# CS217: Artificial Intelligence and Machine Learning (associated lab: CS240)

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

*Week7 of 17feb25, Predicate Calculus, Search in Chess*

# Main points covered: week6 of 10feb25

# Important pointes associated with FFNN BP

Local Minima

Momentum Factor

Symmetry Breaking

**Hilbert's formalization** of propositional calculus

1. Elements are *propositions* : Capital letters

2. Operator is only one : → (called implies)

3. Special symbol $F$ (called 'false')

4. Two other symbols : *'('* and *')'*

5. Well formed formula is constructed according to the grammar

$$WFF \rightarrow P|F|WFF \rightarrow WFF$$

6. Inference rule : only one

Given $A \rightarrow B$ and

$A$

write $B$

known as MODUS PONENS

7. Axioms : Starting structures

A1: $(A \to (B \to A))$

A2: $((A \to (B \to C)) \to ((A \to B) \to (A \to C)))$

A3 $(((A \to F) \to F) \to A)$

This formal system defines the propositional calculus

# A very useful theorem (Actually a meta theorem, called deduction theorem)

Statement

If

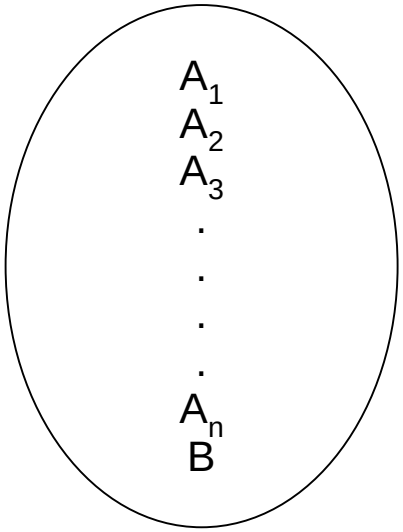
$A_1, A_2, A_3 \ldots\ldots\ldots\ldots A_n \vdash B$

then

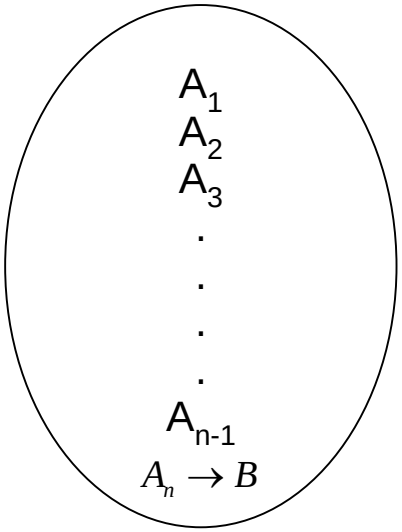
$A_1, A_2, A_3, \ldots\ldots\ldots\ldots A_{n-1} \vdash A_n \rightarrow B$

$\vdash$ is read as 'derives'

Given

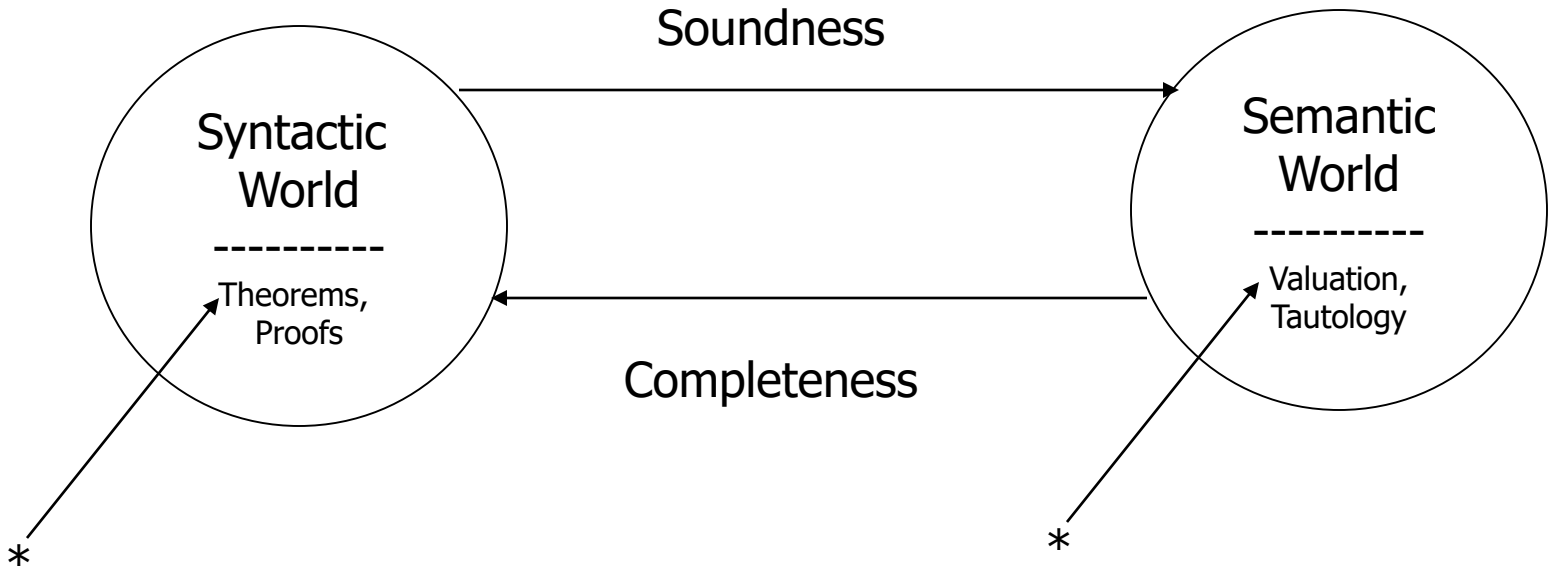



Picture 1




Picture 2

# Soundness, Completeness & Consistency



# Consistency

The System should not be able to prove both P and ~P, *i.e.*, should not be able to derive

$\boxed{F}$

# End of main points

# Predicate calculus

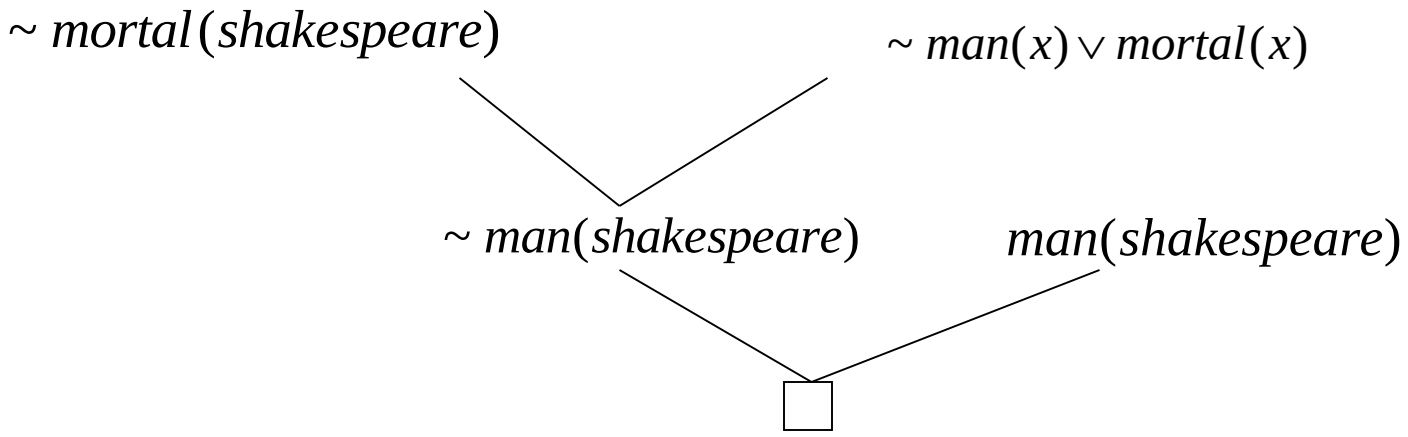
# Insight into resolution

# Resolution - Refutation

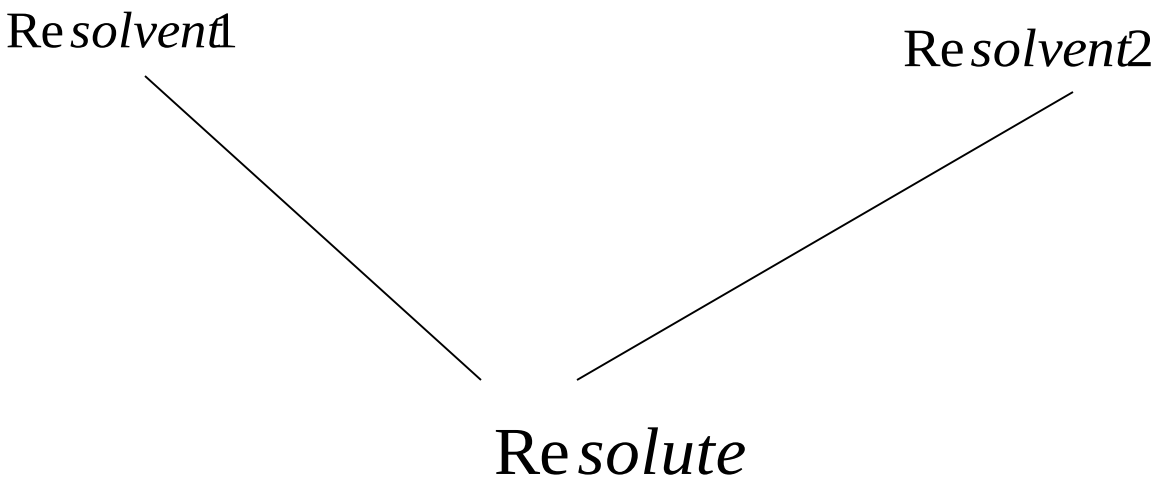
- $man(x) \rightarrow mortal(x)$
  - *Convert to clausal form*
  - *~man(shakespeare) \/ mortal(x)*
- Clauses in the knowledge base
  - *~man(shakespeare) \/ mortal(x)*
  - *man(shakespeare)*
  - *mortal(shakespeare)*

# Resolution – Refutation contd

- *Negate the goal*
  - *~man(shakespeare)*
- Get a pair of resolvents



# Resolution Tree



# Search in resolution

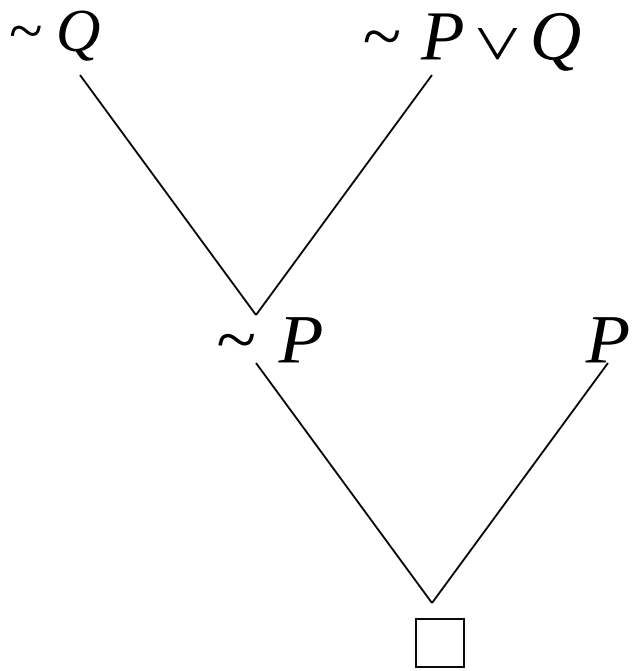
- Heuristics for Resolution Search
  - Goal Supported Strategy
    - Always start with the negated goal
  - Set of support strategy
    - Always one of the resolvents is the most recently produced resolute

# Inferencing in Predicate Calculus

- Forward chaining
  - Given P, $P \rightarrow Q$, to infer Q
  - P, match *L.H.S* of
  - Assert Q from *R.H.S*
- Backward chaining
  - Q, Match *R.H.S* of $P \rightarrow Q$
  - assert P
  - Check if P exists
- Resolution – Refutation
  - Negate goal
  - Convert all pieces of knowledge into clausal form (disjunction of literals)
  - See if contradiction indicated by null clause $\square$ can be derived

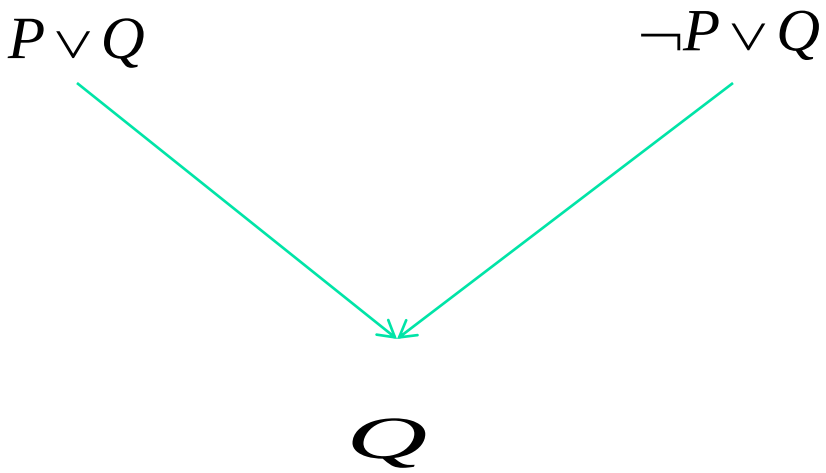
1. $P$
2. $P \rightarrow Q$ converted to $\sim P \vee Q$
3. $\sim Q$

Draw the resolution tree (actually an inverted tree). Every node is a clausal form and branches are intermediate inference steps.



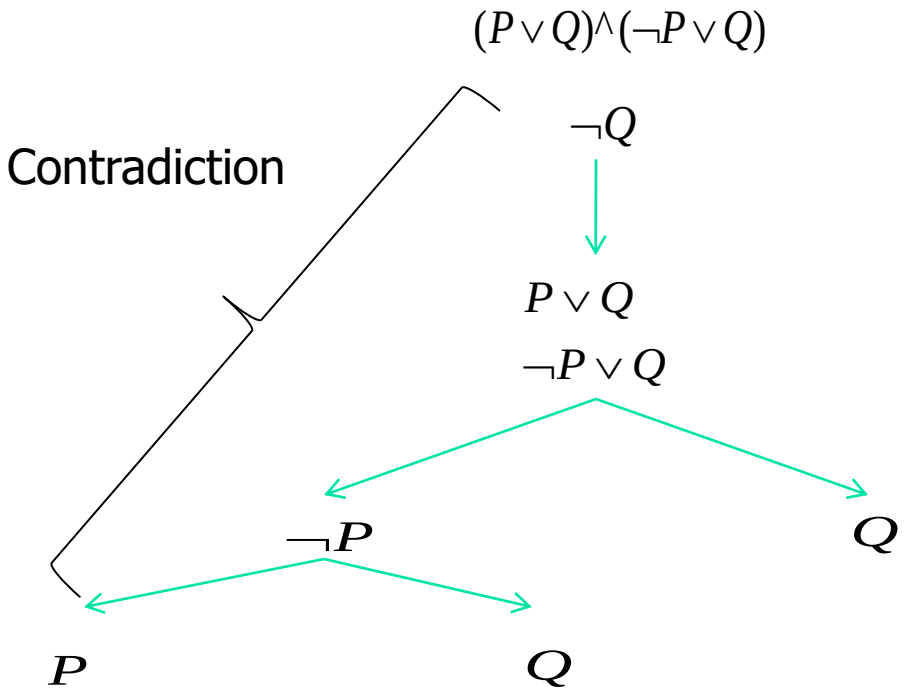
# Theoretical basis of Resolution

- Resolution is proof by contradiction
- ***resolvent1 .AND. resolvent2 => resolute*** is a tautology



# Tautologiness of Resolution

- Using Semantic Tree (assume the RHS of → is false)



# Theoretical basis of Resolution (cont …)

- Monotone Inference
  - Size of Knowledge Base goes on increasing as we proceed with resolution process since intermediate resolvents added to the knowledge base
- Non-monotone Inference
  - Size of Knowledge Base does not increase
  - Human beings use non-monotone inference

# Back to Himalayan Club

# Himalayan Club example

- Introduction through an example *(Zohar Manna, 1974):*
  - Problem: A, B and C belong to the Himalayan club. Every member in the club is either a mountain climber or a skier or both. A likes whatever B dislikes and dislikes whatever B likes. A likes rain and snow. No mountain climber likes rain. Every skier likes snow. *Is there a member who is a mountain climber and not a skier?*
- Given knowledge has:
  - Facts
  - Rules

# Example contd.

- Let *mc* denote mountain climber and *sk* denotes skier. Knowledge representation in the given problem is as follows:
  1. *member(A)*
  2. *member(B)*
  3. *member(C)*
  4. $\forall x[member(x) \rightarrow (mc(x) \vee sk(x))]$
  5. $\forall x[mc(x) \rightarrow \sim like(x,rain)]$
  6. $\forall x[sk(x) \rightarrow like(x, snow)]$
  7. $\forall x[like(B, x) \rightarrow \sim like(A, x)]$
  8. $\forall x[\sim like(B, x) \rightarrow like(A, x)]$
  9. *like(A, rain)*
  10. *like(A, snow)*
  11. *Question:* $\exists x[member(x) \wedge mc(x) \wedge \sim sk(x)]$
- We have to infer the $11^{th}$ expression from the given 10.
- Done through Resolution Refutation.

# Club example: Inferencing

1. $member(A)$
2. $member(B)$
3. $member(C)$
4. $\forall x[member(x) \rightarrow (mc(x) \vee sk(x))]$
   - Can be written as
   - $[member(x) \rightarrow (mc(x) \vee sk(x))]$
   - $\sim member(x) \vee mc(x) \vee sk(x)$
5. $\forall x[sk(x) \rightarrow lk(x, snow)]$
   - $\sim sk(x) \vee lk(x, snow)$
6. $\forall x[mc(x) \rightarrow \sim lk(x, rain)]$
   - $\sim mc(x) \vee \sim lk(x, rain)$
7. $\forall x[like(A, x) \rightarrow \sim lk(B, x)]$
   - $\sim like(A, x) \vee \sim lk(B, x)$

8. $\forall x[\sim lk(A,x) \rightarrow lk(B,x)]$

   – $lk(A,x) \vee lk(B,x)$

9. $lk(A,rain)$

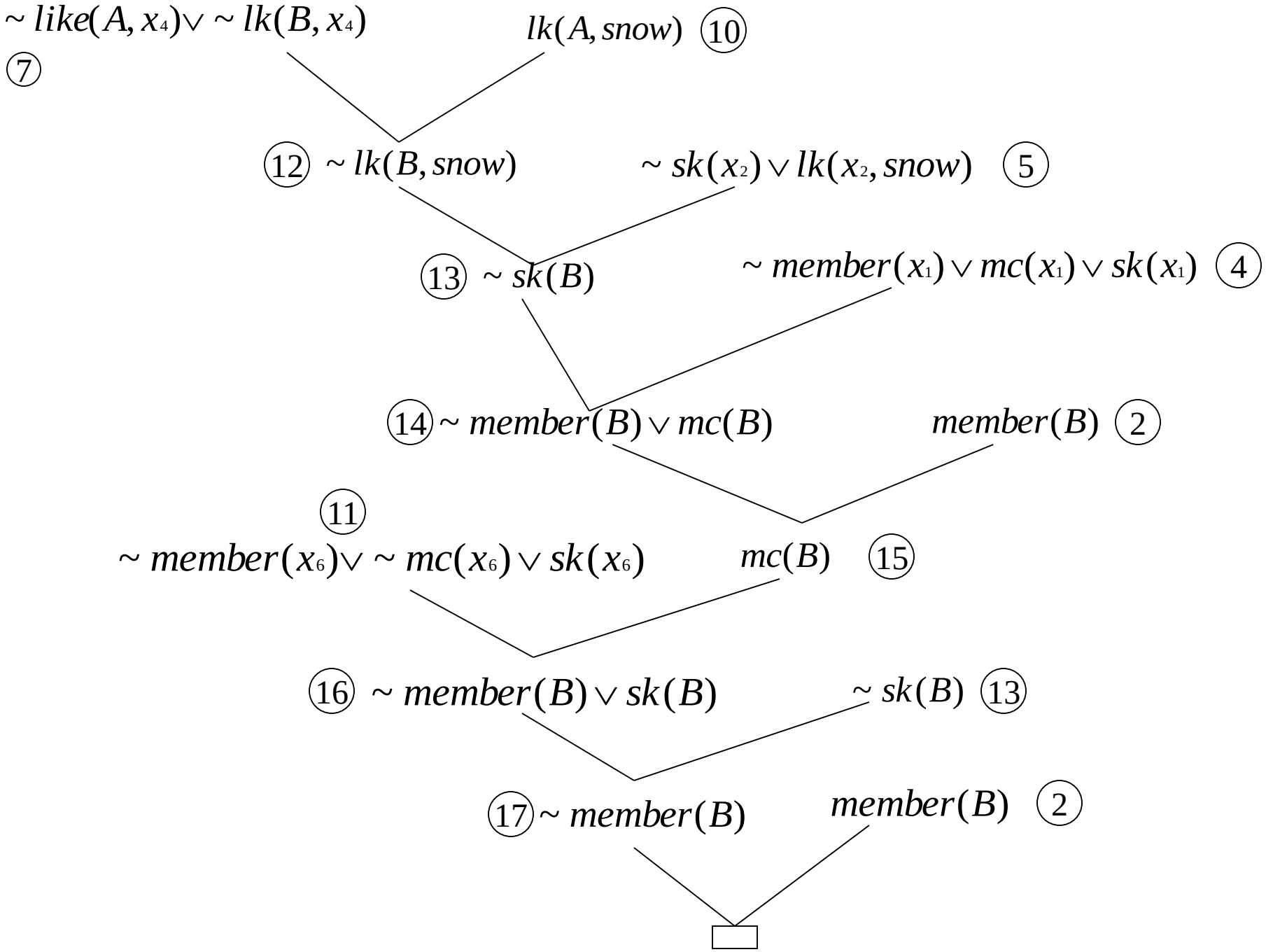
10. $lk(A,snow)$

11. $\exists x[member(x) \wedge mc(x) \wedge \sim sk(x)]$

   – Negate– $\forall x[\sim member(x) \vee \sim mc(x) \vee sk(x)]$

- Now standardize the variables apart which results in the following

1. $member(A)$
2. $member(B)$
3. $member(C)$
4. $\sim member(x_1) \vee mc(x_1) \vee sk(x_1)$
5. $\sim sk(x_2) \vee lk(x_2, snow)$
6. $\sim mc(x_3) \vee \sim lk(x_3, rain)$
7. $\sim like(A, x_4) \vee \sim lk(B, x_4)$
8. $lk(A, x_5) \vee lk(B, x_5)$
9. $lk(A, rain)$
10. $lk(A, snow)$
11. $\sim member(x_6) \vee \sim mc(x_6) \vee sk(x_6)$


~ like(A, x4) ∨ ~ lk(B, x4)
7
lk(A, snow)
10
12 ~ lk(B, snow)
~ sk(x2) ∨ lk(x2, snow)
5
13 ~ sk(B)
~ member(x1) ∨ mc(x1) ∨ sk(x1)
4
14 ~ member(B) ∨ mc(B)
member(B)
2
11
~ member(x6) ∨ ~ mc(x6) ∨ sk(x6)
mc(B)
15
16 ~ member(B) ∨ sk(B)
~ sk(B)
13
17 ~ member(B)
member(B)
2

# Well known examples in Predicate Calculus

- Man is mortal : rule

  $\forall x[man(x) \rightarrow mortal(x)]$

- shakespeare is a man

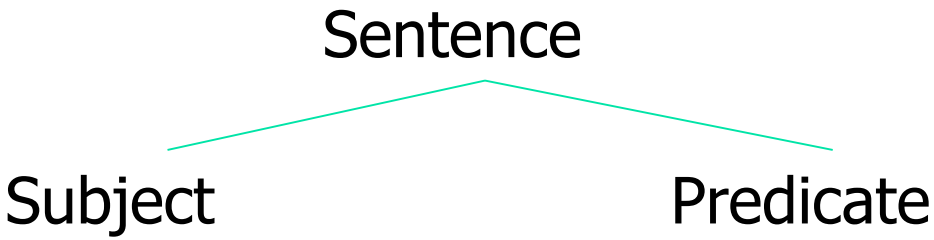
  man(shakespeare)

- To infer shakespeare is mortal

  mortal(shakespeare)

# Predicate Calculus: origin

- Predicate calculus originated in language




Grass is green

Subject

Predicate

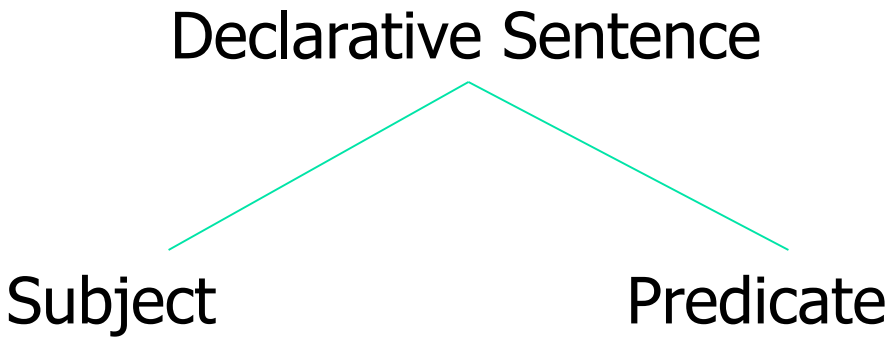
green(grass)

$P : D \rightarrow \{T, F\}$

$D$ : Domain

# Predicate Calculus: only for declarative sentences

- Is grass green? (Interrogative)
- Oh, grass is green! (Exclamatory)




- Grass which is supple is green

$$\forall x(\text{grass}(x)) \wedge \text{supple}(x) \rightarrow \text{green}(x))$$

# Predicate Calculus: more expressive power than propositional calculus

- 2 is even and is divisible by 2: P1
- 4 is even and is divisible by 2: P2
- 6 is even and is divisible by 2: P3

Generalizing,

$$\forall x((Integer(x) \wedge even(x) \Rightarrow divides(2, x))$$

# Predicate Calculus: finer than propositional calculus

1. Finer Granularity (Grass is green, ball is green, leaf is green (green(x)))
2. Succinct description for infinite number of statements which would need $\propto$ number of properties

3 place predicate

Example: x gives y to z give(x,y,z)

4 place predicate

Example: x gives y to z through w give(x,y,z,w)

# Double causative in Hindi giving rise to higher place predicates

- जॉन ने खाना खाया
  John ne khana khaya
  John <CM> food ate
  John ate food
  *eat(John, food)*

- जॉन ने जैक को खाना खिलाया
  John ne Jack ko khana khilaya
  John <CM> Jack <CM> food fed
  John fed Jack
  *eat(John, Jack, food)*

- जॉन ने जैक को जिल के द्वारा खाना खिलाया
  John ne Jack ko Jill ke dvara khana khilaya
  John <CM> Jack <CM> Jill <CM> food made-to-eat
  John fed Jack through Jill
  *eat(John, Jack, Jill, food)*

# PC primitive: N-ary Predicate

$$P(a_1, \ldots a_n)$$

$$P : D^n \rightarrow \{T, F\}$$

- Arguments of predicates can be variables and constants
- Ground instances : Predicate all whose arguments are constants

# N-ary Functions

$$f : D^n \rightarrow D$$

*president(India) : Droupadi Murmu*

- Constants & Variables : Zero-order objects
- Predicates & Functions : First-order objects

*Prime minister of Fiji is older than the president of Fiji*

*older(prime_minister(Fiji), president(Fiji))*

# Operators

$$\wedge \vee \sim \oplus \forall \rightarrow \exists$$

- Universal Quantifier $\forall$
- Existential Quantifier $\exists$

All men are mortal

$$\forall x[man(x) \rightarrow mortal(x)]$$

Some men are rich

$$\exists x[man(x) \wedge rich(x)]$$

# Tautologies

$$\sim \forall x(p(x)) \rightarrow \exists x(\sim p(x))$$

$$\sim \exists x(p(x)) \rightarrow \forall x(\sim p(x))$$

- 2nd tautology in English:
  - *Not a single man in this village is educated implies all men in this village are uneducated*
- Tautologies are important instruments of logic, but uninteresting statements!

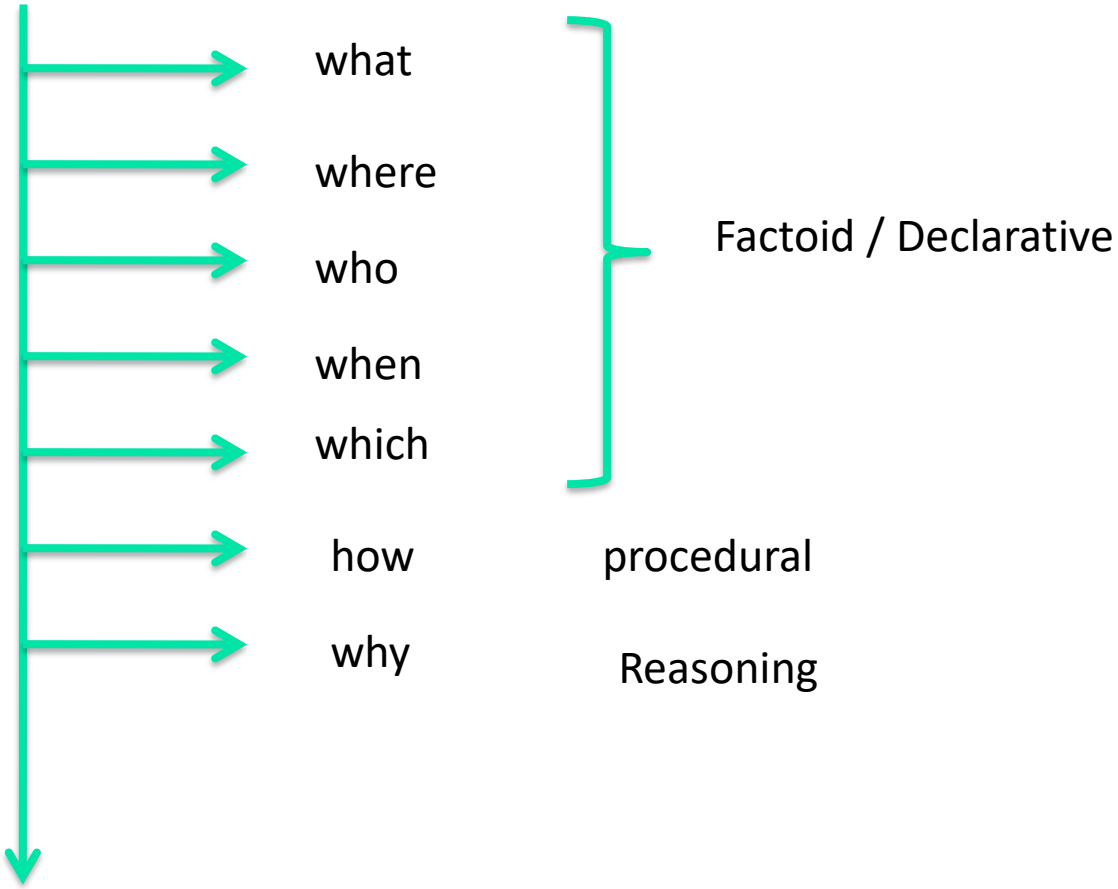
# Inferencing: Forward Chaining

- *man(x) → mortal(x)*
  - *Dropping the quantifier, implicitly Universal quantification assumed*
  - *man(shakespeare)*
- Goal mortal(shakespeare)
  - Found in one step
  - x = shakespeare, unification

# Backward Chaining

- $man(x) \rightarrow mortal(x)$
- Goal mortal(shakespeare)
  - x = shakespeare
  - Travel back over and hit the fact asserted
  - man(shakespeare)

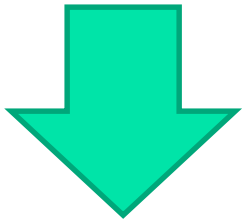
# Wh-Questions and Knowledge



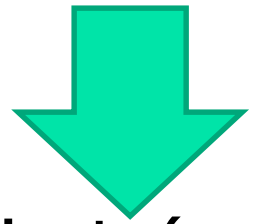
# Fixing Predicates

- Natural Sentences

<Subject> <verb> <object>



Verb(subject,object)



predicate(subject)

# Examples

- Ram is a boy
  - Boy(Ram)?
  - Is_a(Ram,boy)?
- Ram Plays Football
  - Plays(Ram,football)?
  - Plays_football(Ram)?

# Knowledge Representation of Complex Sentence

- *"In every city there is a thief who is beaten by every policeman in the city"*

# Knowledge Representation of Complex Sentence

- *"In every city there is a thief who is beaten by every policeman in the city"*

$$\forall x[\text{city}(x) \rightarrow \{\exists y((\text{thief}(y) \wedge \text{lives\_in}\ (y, x)) \wedge \forall z(\text{policeman}(z, x) \rightarrow \text{beaten\_by}\ (z, y)))\}]$$

# Interpretation in Logic

- Logical expressions or formulae are "FORMS" (placeholders) for whom contents are created through interpretation.
- Example:

$$\exists F\left[\left\{F(a)=b\right\}\wedge\forall x\left\{P(x)\rightarrow\left(F(x)=g\left(x,F\left(h(x)\right)\right)\right)\right\}\right]$$

- This is a Second Order Predicate Calculus formula.
- Quantification on 'F' which is a function.

# Examples

- Interpretation:1

  $D=N$ (natural numbers)

  $a = 0$ and $b = 1$

  $x \in N$

  $P(x)$ stands for $x > 0$

  $g(m,n)$ stands for $(m \times n)$

  $h(x)$ stands for $(x - 1)$

- Above interpretation defines **Factorial**

# Examples (contd.)

- Interpretation:2

  $D$={strings)

  $a = b = \lambda$

  $P(x)$ stands for "x is a non empty string"

  $g(m, n)$ stands for "append head of m to n"

  $h(x)$ stands for *tail(x)*

- Above interpretation defines "reversing a string"

# Search inside Chess!

Dion Reji & Soham Dahane

IIT Bombay

February 18, 2025

# Games as Search Problems

- Games can be formulated as search problems with:

# Games as Search Problems

- Games can be formulated as search problems with:
  - **State Space**

# Games as Search Problems

- Games can be formulated as search problems with:
  - **State Space**
  - **Operators**

# Games as Search Problems

- Games can be formulated as search problems with:
  - **State Space**
  - **Operators**
  - **Initial State**

# Games as Search Problems

- Games can be formulated as search problems with:
  - **State Space**
  - **Operators**
  - **Initial State**
  - **Goal States**

# Games as Search Problems

- Games can be formulated as search problems with:
  - **State Space**
  - **Operators**
  - **Initial State**
  - **Goal States**
- We consider games with two players - Black and White

# Games as Search Problems

- Games can be formulated as search problems with:
  - **State Space**
  - **Operators**
  - **Initial State**
  - **Goal States**
- We consider games with two players - Black and White
- We visualise these two player games in a game graph

# State Space Representation

- State: Complete description of game position

# State Space Representation

- State: Complete description of game position
- Components:

# State Space Representation

- State: Complete description of game position
- Components:
  - Current configuration of the game arena

# State Space Representation

- State: Complete description of game position
- Components:
  - Current configuration of the game arena
  - Player to move

# State Space Representation

- State: Complete description of game position
- Components:
  - Current configuration of the game arena
  - Player to move
  - Additional information (e.g., castling rights)

# State Space Representation

- State: Complete description of game position
- Components:
  - Current configuration of the game arena
  - Player to move
  - Additional information (e.g., castling rights)
- State transitions occur through operators

# Operators in Games

- Operators = Legal moves. What do you mean by legal?

# Operators in Games

- Operators = Legal moves. What do you mean by legal?
- Two distinct sets:

# Operators in Games

- Operators = Legal moves. What do you mean by legal?
- Two distinct sets:
  - White's legal moves ($O_w$)

# Operators in Games

- Operators = Legal moves. What do you mean by legal?
- Two distinct sets:
  - White's legal moves ($O_w$)
  - Black's legal moves ($O_b$)

# Operators in Games

- Operators = Legal moves. What do you mean by legal?
- Two distinct sets:
  - White's legal moves ($O_w$)
  - Black's legal moves ($O_b$)
- At each state:

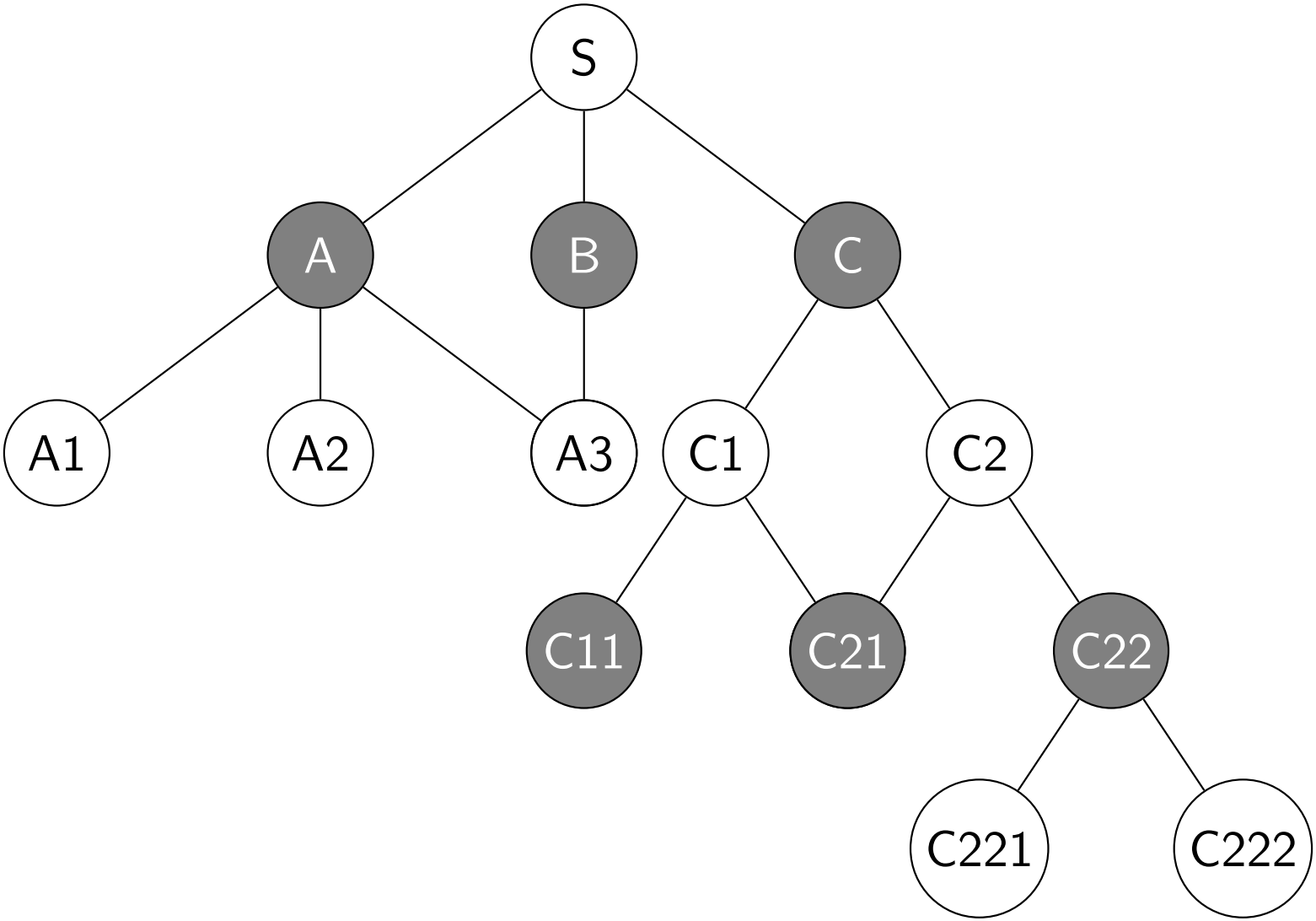
# Operators in Games

- Operators = Legal moves. What do you mean by legal?
- Two distinct sets:
  - White's legal moves ($O_w$)
  - Black's legal moves ($O_b$)
- At each state:
  - Only one set is applicable

# Operators in Games

- Operators = Legal moves. What do you mean by legal?
- Two distinct sets:
  - White's legal moves ($O_w$)
  - Black's legal moves ($O_b$)
- At each state:
  - Only one set is applicable
  - Determined by whose turn it is

# A Simple Game with Two Players



# Chess as a Search Problem

- State space:

# Chess as a Search Problem

- State space:
  - 8×8 board configuration

# Chess as a Search Problem

- State space:
  - 8×8 board configuration
  - Position of all pieces

# Chess as a Search Problem

- State space:
  - 8×8 board configuration
  - Position of all pieces
  - Turn indicator

# Chess as a Search Problem

- State space:
  - 8×8 board configuration
  - Position of all pieces
  - Turn indicator
  - Castling rights, en passant possibilities

# Chess as a Search Problem

- State space:
  - 8×8 board configuration
  - Position of all pieces
  - Turn indicator
  - Castling rights, en passant possibilities
- Initial state: Standard chess setup

# Chess as a Search Problem

- State space:
  - 8×8 board configuration
  - Position of all pieces
  - Turn indicator
  - Castling rights, en passant possibilities
- Initial state: Standard chess setup
- Goal states: Checkmate positions

# Chess Operators

- White's operators ($O_w$):

# Chess Operators

- White's operators ($O_w$):
  - All legal moves for white pieces

# Chess Operators

- White's operators ($O_w$):
  - All legal moves for white pieces
  - Example: e2-e4, Nf3, O-O

# Chess Operators

- White's operators ($O_w$):
  - All legal moves for white pieces
  - Example: e2-e4, Nf3, O-O
- Black's operators ($O_b$):

# Chess Operators

- White's operators ($O_w$):
  - All legal moves for white pieces
  - Example: e2-e4, Nf3, O-O
- Black's operators ($O_b$):
  - All legal moves for black pieces

# Chess Operators

- White's operators ($O_w$):
  - All legal moves for white pieces
  - Example: e2-e4, Nf3, O-O
- Black's operators ($O_b$):
  - All legal moves for black pieces
  - Same move types as white

# Chess Operators

- White's operators ($O_w$):
  - All legal moves for white pieces
  - Example: e2-e4, Nf3, O-O
- Black's operators ($O_b$):
  - All legal moves for black pieces
  - Same move types as white
- Each operator transforms current state to new state

# Example

Consider the game of chess with the following configuration of the board.

# Example

Consider the game of chess with the following configuration of the board.

# Example

Consider the game of chess with the following configuration of the board. Now Black player makes a move (Bc8 $\rightarrow$ Bg4) indicating an action and taking to a new game state.

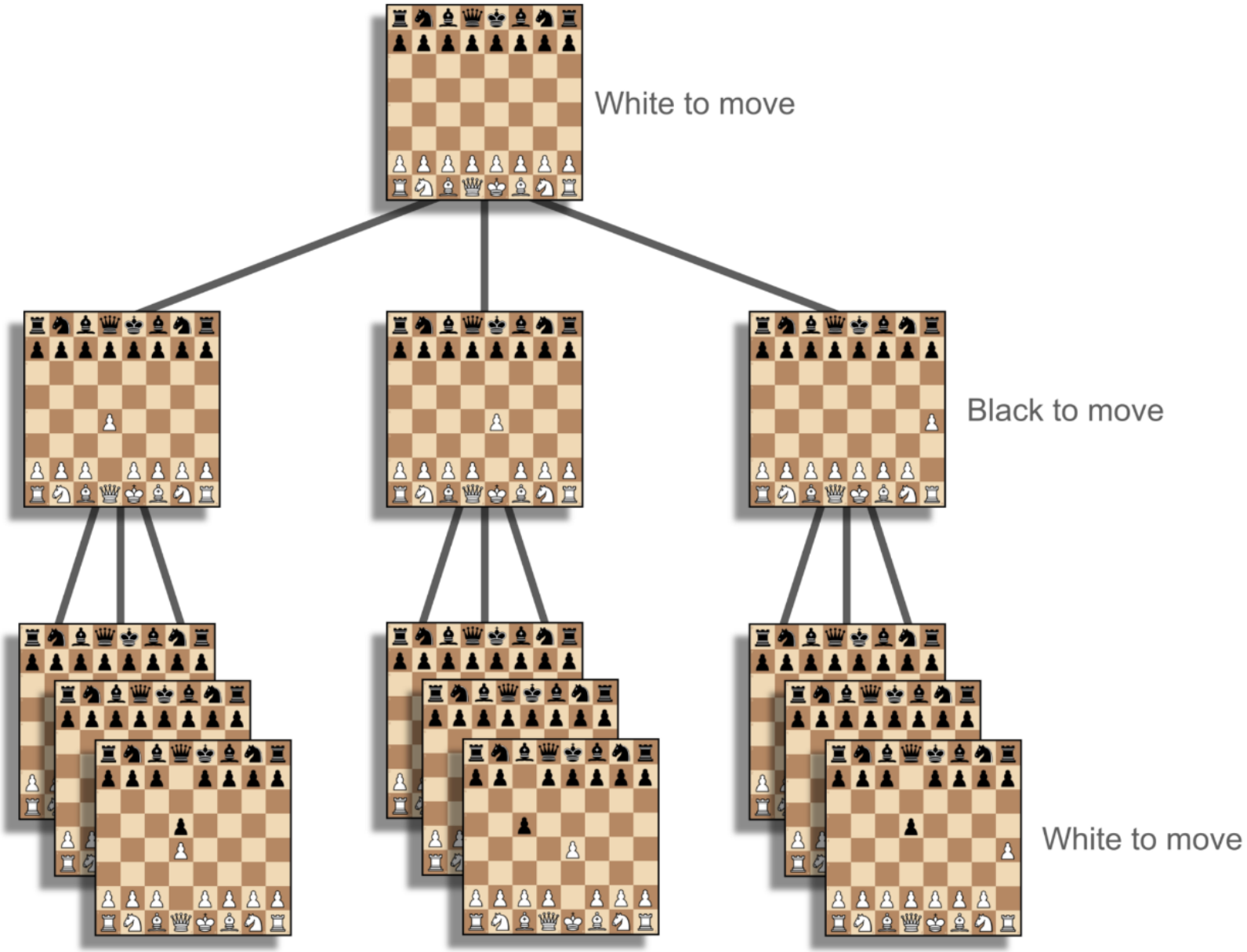
# Example

Consider the game of chess with the following configuration of the board. Now Black player makes a move (Bc8 $\rightarrow$ Bg4) indicating an action and taking to a new game state.

# Example

Consider the game of chess with the following configuration of the board. Now Black player makes a move (Bc8 $\rightarrow$ Bg4) indicating an action and taking to a new game state.

# Snippet of Game Graph of chess



## Question

How will we solve a game?

What do you mean by solving?

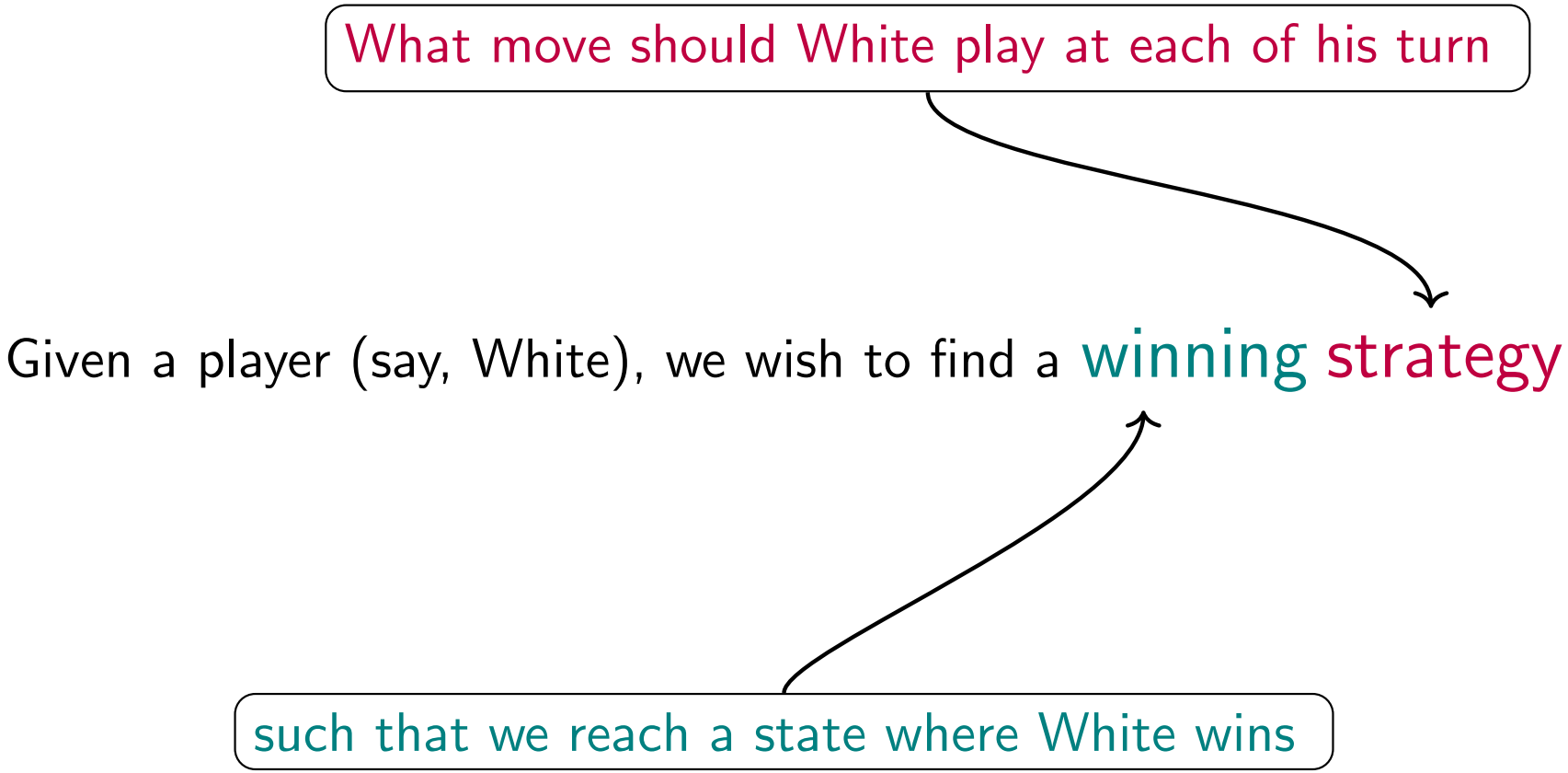
# Solving a Game

Given a player (say, White), we wish to find a winning strategy

# Solving a Game


Given a player (say, White), we wish to find a winning strategy

# Solving a Game



# Search and Games

Can we solve a game by Searching?

# Search and Games

Can we solve a game by Searching?

- Obtain the complete game graph - States, Operator, Initial State, Goal States

# Search and Games

Can we solve a game by Searching?

- Obtain the complete game graph - States, Operator, Initial State, Goal States
- What will be our Goal States?

# Search and Games

Can we solve a game by Searching?

- Obtain the complete game graph - States, Operator, Initial State, Goal States
- What will be our Goal States? All states where White wins.

# Search and Games

Can we solve a game by Searching?

- Obtain the complete game graph - States, Operator, Initial State, Goal States
- What will be our Goal States? All states where White wins.
- We start searching the Goal States from Initial State

# Search and Games

Can we solve a game by Searching?

- Obtain the complete game graph - States, Operator, Initial State, Goal States
- What will be our Goal States? All states where White wins.
- We start searching the Goal States from Initial State
- What will our search algorithm return?

# Search and Games

Can we solve a game by Searching?

- Obtain the complete game graph - States, Operator, Initial State, Goal States
- What will be our Goal States? All states where White wins.
- We start searching the Goal States from Initial State
- What will our search algorithm return? Path from start state to Goal states.

Can we getting a winning strategy from the path found by our search algorithm?

# Limitations of Search

- For most games the state space is enoromously large. For chess $\sim 10^{40}$ nodes - practically impossible to search.

# Limitations of Search

- For most games the state space is enoromously large. For chess $\sim 10^{40}$ nodes - practically impossible to search.
- Search might not give us a strategy to solve a game. We can rather focus on simpler problems!

# Limitations of Search

- For most games the state space is enoromously large. For chess $\sim 10^{40}$ nodes - practically impossible to search.
- Search might not give us a strategy to solve a game. We can rather focus on simpler problems!
- A* excels on **simplified subproblems** or **variants**:
  1. **Knight's Shortest Path Problem**
  2. **Endgame Tablebase Approximation**

# Knight's Shortest Path Problem

- **Objective:** Find the minimum number of moves for a knight to travel from a start square to a target square on an $8 \times 8$ chessboard.

# Knight's Shortest Path Problem

- **Objective:** Find the minimum number of moves for a knight to travel from a start square to a target square on an $8 \times 8$ chessboard.
- **State Space:**

$$S = \{(i, j) \mid 1 \leq i, j \leq 8\}$$

# Knight's Shortest Path Problem

- **Objective:** Find the minimum number of moves for a knight to travel from a start square to a target square on an $8 \times 8$ chessboard.
- **State Space:**

$$S = \{(i, j) \mid 1 \leq i, j \leq 8\}$$

- **Actions:** Legal knight moves:

$$(x+2,\, y+1),\ (x+2,\, y-1),\ (x-2,\, y+1),\ (x-2,\, y-1),$$
$$(x+1,\, y+2),\ (x+1,\, y-2),\ (x-1,\, y+2),\ (x-1,\, y-2)$$

# Knight's Shortest Path Problem

- **Objective:** Find the minimum number of moves for a knight to travel from a start square to a target square on an $8 \times 8$ chessboard.
- **State Space:**

$$S = \{(i, j) \mid 1 \leq i, j \leq 8\}$$

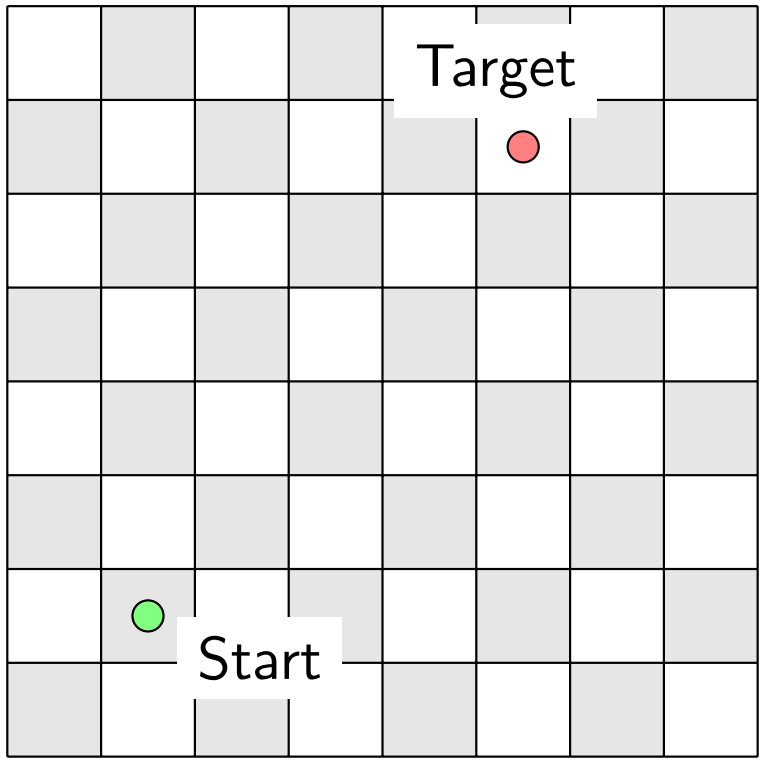
- **Actions:** Legal knight moves:

$$(x+2,\, y+1),\ (x+2,\, y-1),\ (x-2,\, y+1),\ (x-2,\, y-1),$$
$$(x+1,\, y+2),\ (x+1,\, y-2),\ (x-1,\, y+2),\ (x-1,\, y-2)$$

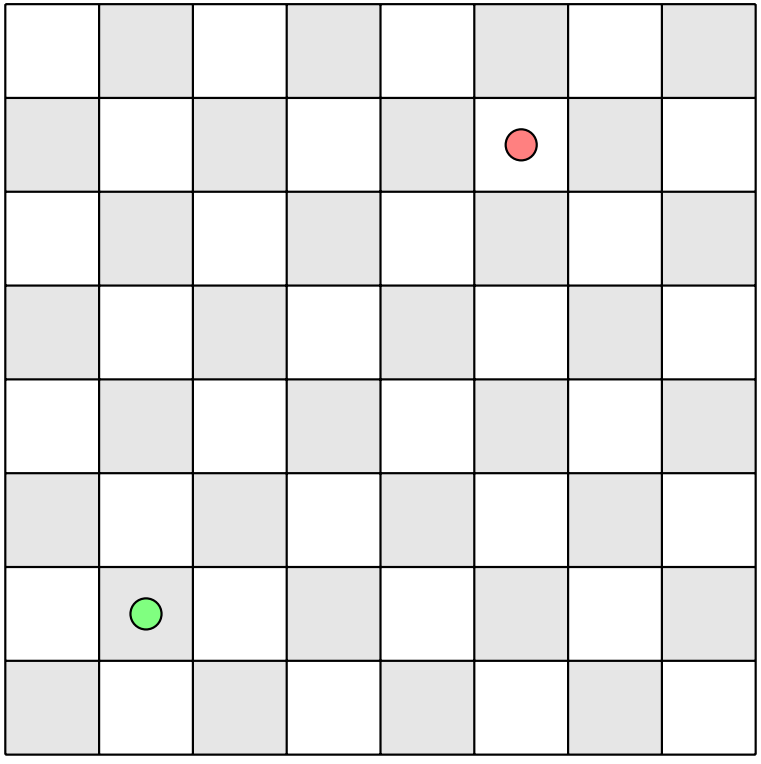
- **Cost Function:** $g(n) =$ number of moves taken so far.
- **Heuristic:** A simple estimate:

$$h(n) \approx \frac{|x - x_t| + |y - y_t|}{3}$$

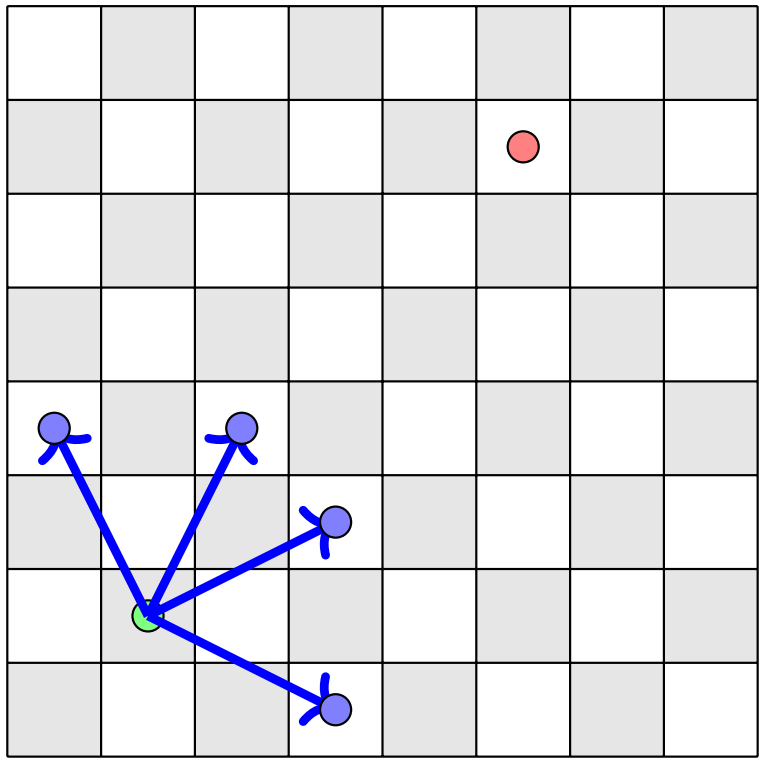
# Knight's Shortest Path - Start and Target



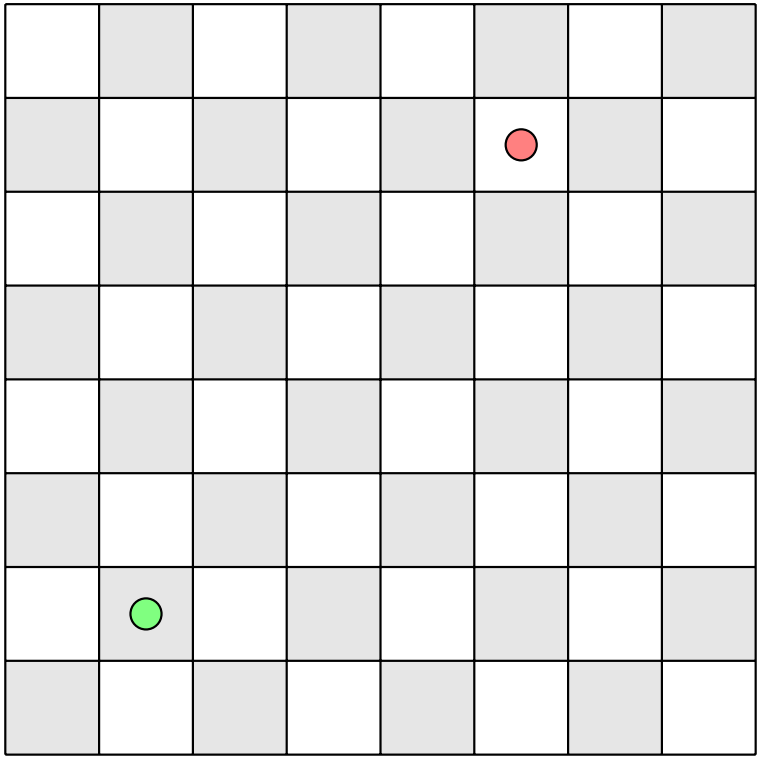
# Knight's Shortest Path - Possible Moves



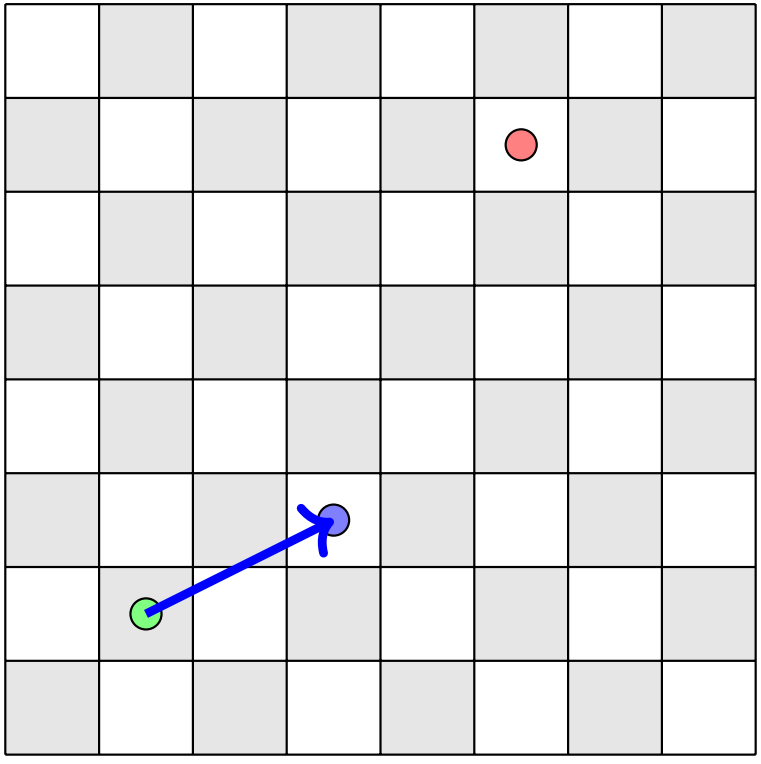
# Knight's Shortest Path - Possible Moves



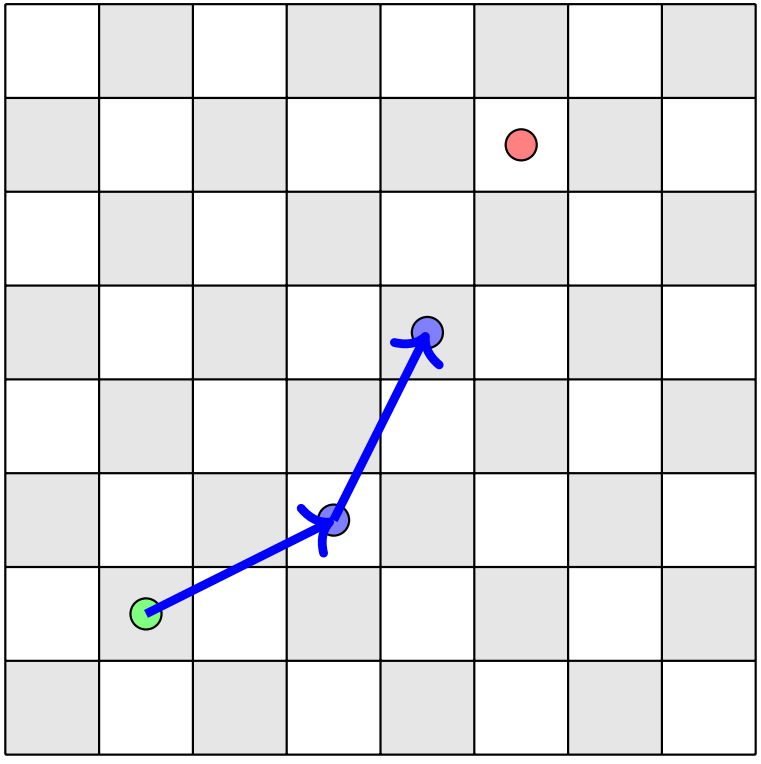
# Knight's Shortest Path - Shortest Path



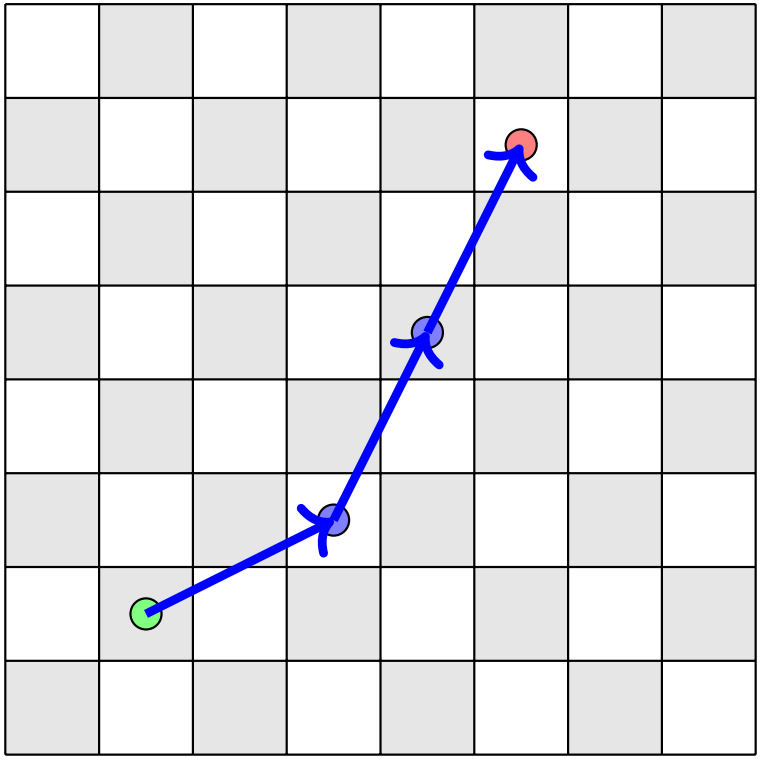
# Knight's Shortest Path - Shortest Path



# Knight's Shortest Path - Shortest Path



# Knight's Shortest Path - Shortest Path



# Endgame Tablebase Approximation

- **Objective:** Determine the sequence of moves to checkmate in a simplified endgame (e.g., King & Queen vs. King).

# Endgame Tablebase Approximation

- **Objective:** Determine the sequence of moves to checkmate in a simplified endgame (e.g., King & Queen vs. King).
- **State Space:** Reduced set of positions with only a few pieces.

# Endgame Tablebase Approximation

- **Objective:** Determine the sequence of moves to checkmate in a simplified endgame (e.g., King & Queen vs. King).
- **State Space:** Reduced set of positions with only a few pieces.
- **Actions:** Legal moves available in the given endgame.

# Endgame Tablebase Approximation

- **Objective:** Determine the sequence of moves to checkmate in a simplified endgame (e.g., King & Queen vs. King).
- **State Space:** Reduced set of positions with only a few pieces.
- **Actions:** Legal moves available in the given endgame.
- **Cost Function:** $g(n) =$ number of moves made.

# Endgame Tablebase Approximation

- **Objective:** Determine the sequence of moves to checkmate in a simplified endgame (e.g., King & Queen vs. King).
- **State Space:** Reduced set of positions with only a few pieces.
- **Actions:** Legal moves available in the given endgame.
- **Cost Function:** $g(n)$ = number of moves made.
- **Heuristic:** Estimate based on:
  - Material advantage.

# Endgame Tablebase Approximation

- **Objective:** Determine the sequence of moves to checkmate in a simplified endgame (e.g., King & Queen vs. King).
- **State Space:** Reduced set of positions with only a few pieces.
- **Actions:** Legal moves available in the given endgame.
- **Cost Function:** $g(n)$ = number of moves made.
- **Heuristic:** Estimate based on:
  - Material advantage.
  - Piece mobility.

# Endgame Tablebase Approximation

- **Objective:** Determine the sequence of moves to checkmate in a simplified endgame (e.g., King & Queen vs. King).
- **State Space:** Reduced set of positions with only a few pieces.
- **Actions:** Legal moves available in the given endgame.
- **Cost Function:** $g(n)$ = number of moves made.
- **Heuristic:** Estimate based on:
  - Material advantage.
  - Piece mobility.
  - Proximity to key squares.

# Endgame Tablebase Approximation

- **Objective:** Determine the sequence of moves to checkmate in a simplified endgame (e.g., King & Queen vs. King).
- **State Space:** Reduced set of positions with only a few pieces.
- **Actions:** Legal moves available in the given endgame.
- **Cost Function:** $g(n)$ = number of moves made.
- **Heuristic:** Estimate based on:
  - Material advantage.
  - Piece mobility.
  - Proximity to key squares.
- Think: Where is it be incorporated to avoid stalemate in the game?

# Endgame Tablebase Approximation

- In any game the heuristic incorporates multiple strategic factors to guide A* search.

# Endgame Tablebase Approximation

- In any game the heuristic incorporates multiple strategic factors to guide A* search.
- **Example Heuristic Function:**

$$h(n) = d(n, \text{edge}) + \text{Mobility}(B_k) + M$$

where:

- $d(n, \text{edge})$ is the distance of the Black king from the nearest edge.
- $\text{Mobility}(B_k)$ is the number of legal moves available to the Black king.
- $M$ is a bonus term if the White king is supporting the queen effectively which can be the manhattan distance between them for example.

# Endgame Tablebase Approximation

- In any game the heuristic incorporates multiple strategic factors to guide A* search.
- **Example Heuristic Function:**

$$h(n) = d(n, \text{edge}) + \text{Mobility}(B_k) + M$$

where:

  - $d(n, \text{edge})$ is the distance of the Black king from the nearest edge.
  - $\text{Mobility}(B_k)$ is the number of legal moves available to the Black king.
  - $M$ is a bonus term if the White king is supporting the queen effectively which can be the manhattan distance between them for example.
- **Heuristic Rationale:**
  - Encourages restricting the Black king's movement.

# Endgame Tablebase Approximation

- In any game the heuristic incorporates multiple strategic factors to guide A* search.
- **Example Heuristic Function:**

$$h(n) = d(n, \text{edge}) + \text{Mobility}(B_k) + M$$

where:

  - $d(n, \text{edge})$ is the distance of the Black king from the nearest edge.
  - $\text{Mobility}(B_k)$ is the number of legal moves available to the Black king.
  - $M$ is a bonus term if the White king is supporting the queen effectively which can be the manhattan distance between them for example.
- **Heuristic Rationale:**
  - Encourages restricting the Black king's movement.
  - Drives the Black king toward a checkmate-friendly position.

# Endgame Tablebase Approximation

- In any game the heuristic incorporates multiple strategic factors to guide A* search.
- **Example Heuristic Function:**

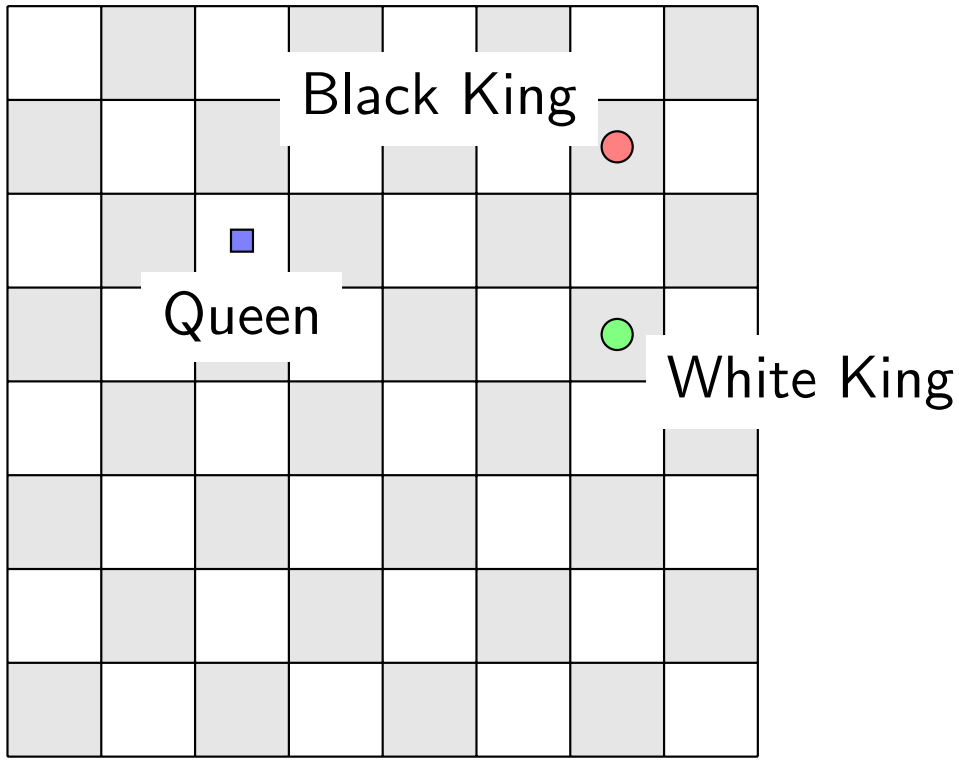
$$h(n) = d(n, \text{edge}) + \text{Mobility}(B_k) + M$$

where:

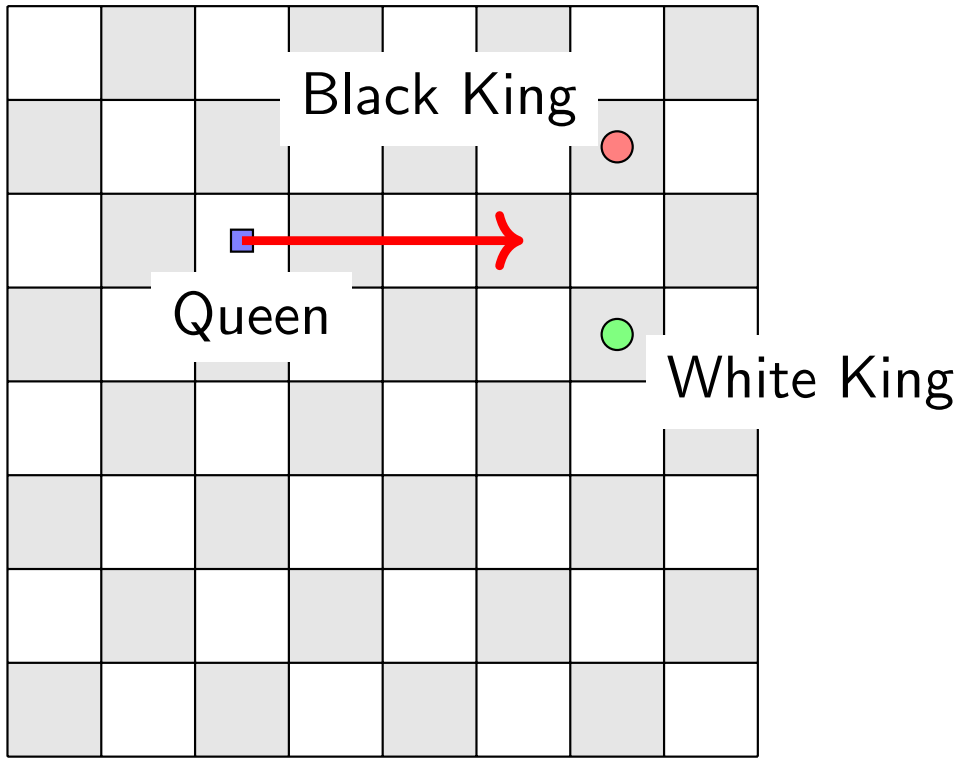
  - $d(n, \text{edge})$ is the distance of the Black king from the nearest edge.
  - $\text{Mobility}(B_k)$ is the number of legal moves available to the Black king.
  - $M$ is a bonus term if the White king is supporting the queen effectively which can be the manhattan distance between them for example.

- **Heuristic Rationale:**
  - Encourages restricting the Black king's movement.
  - Drives the Black king toward a checkmate-friendly position.
  - Optimizes coordination between White's king and queen.

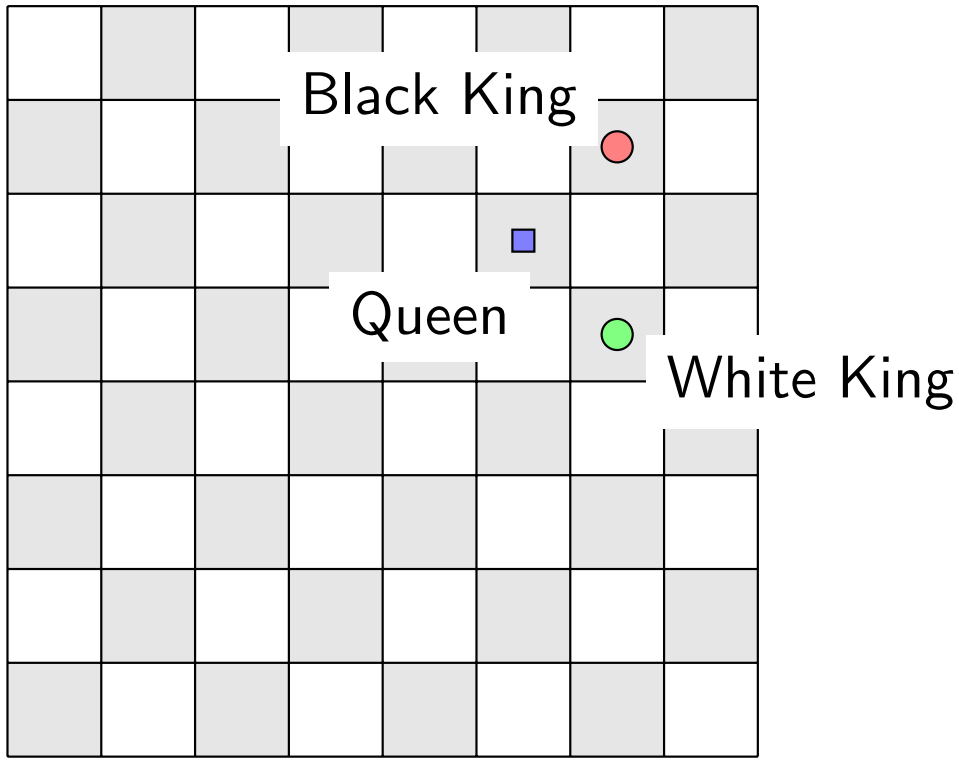
# Endgame Tablebase Approximation



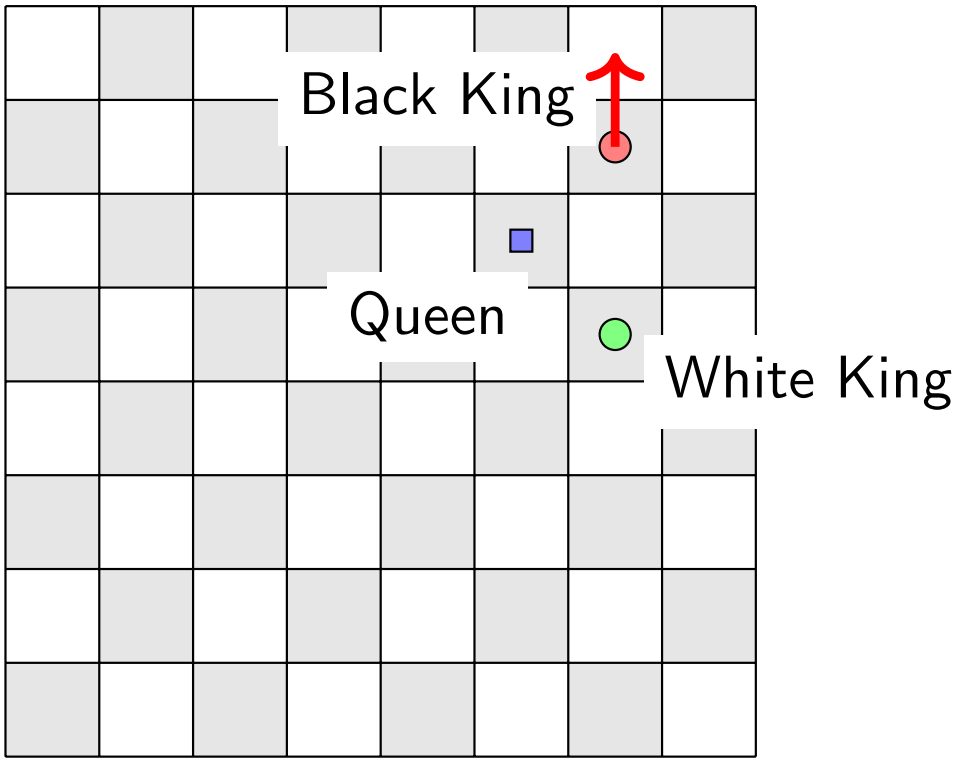
# Endgame Tablebase Approximation



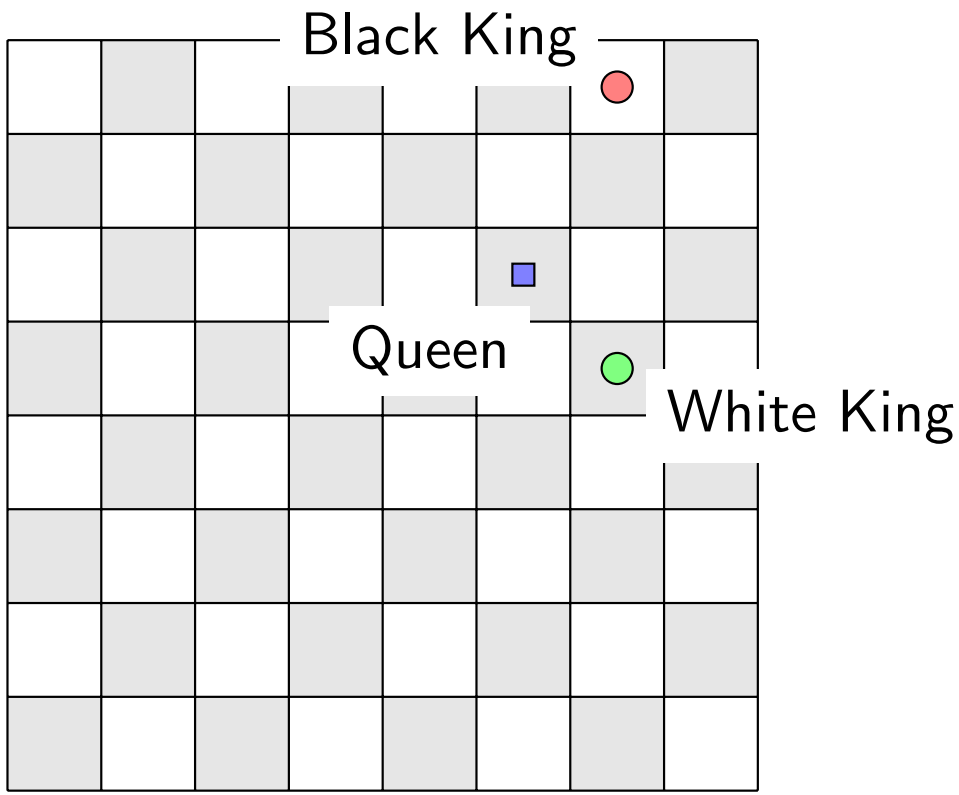
# Endgame Tablebase Approximation



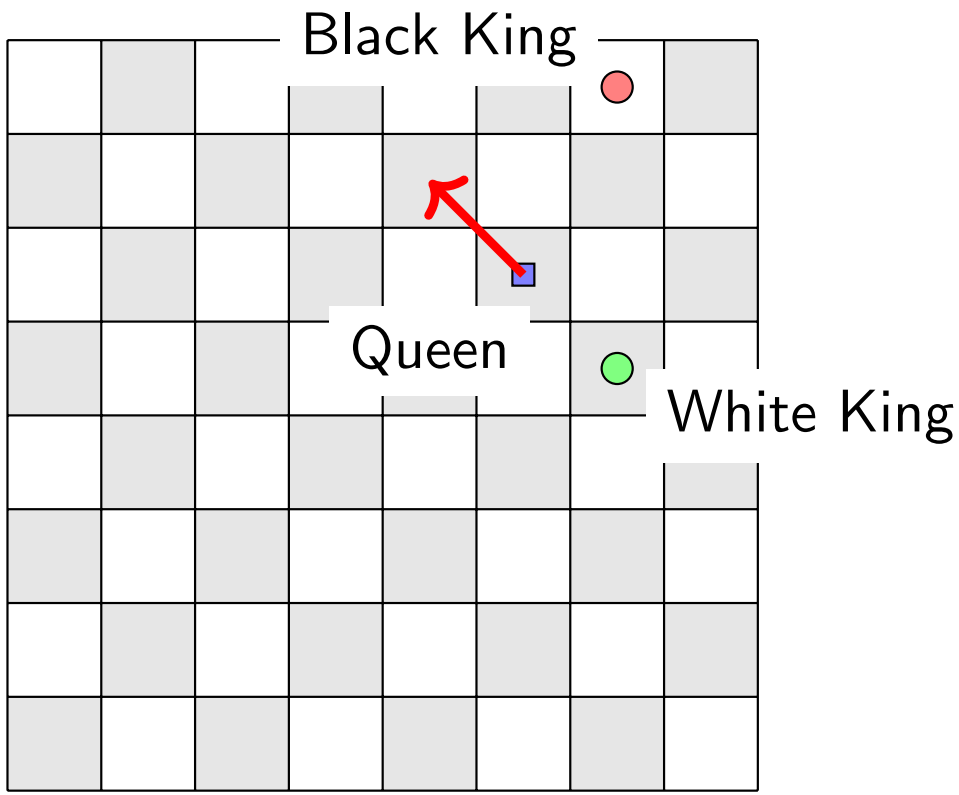
# Endgame Tablebase Approximation



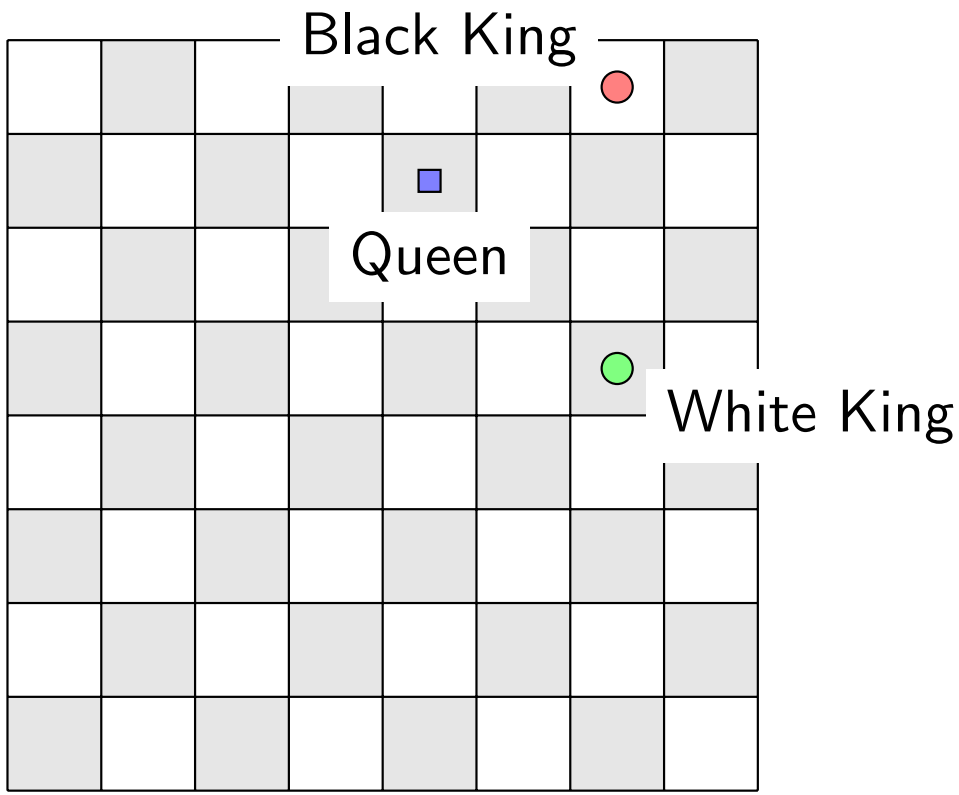
# Endgame Tablebase Approximation



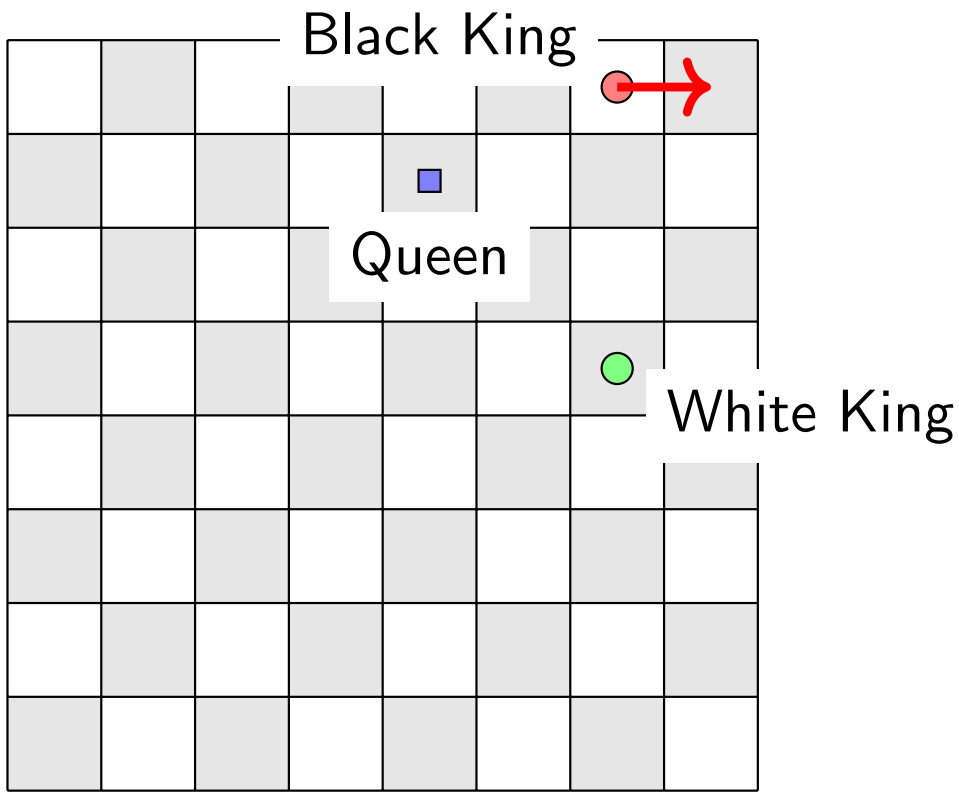
# Endgame Tablebase Approximation



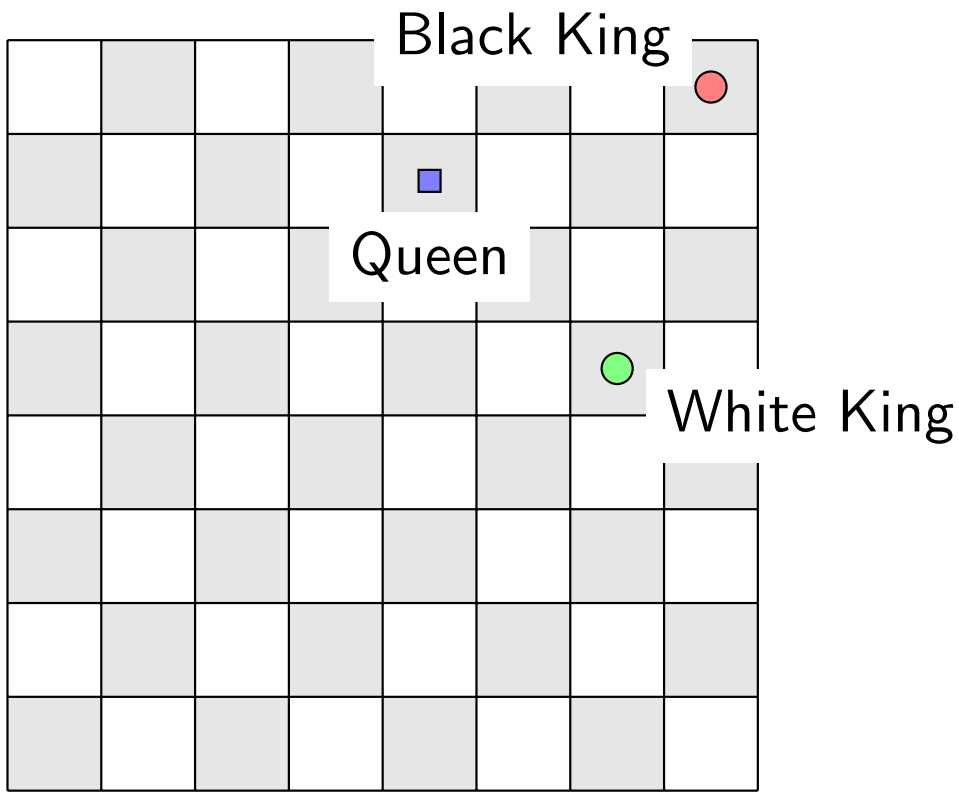
# Endgame Tablebase Approximation



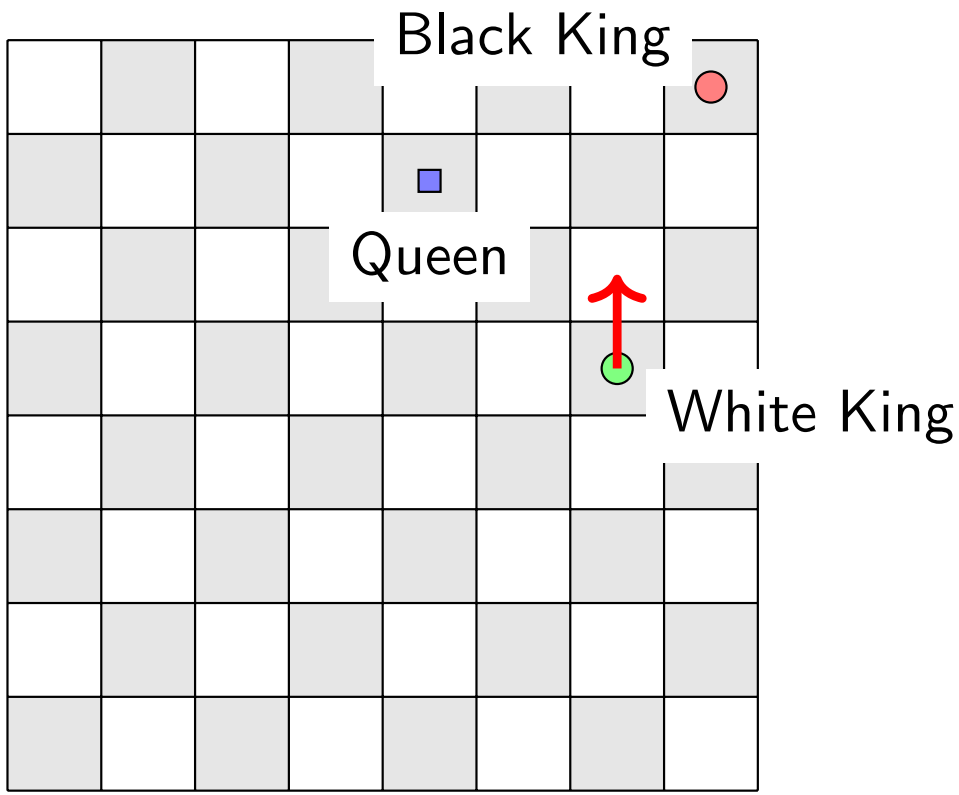
# Endgame Tablebase Approximation



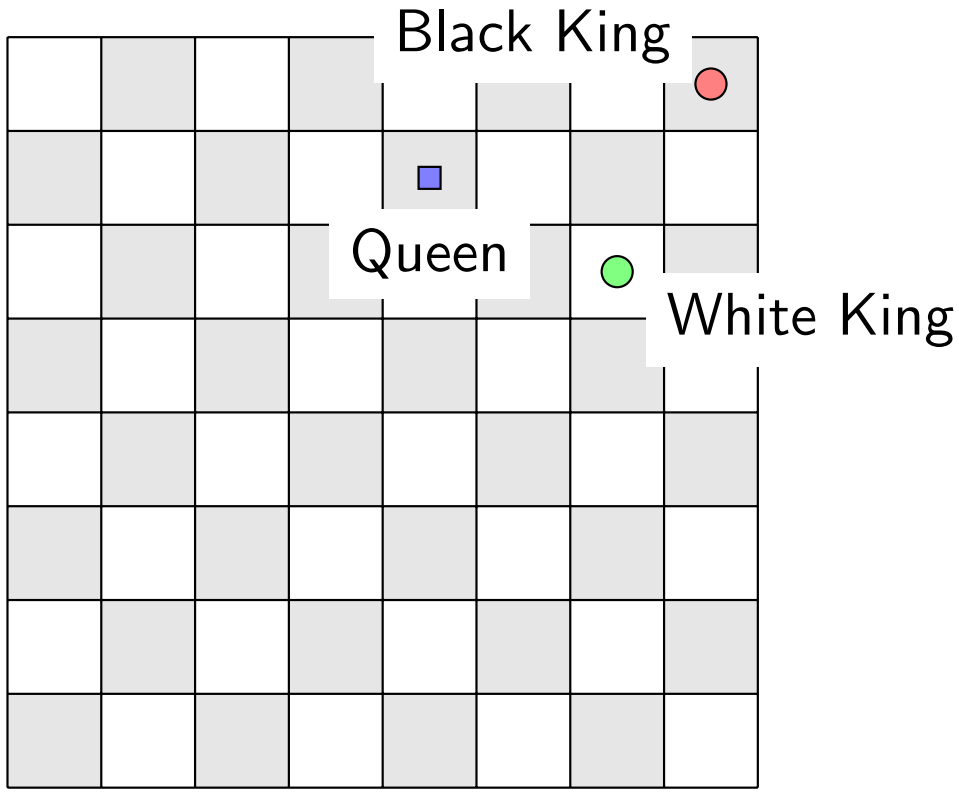
# Endgame Tablebase Approximation



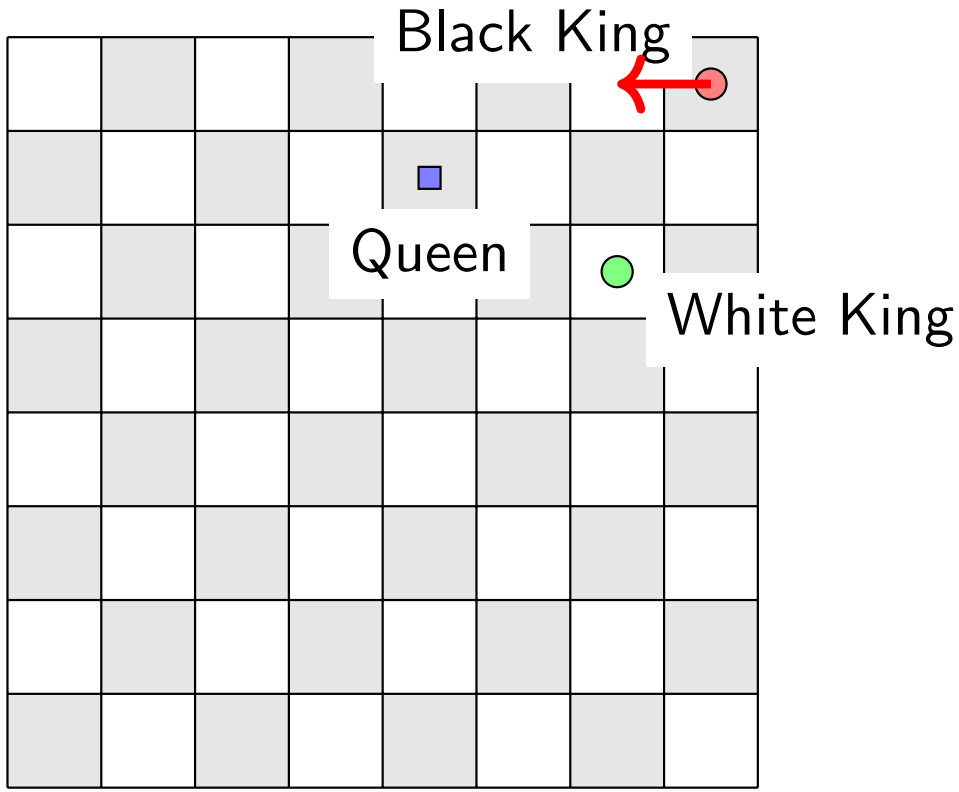
# Endgame Tablebase Approximation



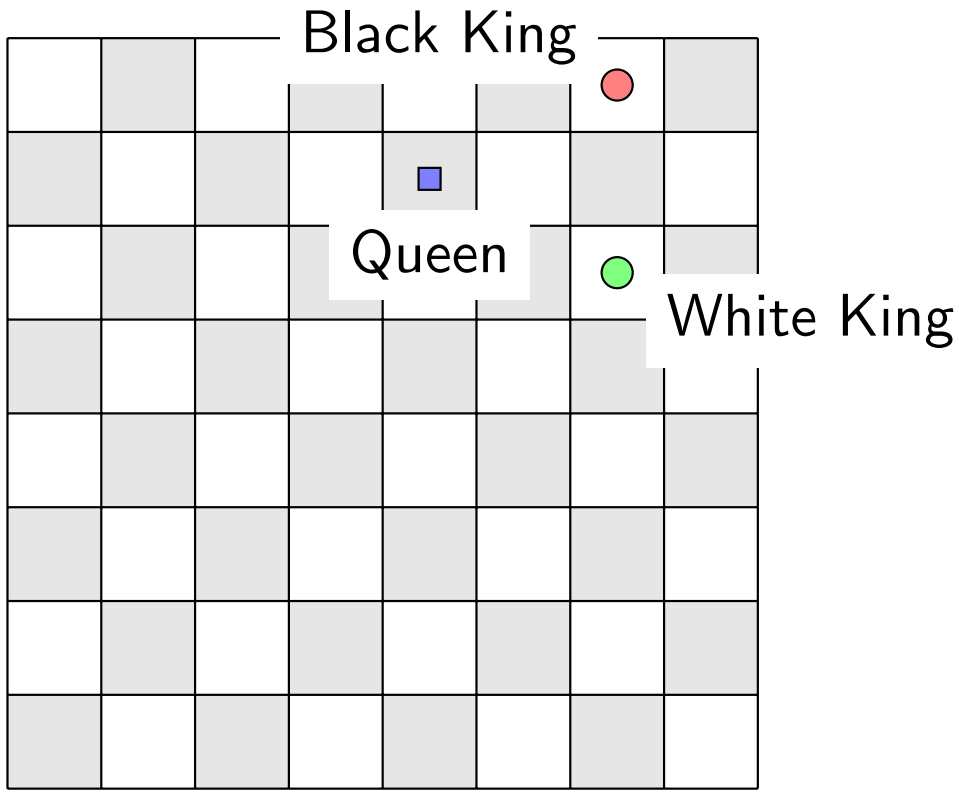
# Endgame Tablebase Approximation



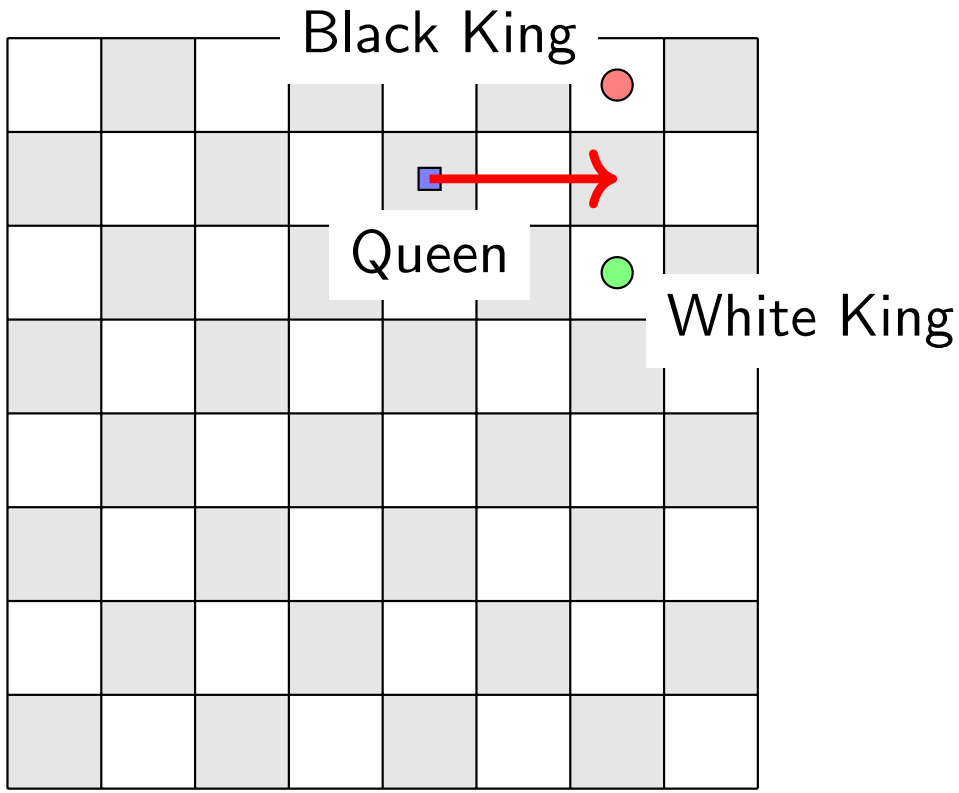
# Endgame Tablebase Approximation



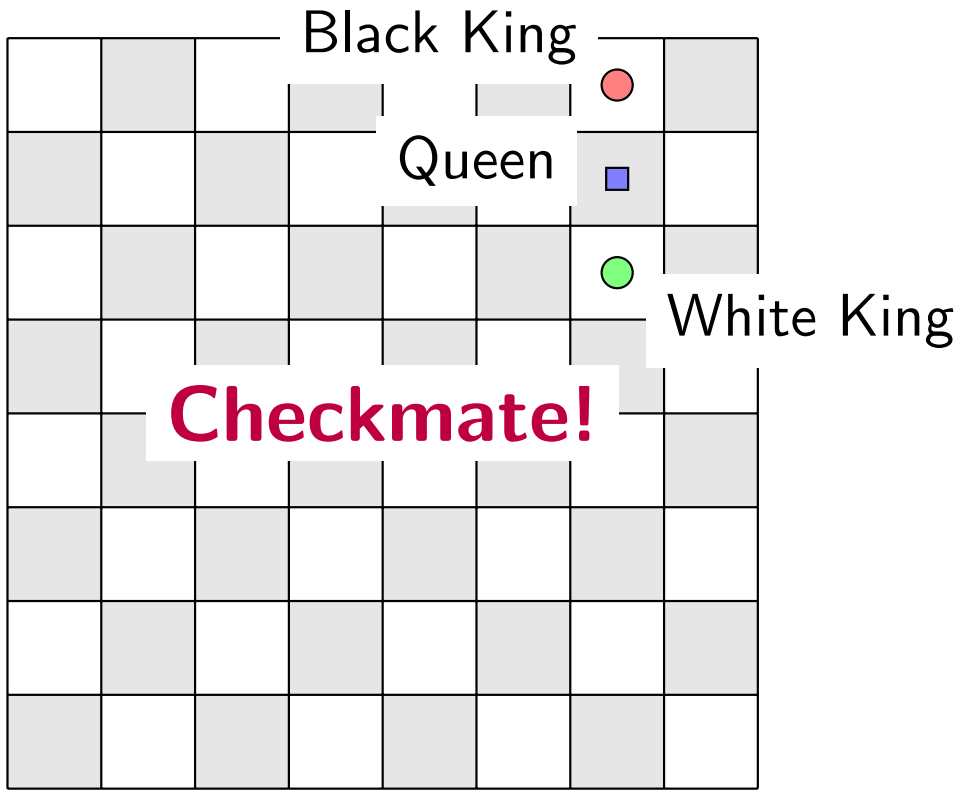
# Endgame Tablebase Approximation



# Endgame Tablebase Approximation



# Endgame Tablebase Approximation



# How else will solve a game?

- Minimax Algorithm

# How else will solve a game?

- Minimax Algorithm
- Alpha-Beta Pruning

# How else will solve a game?

- Minimax Algorithm
- Alpha-Beta Pruning
- Can we change the structure of graph?

# How else will solve a game?

- Minimax Algorithm
- Alpha-Beta Pruning
- Can we change the structure of graph?
- Neural Networks

# Thank You...