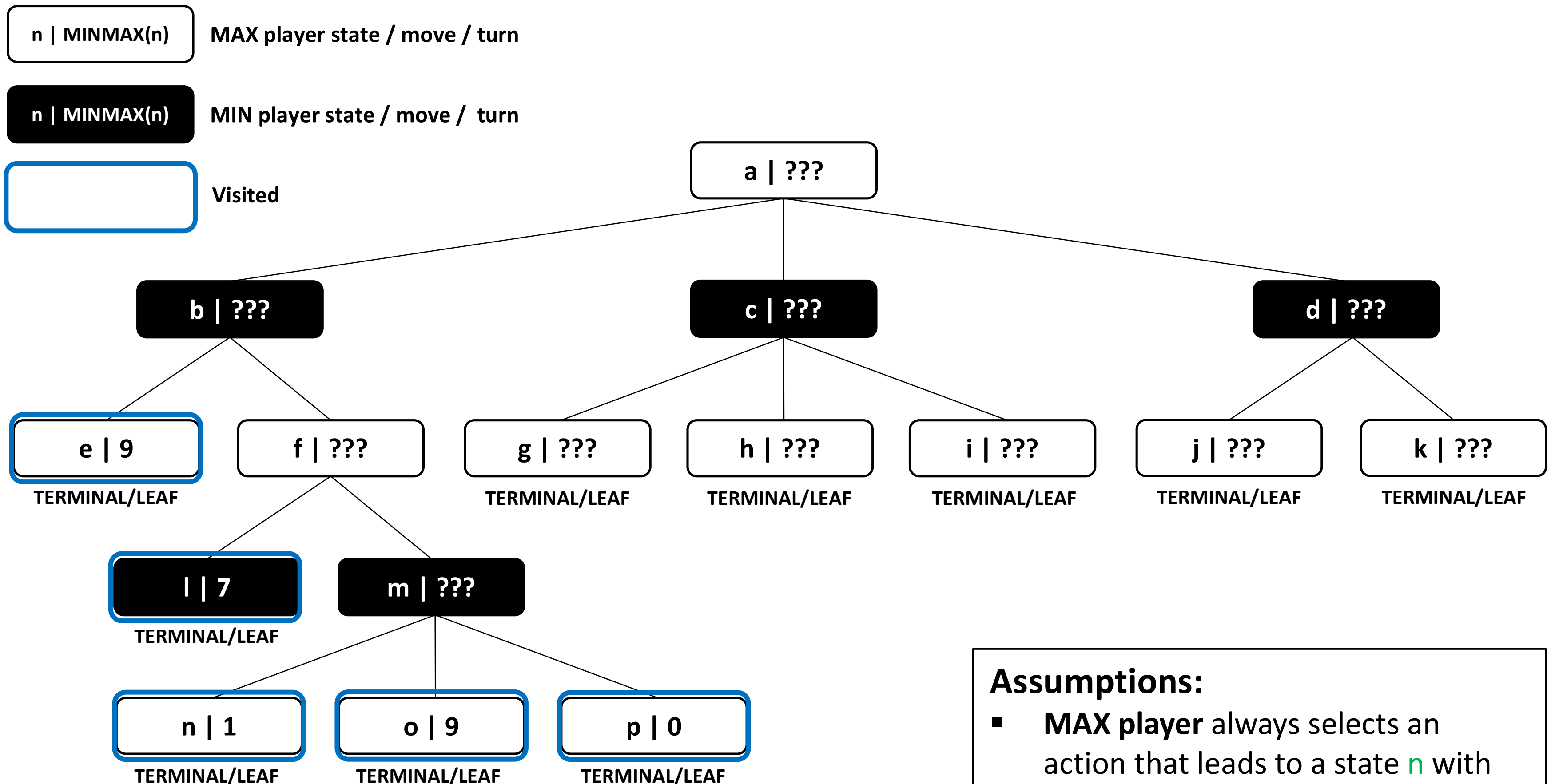
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX),\ if\ ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)),\ if\ TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)),\ if\ TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



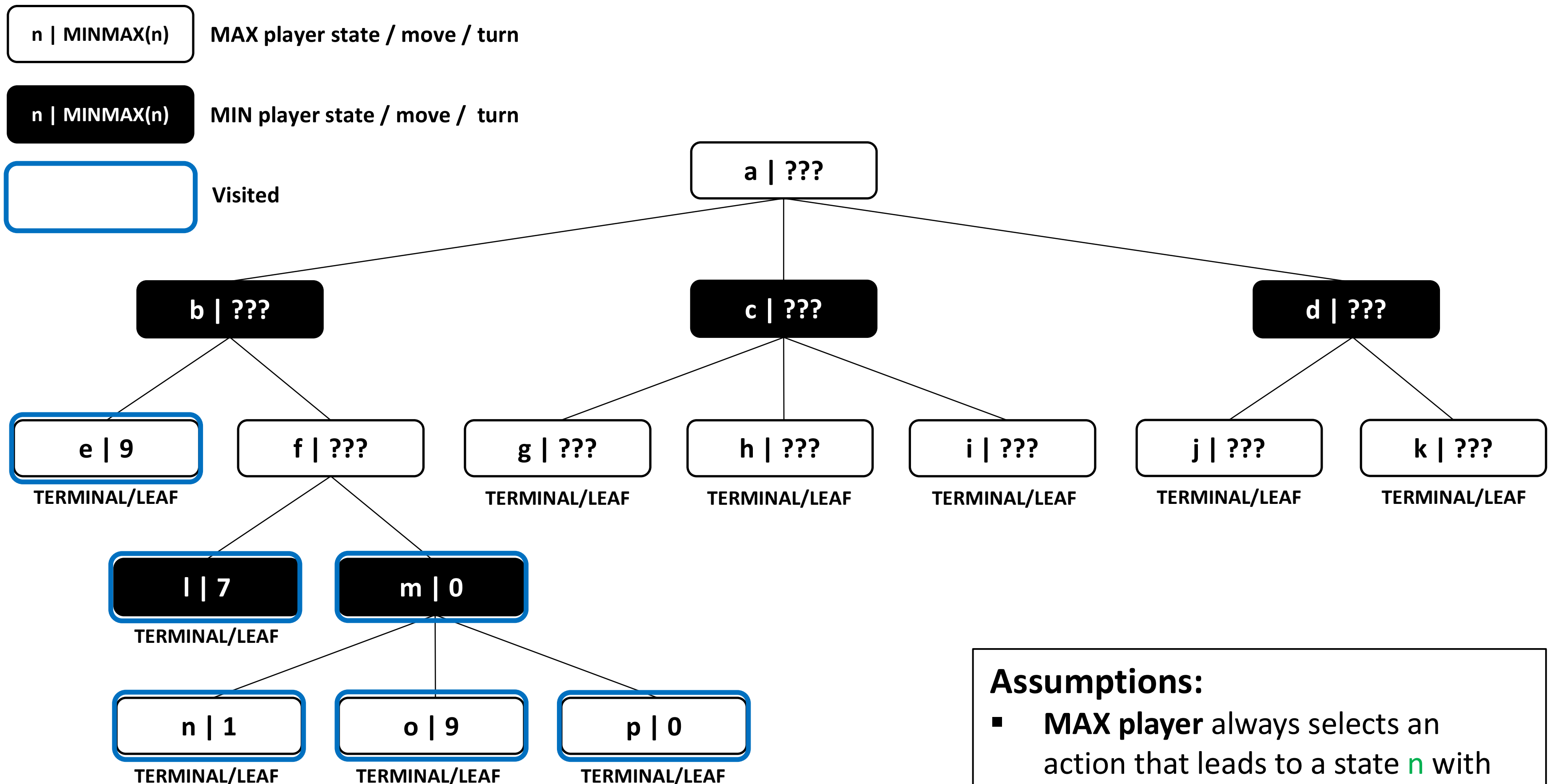
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



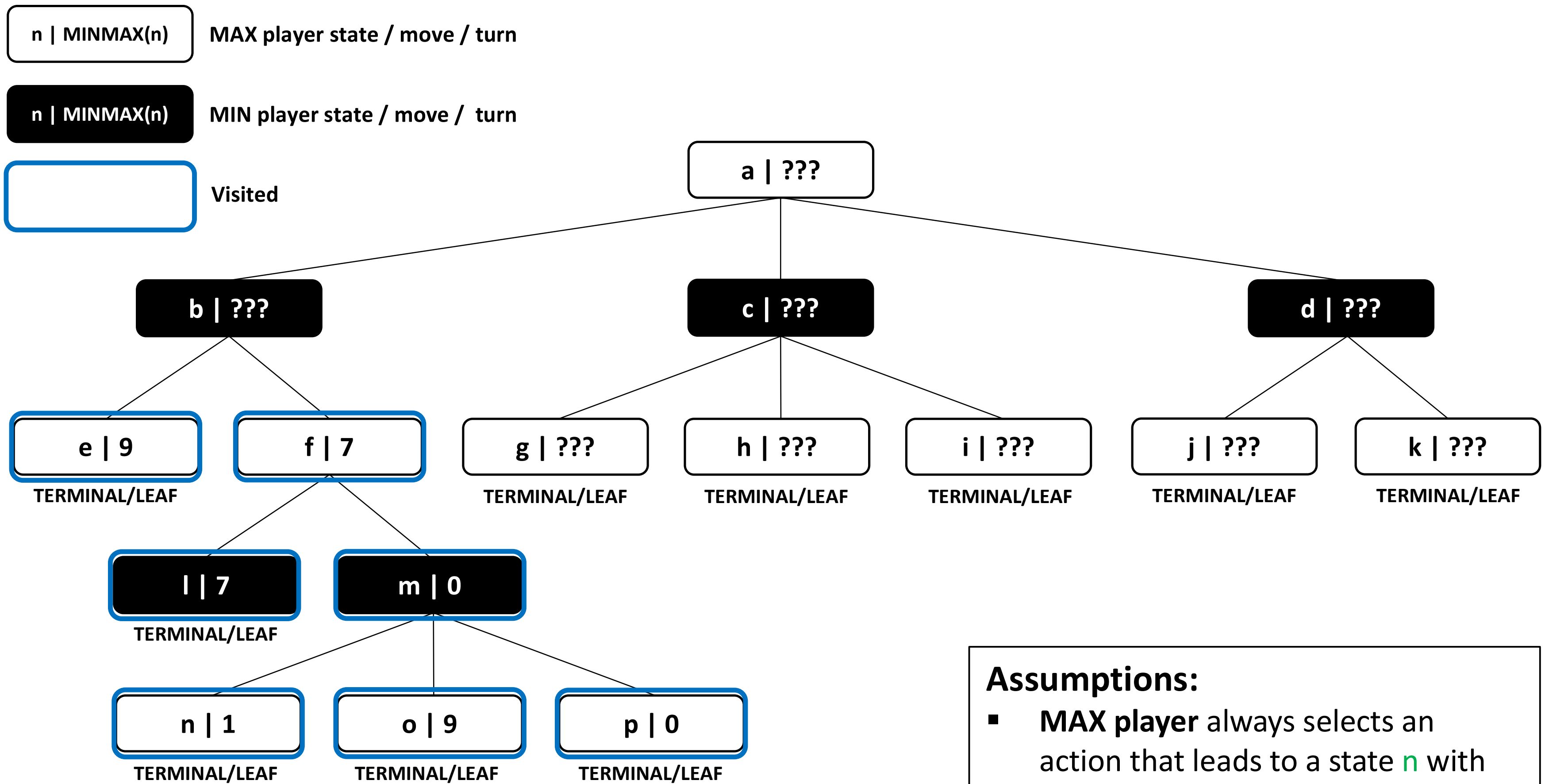
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n, a)), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n, a)), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



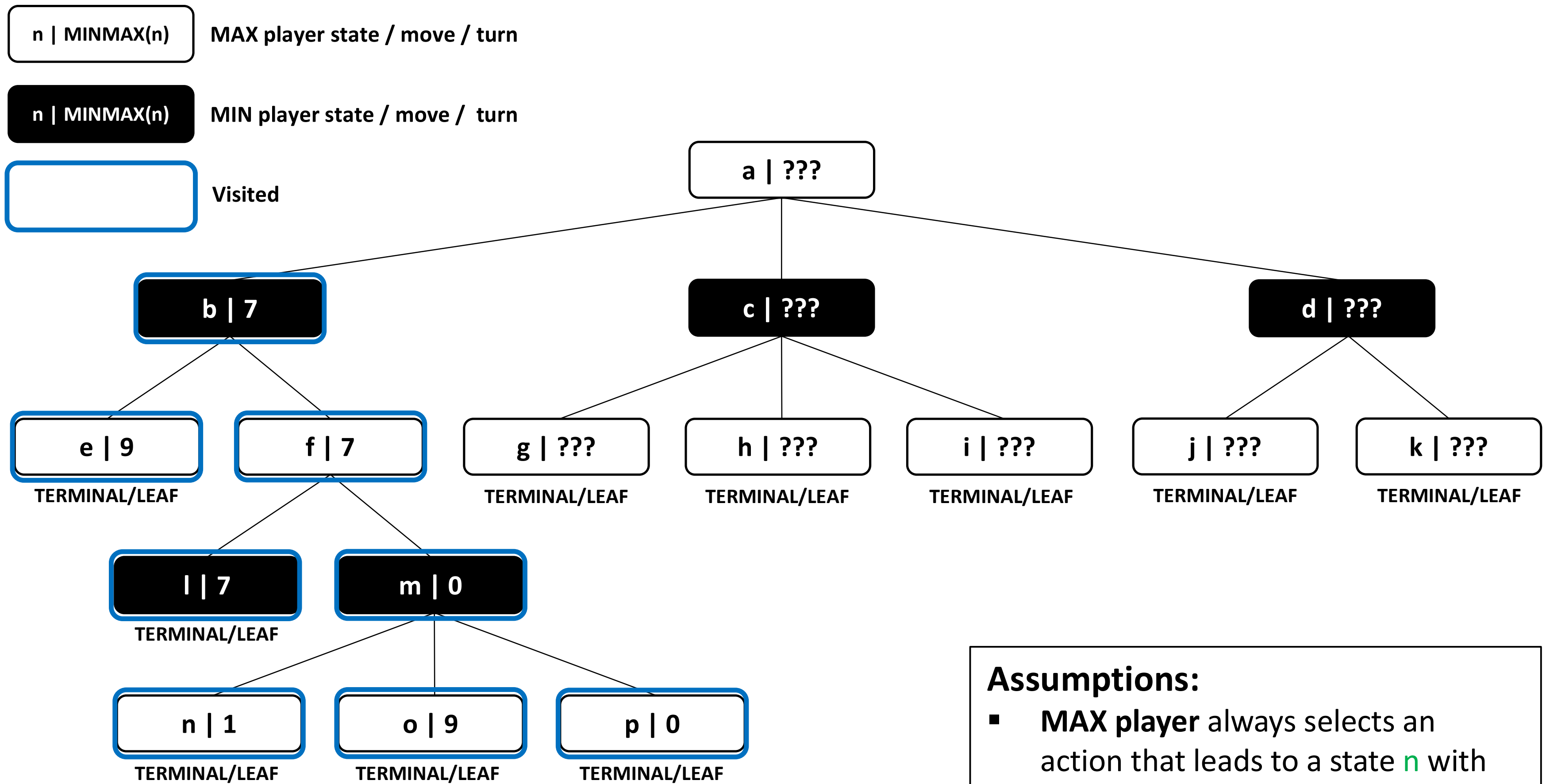
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



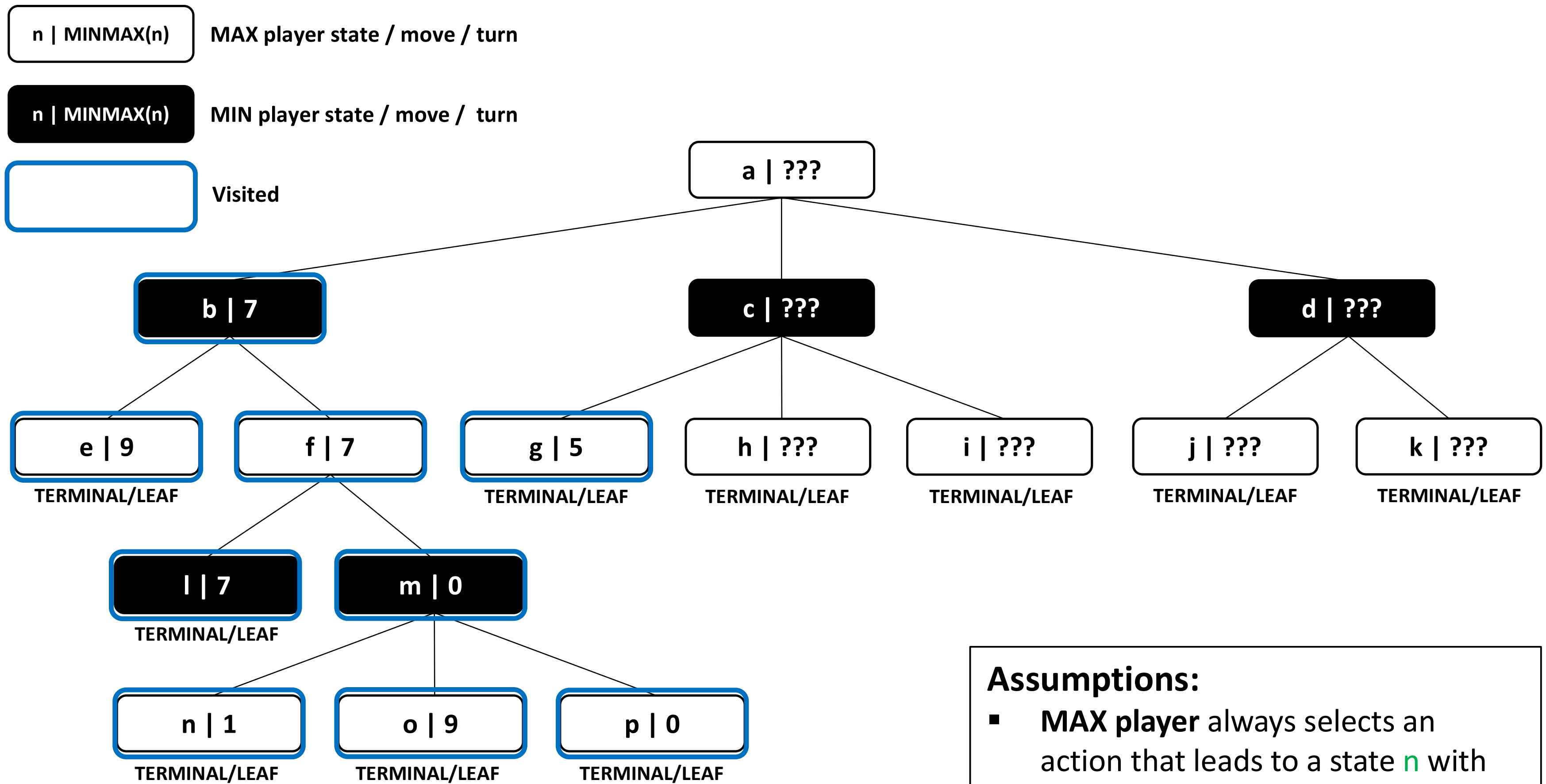
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



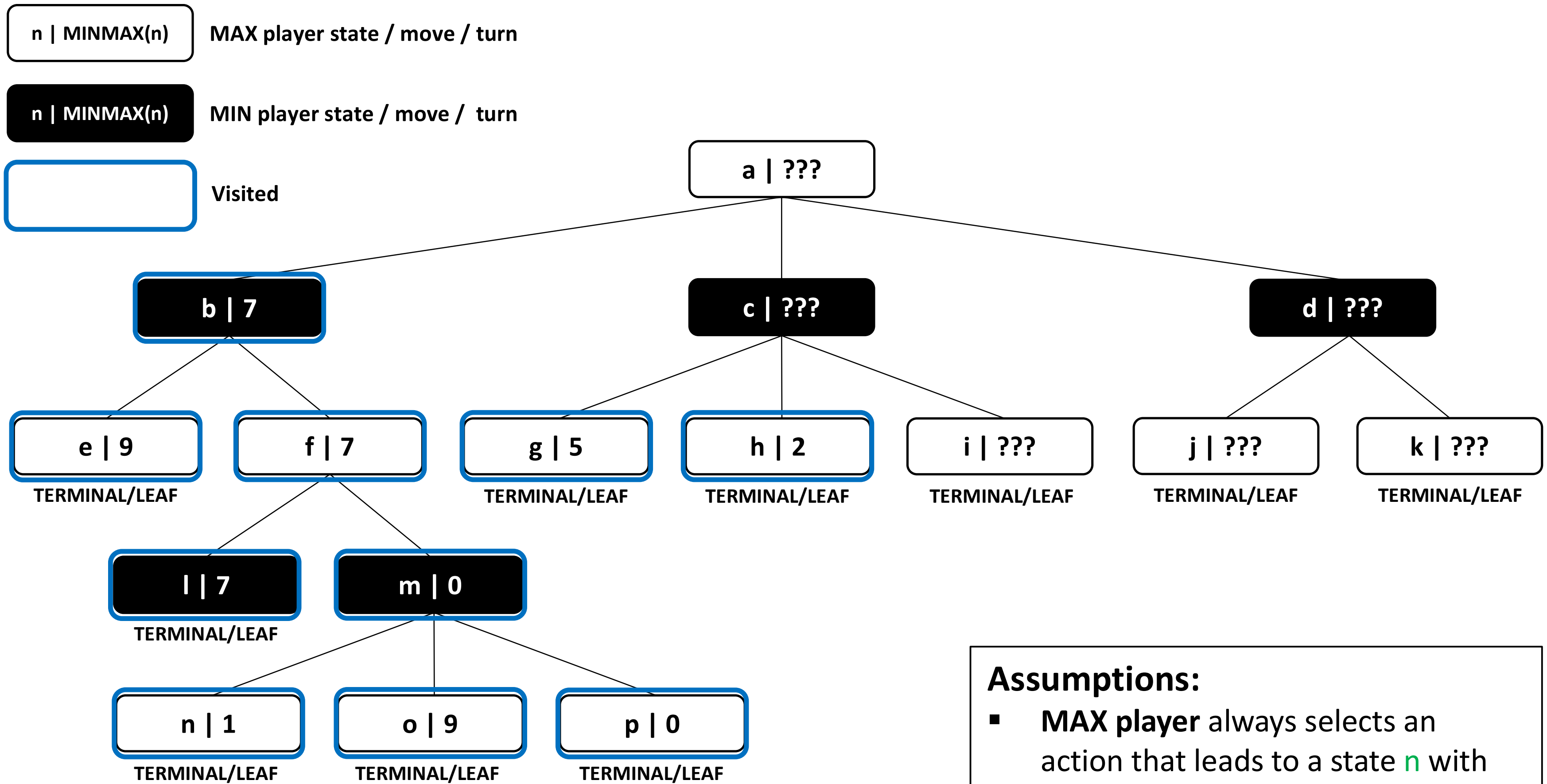
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



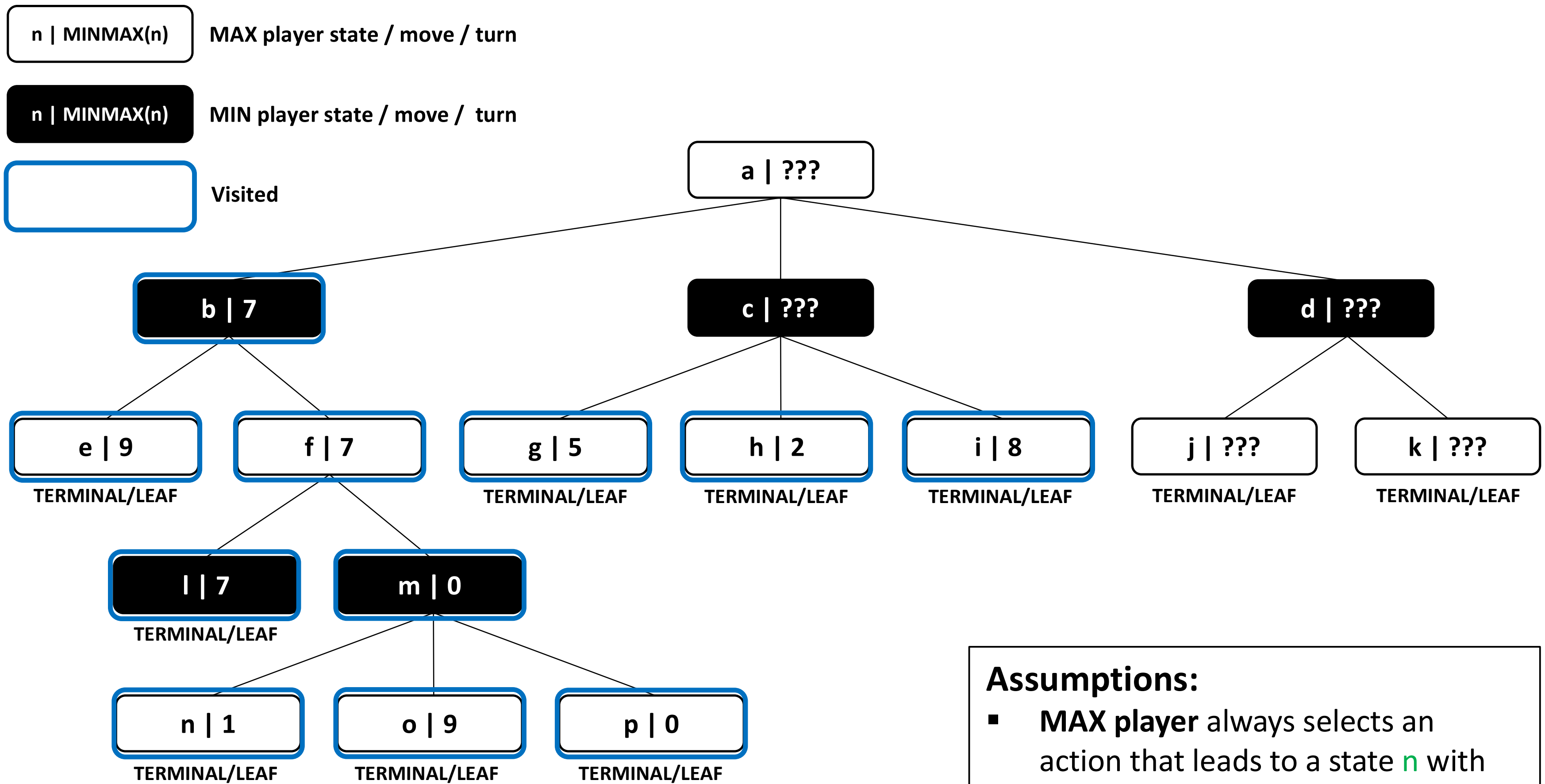
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



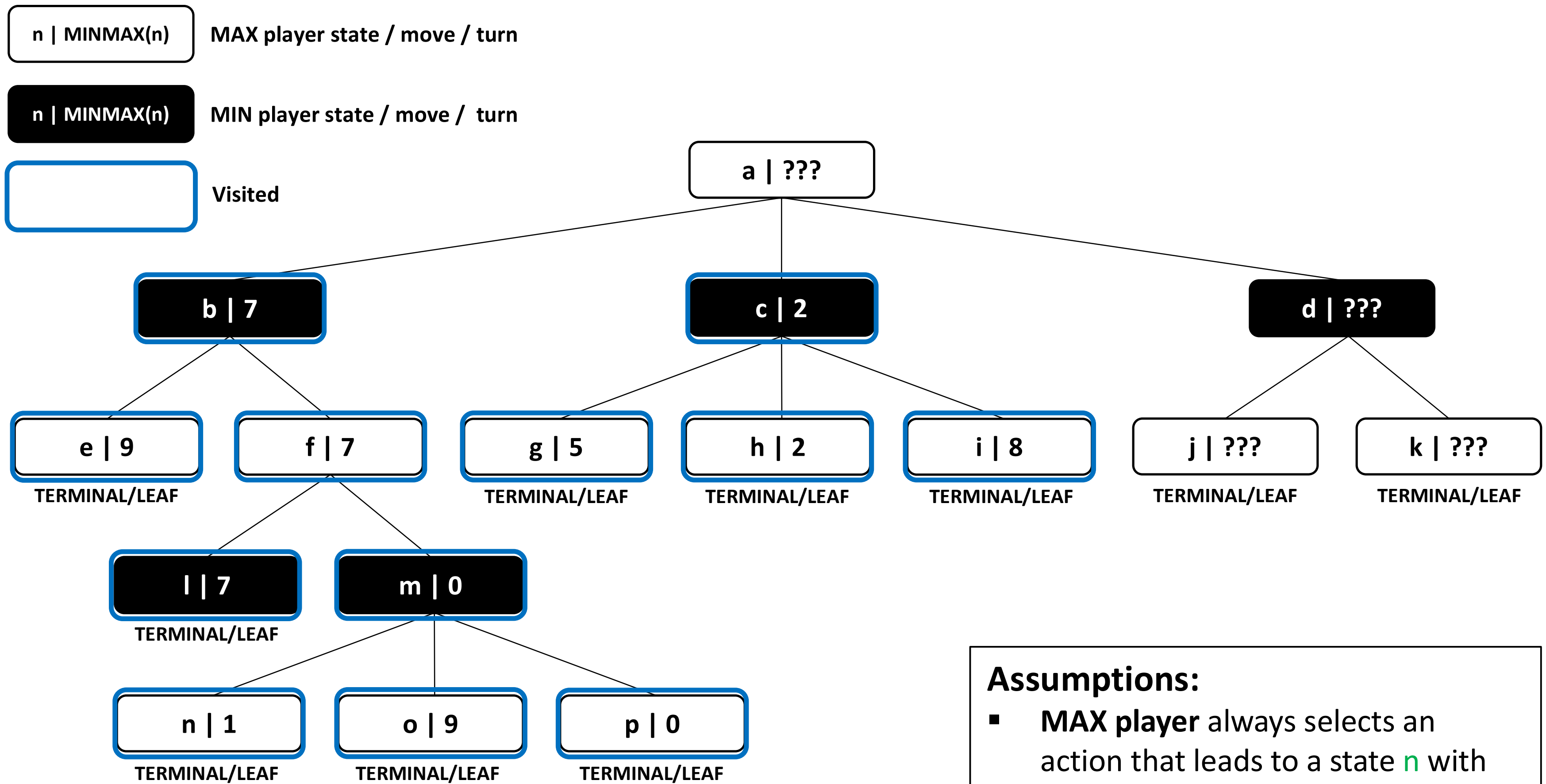
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \; if \; ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \; if \; TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \; if \; TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



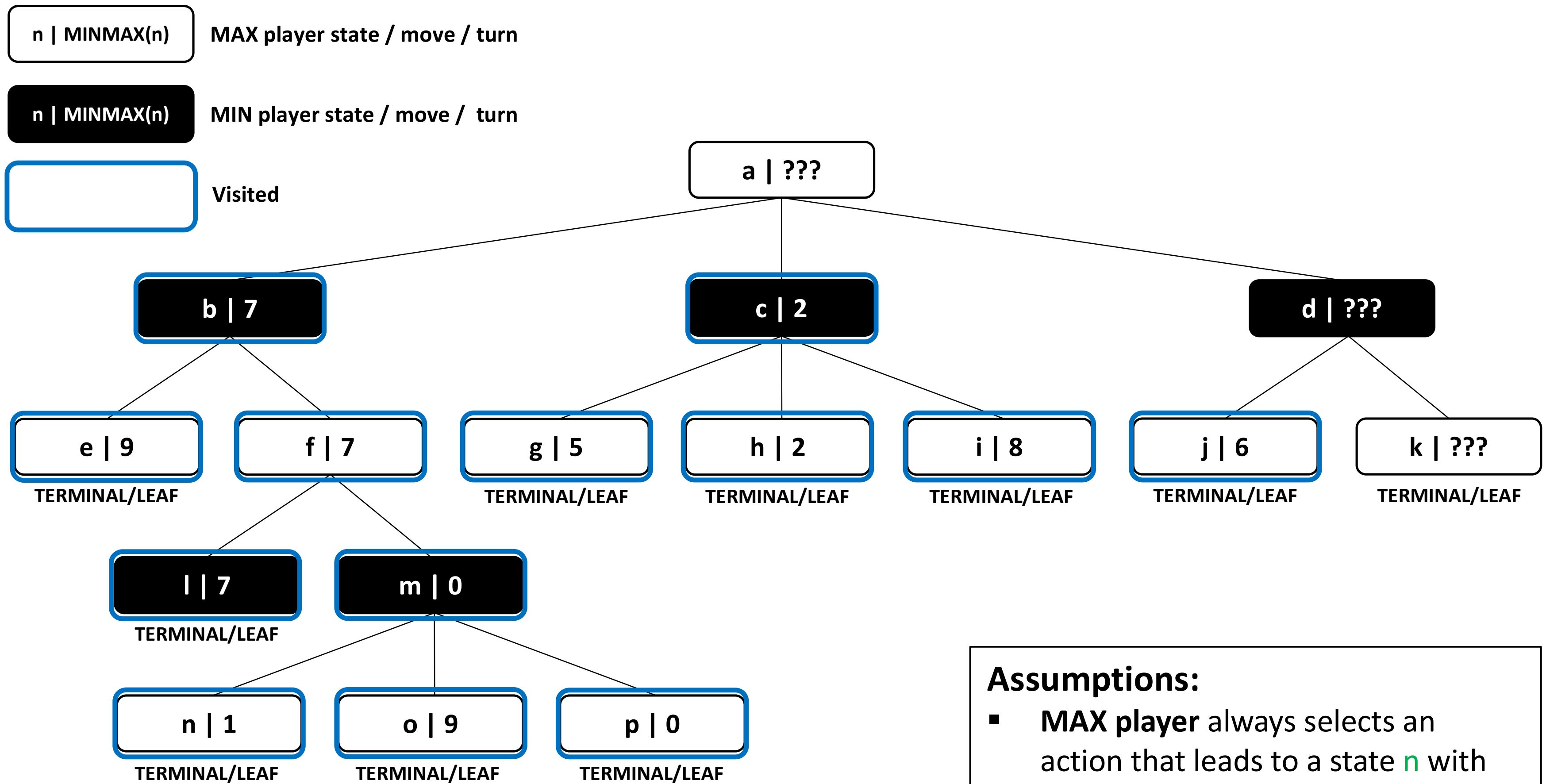
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \; if \; ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \; if \; TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \; if \; TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



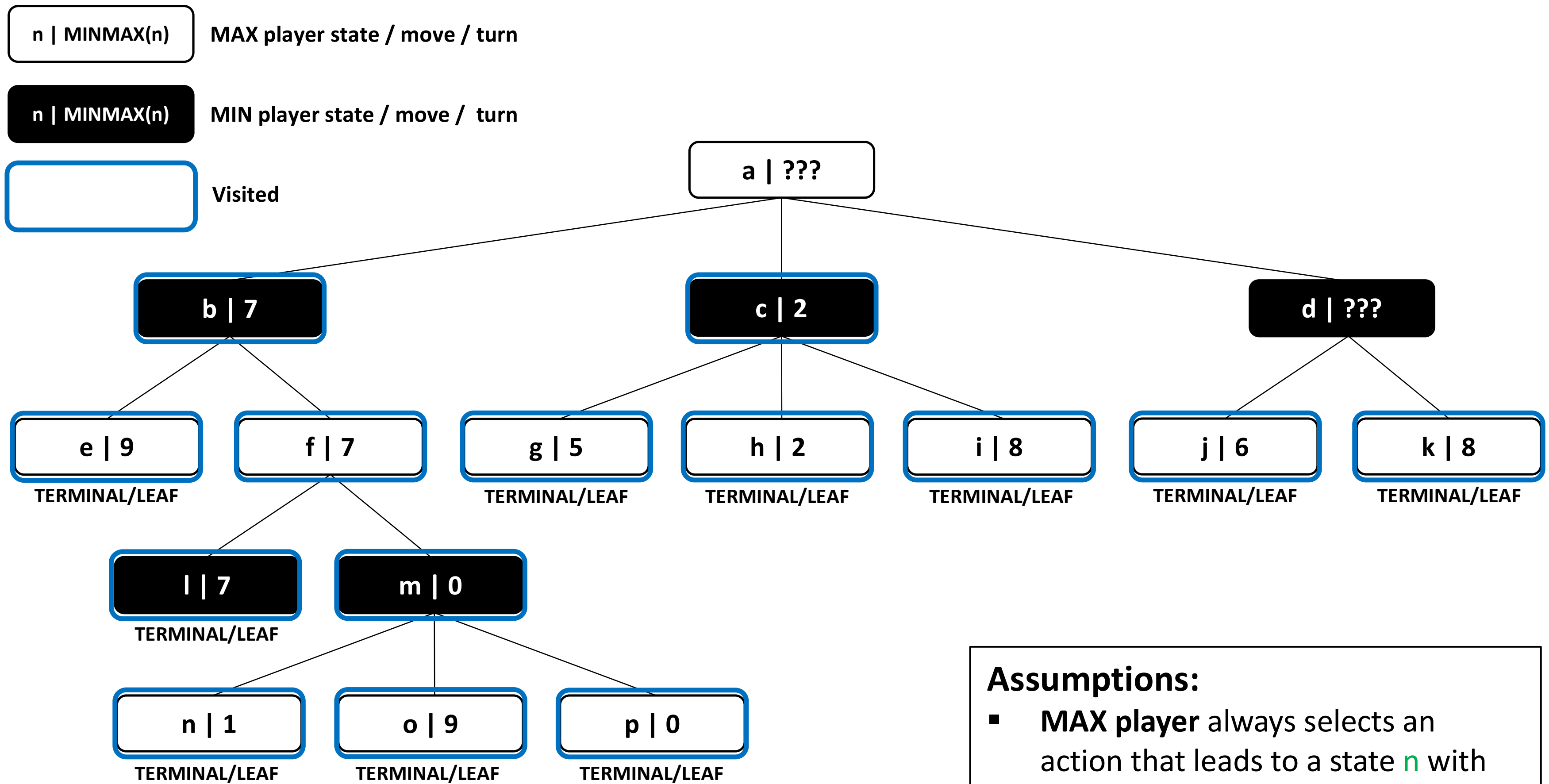
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), if\ ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), if\ TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), if\ TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



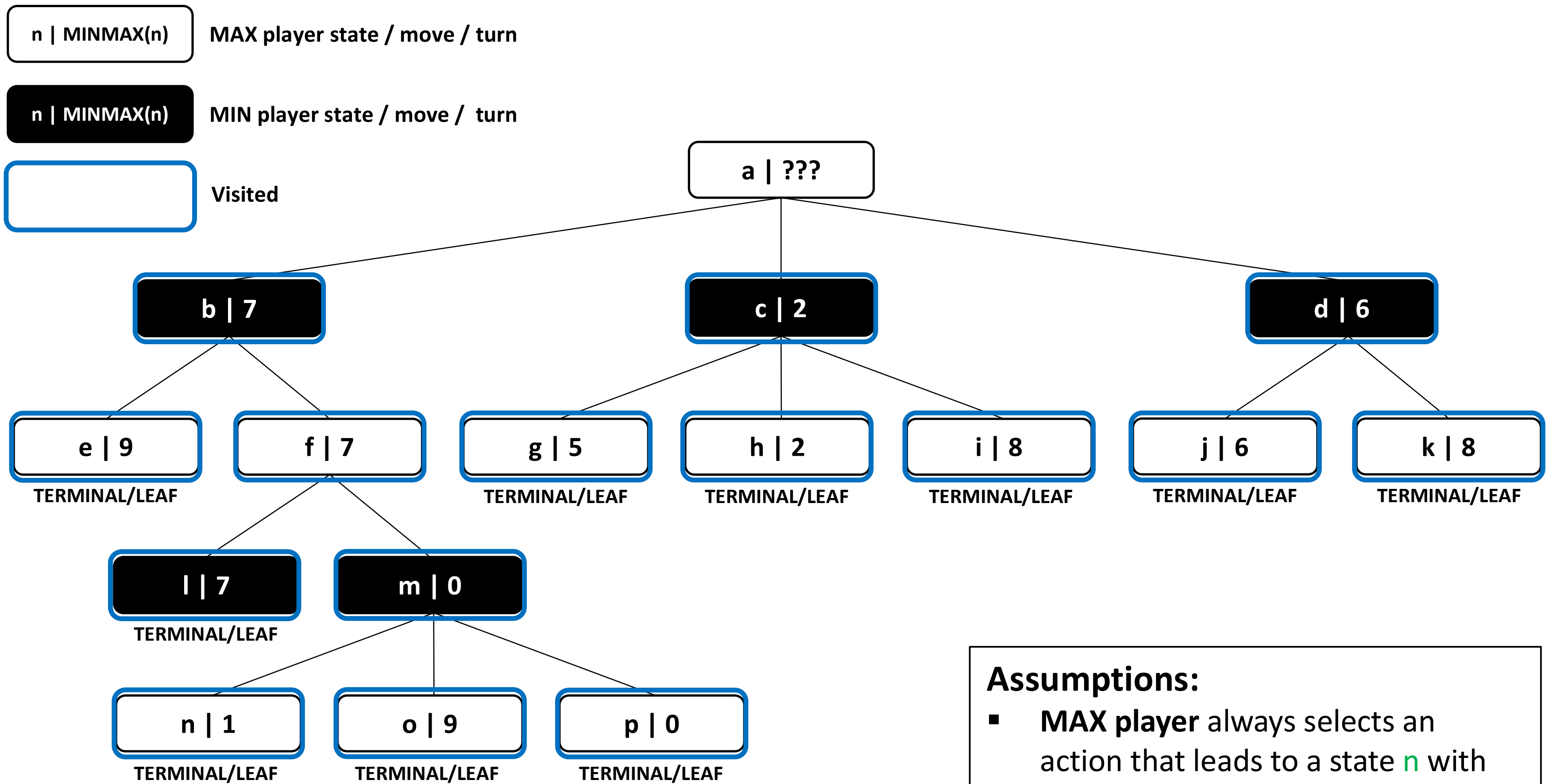
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \; if \; ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a), \; if \; TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a), \; if \; TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



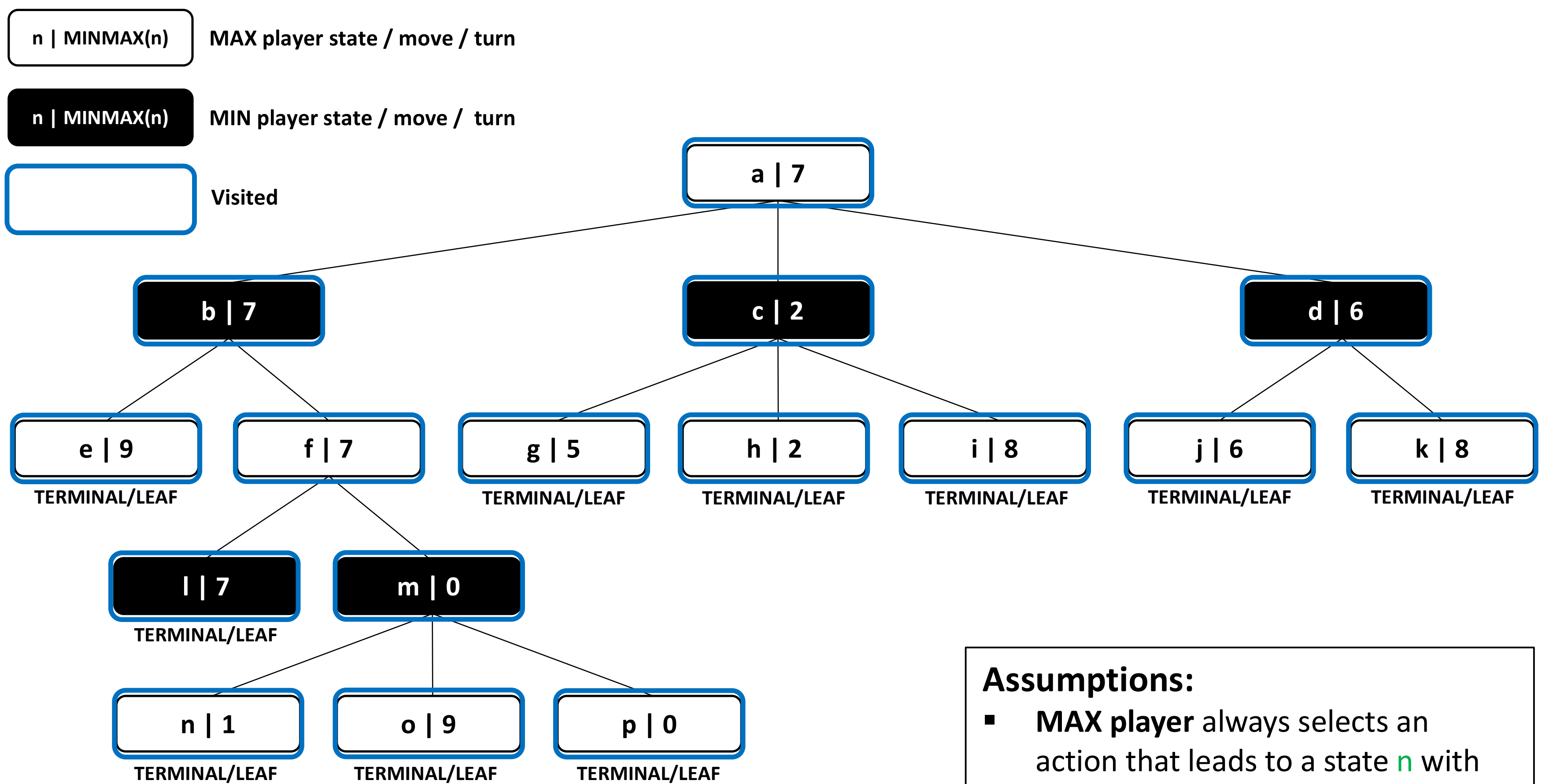
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), if\ ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), if\ TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), if\ TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



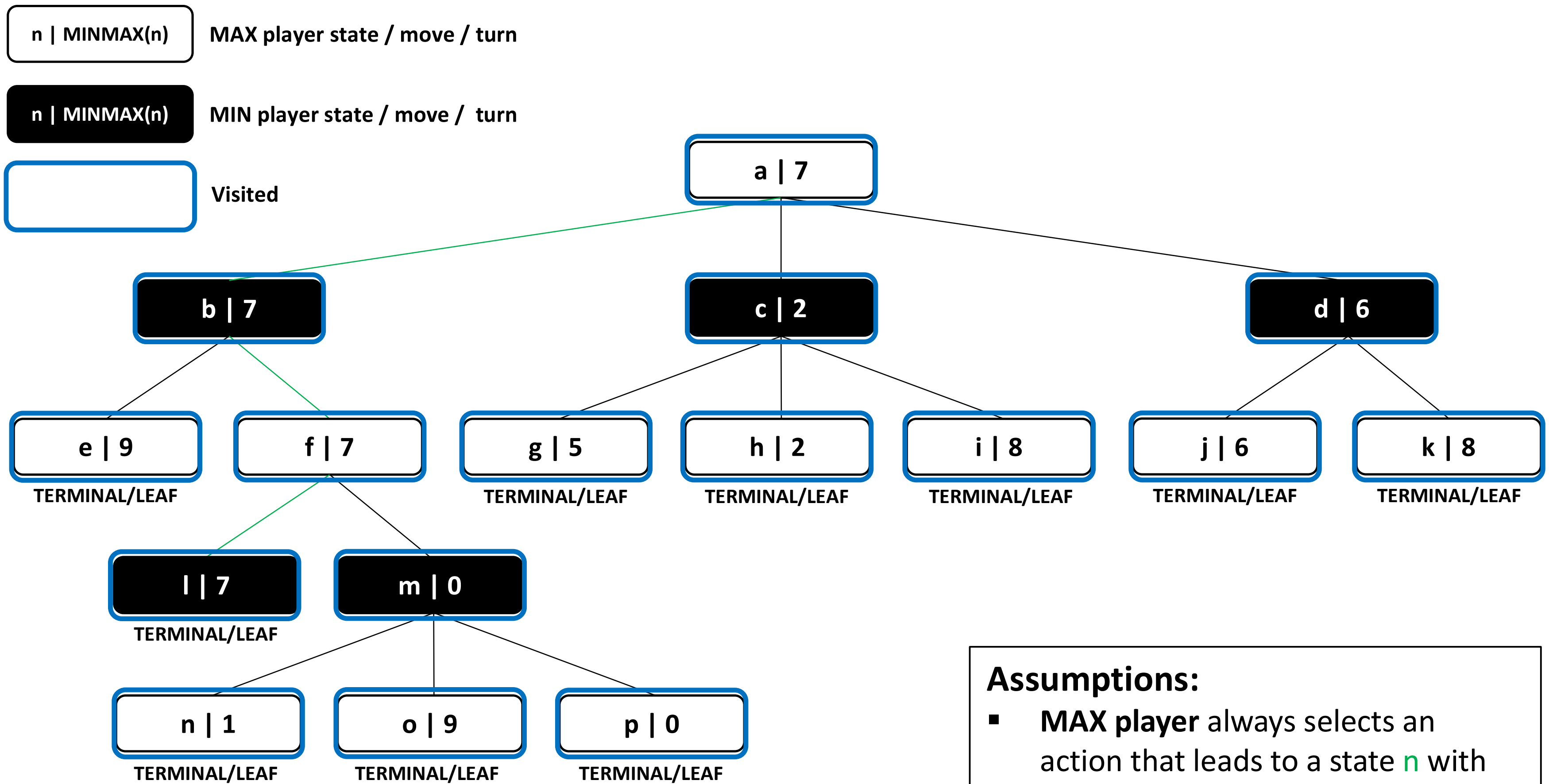
$$MINMAX(n) = MINMAX(State_n)$$

$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \; if \; ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n, a)), \; if \; TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n, a)), \; if \; TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

Example MinMax Search Tree



$$MINMAX(n) = MINMAX(State_n)$$

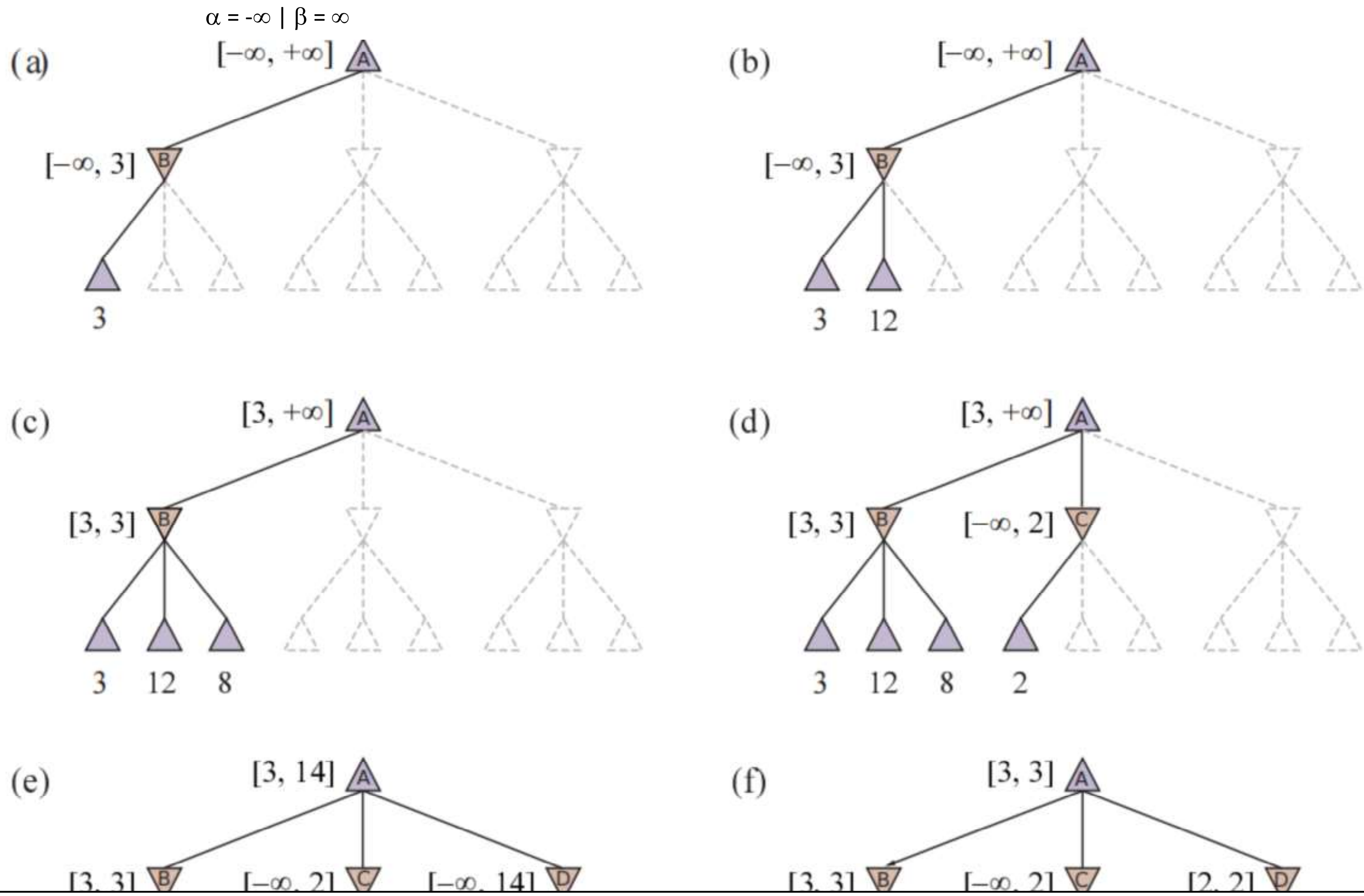
$$MINMAX(n) = \begin{cases} UTILITY(n, MAX), \text{ if } ISTERMINAL(n) \\ max_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MAX \\ min_{a \in ACTIONS(n)} MINMAX(RESULT(n,a)), \text{ if } TOMOVE(s) = MIN \end{cases}$$

Assumptions:

- **MAX player** always selects an action that leads to a state n with **maximum** MINIMAX(n) value
- **MIN player** always selects an action that leads to a state n with **minimum** MINIMAX(n) value
- **BOTH players** always play optimally

MinMax: What is the Challenge?

Example MinMax with α-β Pruning



α: the value of the best (highest-value) choice we have found so far at any choice point along the path for MAX player ("at least")
β: the value of the best (lowest-value) choice we have found so far at any choice point along the path for MIN player ("at most")

Example MinMax with α-β Pruning