loss, or a draw.

- A leaf node corresponds to an end-of-game configuration: a win, a

- The initial configuration of the "game board" is the root node.

- Each node in the game tree corresponds to a possible configuration of

  We can model the game as a multiway tree.

  victory by one player over the other or (2) a draw.

- Further, we assume that the game will always end in either (1) a

  making moves.

Suppose that we have a two-player game in which the players take turns

Game Trees
Game Trees: An Example (cont.)
where \( x \) is a leaf.

\[
p(x) = \begin{cases} 0 & \text{if John loses} \\ 1 & \text{if John wins} \\ -1 & \text{if John ties} \end{cases}
\]

Conceptually, John defines a "pay-off" function \( p \) as follows:

- Concretely, if John wins he gets 1, if he losses he gets -1, and if he ties, he gets 0.

For each leaf node \( x \), John can assign a value of 1 to \( x \) if \( x \) wins, 0 if
has the minimum value (from John's perspective).

The idea is that John would move to a node that value for him, and Mary would do her best by moving to a node that

\[ X \text{ is not a leaf.} \]

if Mary moves next

\[ \{ x : V(c) \mid c \text{ is a child node of } x \} \text{ within} \]

if John moves next

\[ \{ x : V(c) \mid c \text{ is a child node of } x \} \text{ max} \]

\[ = (X) \land \]

John assigns a value to a non-leaf node \( X \) as follows:

\[ \text{Min-Max Algorithm (cont.).} \]
At best, you can examine the game tree up to a certain depth.

In general, it is not possible to examine all least nodes of a non-trivial game.

Min-Max Algorithm (cont.)
\[
\text{max}(a, q) = \min(-a, -q)
\]

Formula: can reformulate the min-max strategy based on the simple mathematical
Instead of flipping between max and min as described above, we

Modified Min-Max Algorithm
- $E(n)$ if Mary is to make the next move.

- $e(n) = E(n)$ if $n$ is a node from which John is

  \[ \text{let } \]

  a game node $n$. 

- $E(n)$ be the payoff function that John uses to evaluate

  \[ \text{let } \]

Modified Min-Max Algorithm (cont.)
of the (immediate) children of $x$. If $x$ is a leaf of the game subtree and $c$ ranges over all

\[ \max(\text{Min-Max}(c)) \]

If $x$ is not a leaf

\[ \text{Min-Max}(x) = e(x) \]

Let

Modifiled Min-Max Algorithm (cont.)
The mini-max formula: given x, a game tree node (i.e., a game board configuration), and d, the number of lookahead moves from x, compute the value of x based on number of children at depth d. If any child has value v, then the best move for the maximizing player is the child with the highest value v, which is the maximum over all possible moves. If any child has value v, then the best move for the minimizing player is the child with the lowest value v, which is the minimum over all possible moves. Therefore, the best move for a player is the one that maximizes the minimum value of the opponent.
This is called "alpha pruning".

For as a value for B and skip the rest of the children of B.

If the value v of a child of B is less or equal to alpha, then we can use

a child node of A.

Let alpha be a lower bound for the value of a max node A, and let B be

the evaluation of some of the node's children.

(\text{\textit{prune}}) the \textit{min-max} value of a game tree node, we can skip

\textbf{Alpha-Beta Pruning}
In the figure below, alpha = 20, and we can prune the rest of the children of C2, once the value of D is (recursively) computed.
This is called "beta pruning".

Use \( v \) as a value for \( C \) and skip the rest of the children of \( C \).

- If the value \( v \) of a child of \( C \) is greater or equal to \( \beta \), then we can stop.

  a child node of \( B \).

Let \( \beta \) be an upper bound for the value of a min node \( B \), and let \( C \) be a child node of \( B \).