# Contents



# 1 Learning Goals

By the end of the exercise, you should be able to

- Formulate a real-world problem as a constraint satisfaction problem by defining variables, domains, and constraints.
- Trace the execution of the backtracking search algorithm with the full AC-3 arc consistency algorithm on the 4-queens problem.
- Trace the execution of the backtracking search algorithm with forward checking on the 4-queens problem.

# 2 The 4-Queens Problem

The 4-queens problem consists of a 4x4 chessboard with 4 queens. The goal is to place the 4 queens on the chessboard such that no two queens can attack each other. Each queen attacks anything in the same row, in the same column, or in the same diagonal.

## 2.1 The CSP formulation

Formulate the state of the 4-queens problem below.

- Assume that exactly one queen is in each column. Given this, we only need to keep track of the row position of each queen.
- Variables: $x_0, x_1, x_2, x_3$ where $x_i$ is the row position of the queen in column $i$, where $i \in \{0, 1, 2, 3\}$.
- Domains: $dom(x_i) = \{0, 1, 2, 3\}$ for all $x_i$.
- Constraints: No pair of queens are in the same row or diagonal.

  $(\forall i(\forall j((i \neq j) \rightarrow ((x_i \neq x_j) \wedge (|x_i - x_j| \neq |i - j|)))))$

  All the constraints are explicitly given below.

  $((x_0 \neq x_1) \wedge (|x_0 - x_1| \neq 1) \wedge (x_0 \neq x_2) \wedge (|x_0 - x_2| \neq 2) \wedge (x_0 \neq x_3) \wedge (|x_0 - x_3| \neq 3) \wedge (x_1 \neq x_2) \wedge (|x_1 - x_2| \neq 1) \wedge (x_1 \neq x_3) \wedge (|x_1 - x_3| \neq 2) \wedge (x_2 \neq x_3) \wedge (|x_2 - x_3| \neq 1))$

Formulate the 4-queens problem as a CSP below.

- State: one queen per column in the leftmost $k$ columns with no pair of queens attacking each other.
- Initial state: no queens on the board.
- Goal state: 4 queens on the board. No pair of queens are attacking each other.
- Successor function: add a queen to the leftmost empty column such that it is not attacked by any other existing queen.

## 2.2 The AC-3 Arc Consistency Algorithm

**Algorithm 1** Revise($X_i, C$)

revised ← false<br>
**for** $x$ in $dom(X_i)$ **do**<br>
    **if** $\neg\exists\, y \in dom(X_j)$ s.t. $(x, y)$ satisfies the constraint $C$ **then**<br>
        remove $x$ from $dom(X_i)$<br>
        revised ← true<br>
    **end if**<br>
**end for**<br>
**return** revised

**Algorithm 2** The AC-3 Algorithm

Put $(v, C)$ in the set $S$ for every variable $v$ and every constraint involving $v$.<br>
**while** $S$ is not empty **do**<br>
    remove $(X_i, C_{ij})$ from $S$ ($C_{ij}$ is a constraint between $X_i$ and $X_j$.)<br>
    **if** Revise($X_i, C_{ij}$) **then**<br>
        **if** $dom(X_i)$ is empty **then return** false<br>
        **for** $X_k$ where $C_{ki}$ is a constraint between $X_k$ and $X_i$ **do**<br>
            add $(X_k, C_{ki})$ to $S$<br>
        **end for**<br>
    **end if**<br>
**end while**<br>
**return** true

## 2.3 Backtracking Search with Arc Consistency

**Algorithm 3** BACKTRACK-INFERENCES(assignment, csp)

**if** assignment is complete **then return** true<br>
var ← SELECT-UNASSIGNED-VARIABLE(csp)<br>
**for all** value in ORDER-DOMAIN-VALUES(var, assignment, csp) **do**<br>
    **if** adding {var = value} satisfies every constraint **then**<br>
        add {var = value} to assignment<br>
        inf-result ← INFERENCES(assignment, csp)<br>
        **if** inf-result is true **then**<br>
            add the inference results to assignment<br>
            result ← BACKTRACK(assignment, csp)<br>
            **if** result is true **then return** result<br>
        **end if**<br>
    **end if**<br>
    remove {var = value} and the inference results from assignment<br>
**end for**<br>
**return** false

# 3 Practice Questions

## 3.1 Arc Consistency with an Initial Assignment

Start with an initial assignment of $x_0 = 0$ for the 4-queens problem. Let's execute the AC-3 algorithm.

The starting domains and assignment:

$x_0 = 0$, $dom(x_1) \in \{0, 1, 2, 3\}$, $dom(x_2) \in \{0, 1, 2, 3\}$, and $dom(x_3) \in \{0, 1, 2, 3\}$

The set of variable-constraint pairs:

$(x_0, x_0 \neq x_1)$, $(x_1, x_0 \neq x_1)$, $(x_0, x_0 \neq x_2)$, $(x_2, x_0 \neq x_2)$, $(x_0, x_0 \neq x_3)$, $(x_3, x_0 \neq x_3)$, $(x_1, x_1 \neq x_2)$, $(x_2, x_1 \neq x_2)$, $(x_1, x_1 \neq x_3)$, $(x_3, x_1 \neq x_3)$, $(x_2, x_2 \neq x_3)$, $(x_3, x_2 \neq x_3)$, $(x_0, |x_0 - x_1| \neq 1)$, $(x_1, |x_0 - x_1| \neq 1)$, $(x_0, |x_0 - x_2| \neq 2)$, $(x_2, |x_0 - x_2| \neq 2)$, $(x_0, |x_0 - x_3| \neq 3)$, $(x_3, |x_0 - x_3| \neq 3)$, $(x_1, |x_1 - x_2| \neq 1)$, $(x_2, |x_1 - x_2| \neq 1)$, $(x_1, |x_1 - x_3| \neq 2)$, $(x_3, |x_1 - x_3| \neq 2)$, $(x_2, |x_2 - x_3| \neq 1)$, $(x_3, |x_2 - x_3| \neq 1)$.

Note that every constraint appears in exactly two pairs, one for each variable in the constraint.

Below, we will write out the details of a few steps. Executing the AC-3 algorithm from start to finish will take roughly 24 steps. I encourage you to trace through all the execution steps on your own.

Answer the three questions below.

**Question 1:**

Let the starting domains and assignment be

$x_0 = 0$, $dom(x_1) \in \{0, 1, 2, 3\}$, $dom(x_2) \in \{0, 1, 2, 3\}$, and $dom(x_3) \in \{0, 1, 2, 3\}$

Suppose that we remove the pair $(x_0, x_0 \neq x_1)$ from the set.

Describe any change to the domain of the variable.

Describe any variable-constraint pairs that we need to add back to the set.

**Question 2:**

Let the starting domains and assignment be

$x_0 = 0$, $dom(x_1) \in \{0, 1, 2, 3\}$, $dom(x_2) \in \{0, 1, 2, 3\}$, and $dom(x_3) \in \{0, 1, 2, 3\}$

Suppose that we remove the pair $(x_1, x_0 \neq x_1)$ from the set.

Describe any change to the domain of the variable.

Describe any variable-constraint pairs that we need to add back to the set.

**Question 3:**

Let the starting domains and assignment be

$x_0 = 0$, $dom(x_1) \in \{2, 3\}$, $dom(x_2) \in \{1, 3\}$, and $dom(x_3) \in \{1, 2\}$

Suppose that we remove the pair $(x_3, |x_2 - x_3| \neq 1)$ from the set.

Describe any change to the domain of the variable.

Describe any variable-constraint pairs that we need to add back to the set.

## 3.2 Backtracking Search with Forward Checking

Start with an initial assignment of $x_0 = 0$ for the 4-queens problem. Execute the backtracking search algorithm with forward checking until a solution is reached.

The starting domains and assignment:

$x_0 = 0$, $dom(x_1) = \{0, 1, 2, 3\}$, $dom(x_2) = \{0, 1, 2, 3\}$, and $dom(x_3) = \{0, 1, 2, 3\}$

Choose variables and values using the following conventions.

- When choosing which variable to assign value to, always choose the left most unassigned variable.
- When choosing which value to assign to a variable, always choose the top unassigned value.

Show the steps of backtracking search with forward checking below.

1. Assign $x_0 = 0$.

   Starting domains and assignment: $x_0 = 0$, $dom(x_1) = \{0, 1, 2, 3\}$, $dom(x_2) = \{0, 1, 2, 3\}$, and $dom(x_3) = \{0, 1, 2, 3\}$

   Forward checking:

   Updated domains and assignment: