Introduction to Artificial Intelligence

English practicals 3: Constraint satisfaction

Marika Ivanová

Department of Theoretical Computer Science and Mathematical Logic (KTIML)
Faculty of Mathematics and Physics

March 1st 2022

Approaches for solving algorithmic problems

Specialized algorithms

- Dijkstra's shortest path algorithm
- Hungarian algorithm for the assignment problem
- Prim's algorithm for minimum spanning tree

Is there an algorithm for every problem?

Approaches for solving algorithmic problems

Specialized algorithms

- Dijkstra's shortest path algorithm
- Hungarian algorithm for the assignment problem
- Prim's algorithm for minimum spanning tree

Is there an algorithm for every problem?

- Infinitely many but countable algorithms

Approaches for solving algorithmic problems

Specialized algorithms

- Dijkstra's shortest path algorithm
- Hungarian algorithm for the assignment problem
- Prim's algorithm for minimum spanning tree

Is there an algorithm for every problem?

- Infinitely many but countable algorithms
- Uncountable decision problems

Approaches for solving algorithmic problems

Specialized algorithms

- Dijkstra's shortest path algorithm
- Hungarian algorithm for the assignment problem
- Prim's algorithm for minimum spanning tree

Is there an algorithm for every problem?

- Infinitely many but countable algorithms
- Uncountable decision problems
- An algorithm does not exist for most of the problems

Approaches for solving algorithmic problems

Specialized algorithms

- Dijkstra's shortest path algorithm
- Hungarian algorithm for the assignment problem
- Prim's algorithm for minimum spanning tree

Is there an algorithm for every problem?

- Infinitely many but countable algorithms
- Uncountable decision problems
- An algorithm does not exist for most of the problems
- There is no algorithm deciding whether a given program stops (The Halting problem)

Approaches for solving algorithmic problems

General approaches for solving problems

- What class of problems to solve?
- How to encode problems of this class?
- Find a general algorithm for solving this class of problems

Approaches for solving algorithmic problems

General approaches for solving problems

- What class of problems to solve?
- How to encode problems of this class?
- Find a general algorithm for solving this class of problems

Examples of problem classes

- (Integer) linear programming (ILP)
- Constraint satisfaction programming (CSP)
- SAT (logical formulae in CNF)

A small reminder

- Problem solving is realized via **search**
- **Tree** search vs **graph** search
- **Uninformed** search - DFS, BFS, Uniform cost search vs **Informed** search - Best first search, A*
- Constraint satisfaction problem (CSP) consists of

A small reminder

- Problem solving is realized via **search**
- **Tree** search vs **graph** search
- **Uninformed** search - DFS, BFS, Uniform cost search vs **Informed** search - Best first search, A*
- Constraint satisfaction problem (CSP) consists of
  - A finite set of **variables** - feature of the world state (position of a queen, number in sudoku tile)

A small reminder

- Problem solving is realized via **search**
- **Tree** search vs **graph** search
- **Uninformed** search - DFS, BFS, Uniform cost search vs **Informed** search - Best first search, A*
- Constraint satisfaction problem (CSP) consists of
  - A finite set of **variables** - feature of the world state (position of a queen, number in sudoku tile)
  - **Domains** - a finite set of values for each variable.

A small reminder

- Problem solving is realized via **search**
- **Tree** search vs **graph** search
- **Uninformed** search - DFS, BFS, Uniform cost search vs **Informed** search - Best first search, A*
- Constraint satisfaction problem (CSP) consists of
  - A finite set of **variables** - feature of the world state (position of a queen, number in sudoku tile)
  - **Domains** - a finite set of values for each variable.
  - A finite set of **constraints** - relation over a subset of variables ($X \leq Y$)

A small reminder

- Problem solving is realized via **search**
- **Tree** search vs **graph** search
- **Uninformed** search - DFS, BFS, Uniform cost search vs **Informed** search - Best first search, A*
- Constraint satisfaction problem (CSP) consists of
  - A finite set of **variables** - feature of the world state (position of a queen, number in sudoku tile)
  - **Domains** - a finite set of values for each variable.
  - A finite set of **constraints** - relation over a subset of variables ($X \leq Y$)

  - A **feasible solution** to a CSP is a complete consistent assignment of values to variables

A small reminder

- Problem solving is realized via **search**
- **Tree** search vs **graph** search
- **Uninformed** search - DFS, BFS, Uniform cost search vs **Informed** search - Best first search, A*
- Constraint satisfaction problem (CSP) consists of
  - A finite set of **variables** - feature of the world state (position of a queen, number in sudoku tile)
  - **Domains** - a finite set of values for each variable.
  - A finite set of **constraints** - relation over a subset of variables ($X \leq Y$)

  - A **feasible solution** to a CSP is a complete consistent assignment of values to variables

  - The arc (V_i, V_j) is arc consistent iff for each value x from the domain D_i there exists a value y in the domain D_j such that the assignment $V_i = x$ and $V_j = y$ satisfies all the binary constraints on V_i, V_j.

Selected quiz questions

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search
- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search

- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

  Yes, both values in each variable have a consistent counterpart

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search
- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

  Yes, both values in each variable have a consistent counterpart
- If arc (X,Y) is consistent, does it mean that arc (Y,X) is also consistent?

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search
- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

  Yes, both values in each variable have a consistent counterpart
- If arc (X,Y) is consistent, does it mean that arc (Y,X) is also consistent?

  No. Consider $D_X = \{1\}$ and $D_Y = \{1, 2\}$

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search
- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

  Yes, both values in each variable have a consistent counterpart
- If arc (X,Y) is consistent, does it mean that arc (Y,X) is also consistent?

  No. Consider $D_X = \{1\}$ and $D_Y = \{1, 2\}$
- TRUE/FALSE: If the problem is not arc consistent it has no solution.

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search
- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

  Yes, both values in each variable have a consistent counterpart
- If arc (X,Y) is consistent, does it mean that arc (Y,X) is also consistent?

  No. Consider $D_X = \{1\}$ and $D_Y = \{1, 2\}$
- TRUE/FALSE: If the problem is not arc consistent it has no solution.

  False

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search
- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

  Yes, both values in each variable have a consistent counterpart
- If arc (X,Y) is consistent, does it mean that arc (Y,X) is also consistent?

  No. Consider $D_X = \{1\}$ and $D_Y = \{1, 2\}$
- TRUE/FALSE: If the problem is not arc consistent it has no solution.

  False
- TRUE/FALSE: If the problem is arc consistent it has a solution.

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

  DFS (backtracking) as tree search
- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

  Yes, both values in each variable have a consistent counterpart
- If arc (X,Y) is consistent, does it mean that arc (Y,X) is also consistent?

  No. Consider $D_X = \{1\}$ and $D_Y = \{1, 2\}$
- TRUE/FALSE: If the problem is not arc consistent it has no solution.

  False
- TRUE/FALSE: If the problem is arc consistent it has a solution.

  False

Selected quiz questions

- What is the best uninformed search algorithm to solve the N-queens problem? Why? (POLL!)

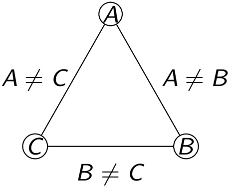
  DFS (backtracking) as tree search
- Is constraint $X \neq Y$, where domains of X and Y are 1,2, arc consistent?

  Yes, both values in each variable have a consistent counterpart
- If arc (X,Y) is consistent, does it mean that arc (Y,X) is also consistent?

  No. Consider $D_X = \{1\}$ and $D_Y = \{1, 2\}$
- TRUE/FALSE: If the problem is not arc consistent it has no solution.

  False
- TRUE/FALSE: If the problem is arc consistent it has a solution.

  False



$D_A = \{1, 2\}$

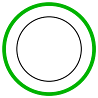
$D_B = \{1, 2\}$

$D_C = \{1, 2\}$

Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$

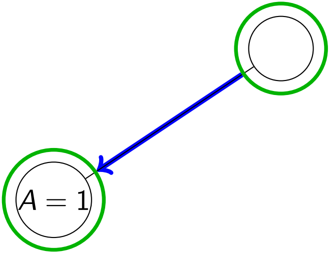
Example: DFS



- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$

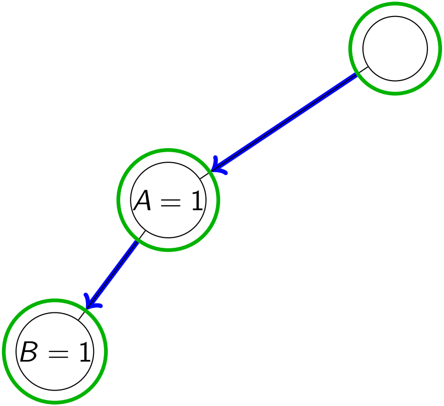
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



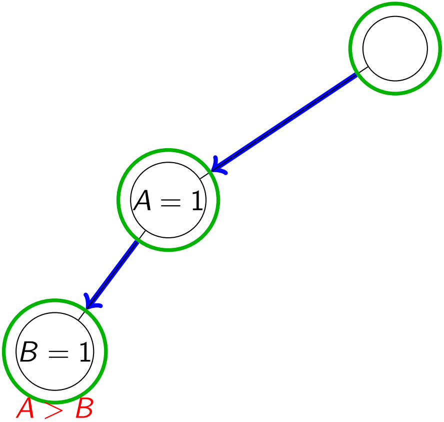
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



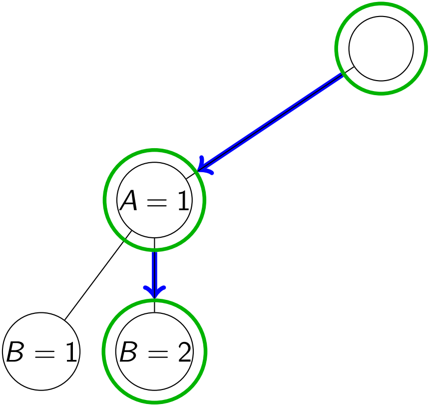
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



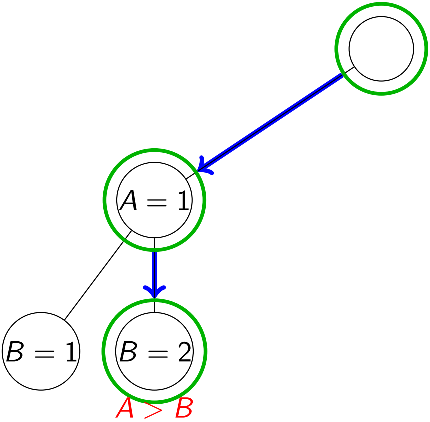
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



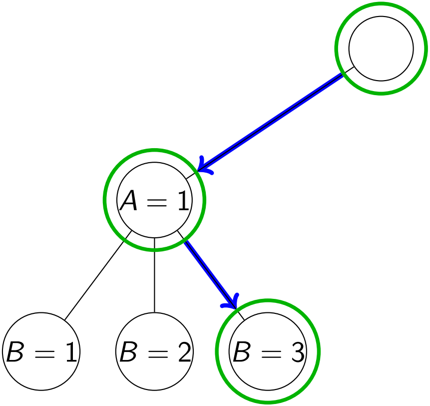
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



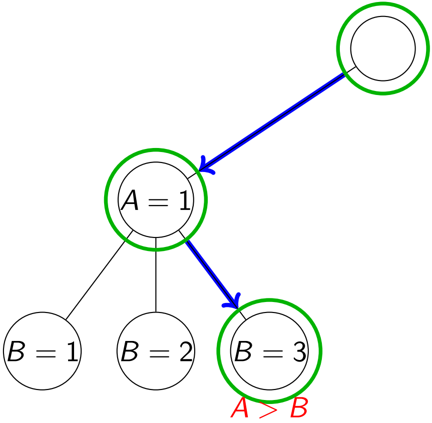
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



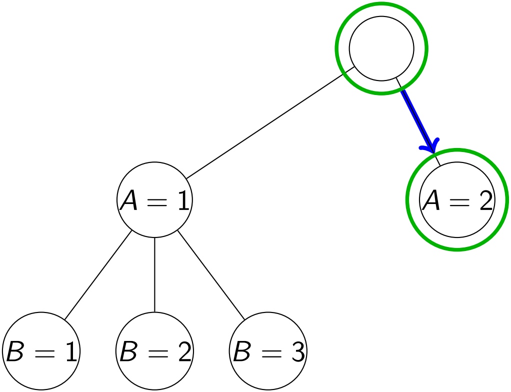
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



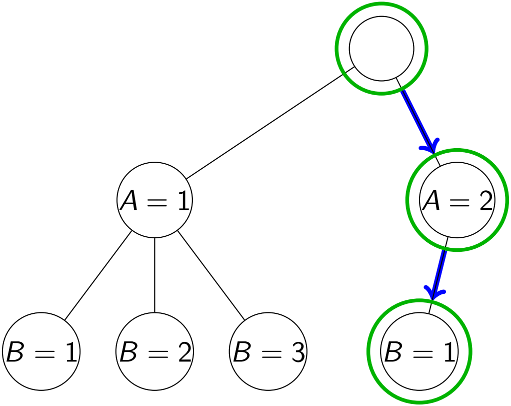
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



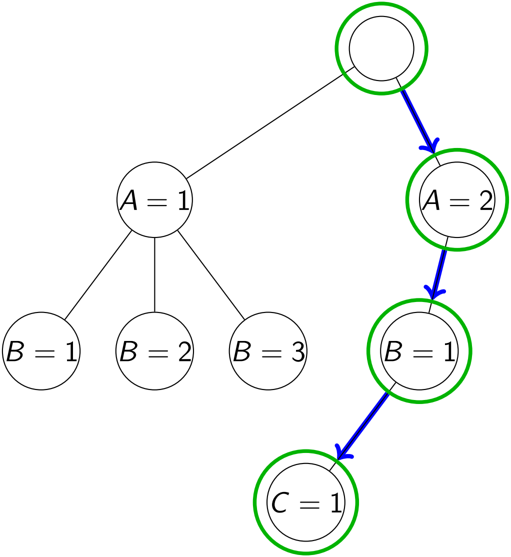
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



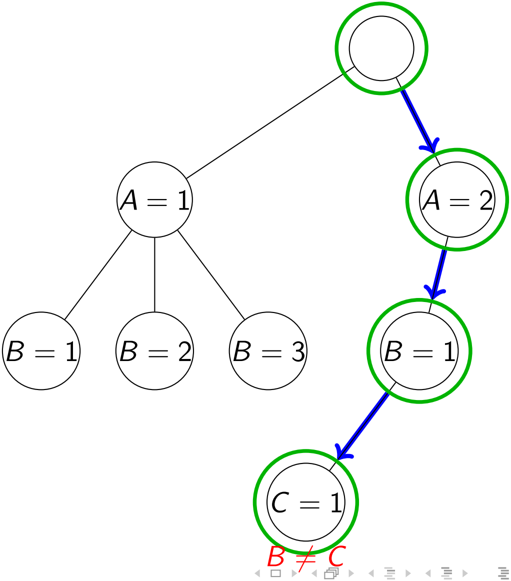
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



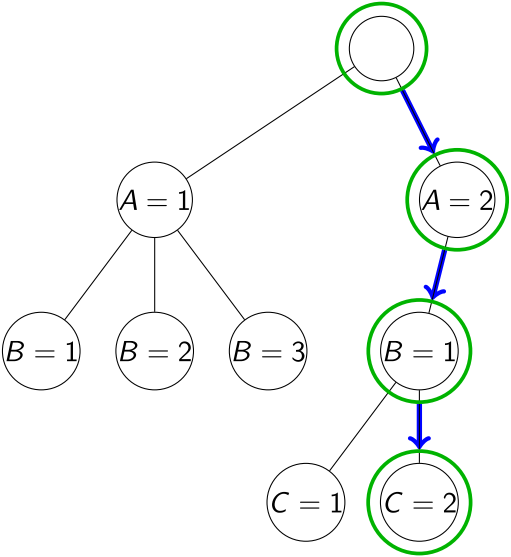
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



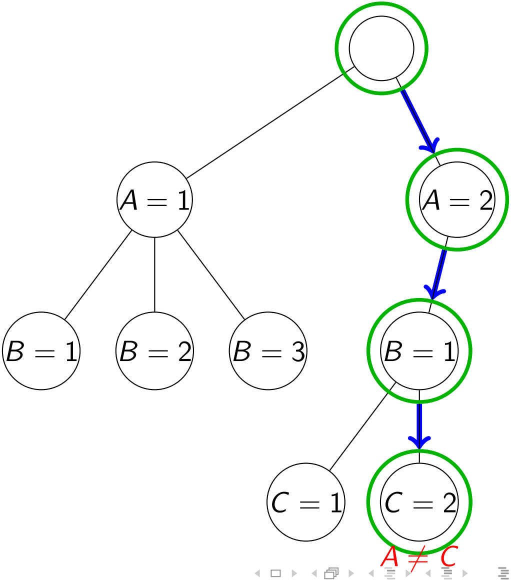
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



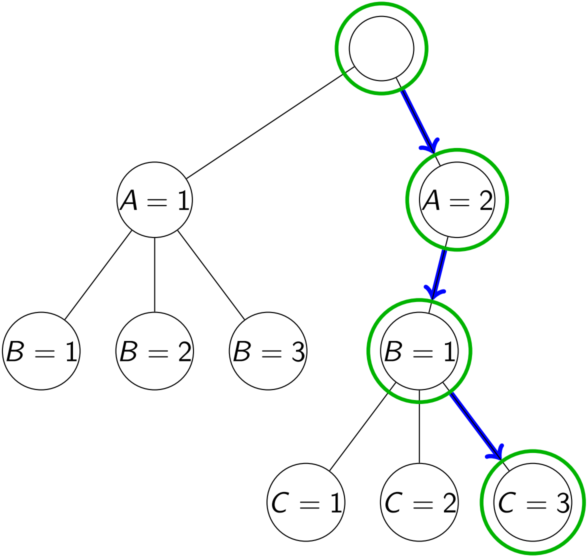
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



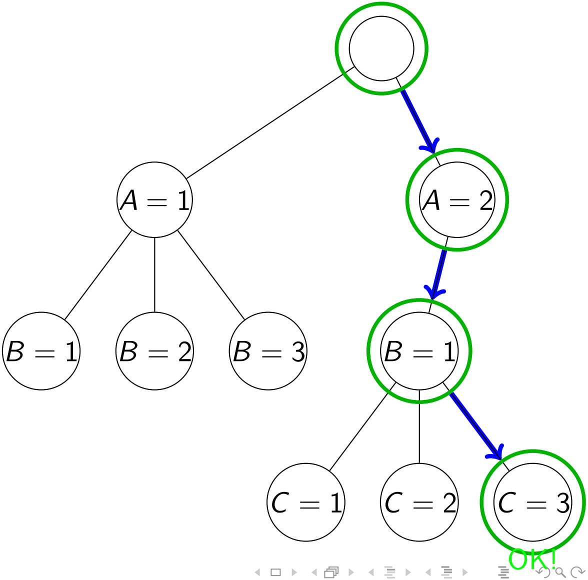
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



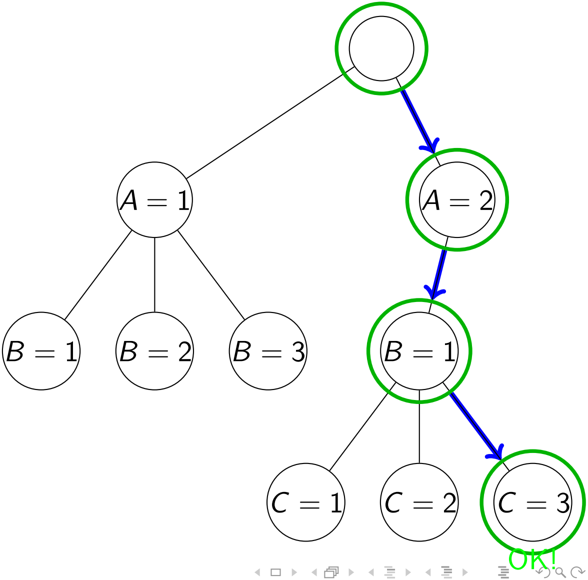
Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$



Example: DFS

- Variables: $\{A, B, C\}$
- Domains: $D_A = D_B = D_C = \{1, 2, 3\}$
- Constraints:
  - $A > B$
  - $B \neq C$
  - $A \neq C$
- Feasible solution:
  $A = 2$
  $B = 1$
  $C = 3$



Example: forward checking

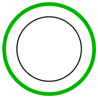
- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, 2, 3\}$

  $D_C = \{1, 2, 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$

Example: forward checking



- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, 2, 3\}$

  $D_C = \{1, 2, 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$

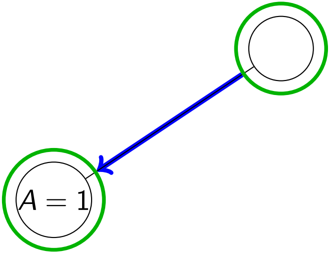
Example: forward checking

- Variables: $\{A, B, C\}$
- Domains:

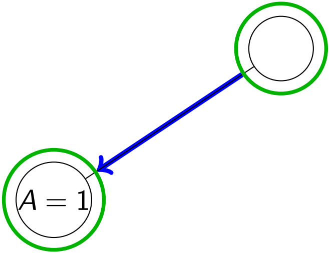
  $D_A = \{1, 2, 3\}$

  $D_B = \{1, 2, 3\}$

  $D_C = \{1, 2, 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$



Example: forward checking



- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{\ ,\ ,\ \}$

  $D_C = \{\ , 2, 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$

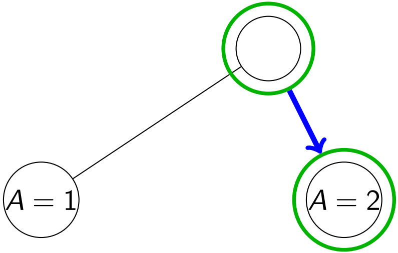
Example: forward checking

- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, 2, 3\}$

  $D_C = \{1, 2, 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$



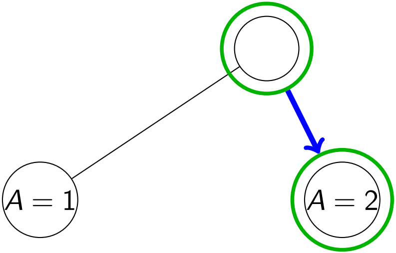
Example: forward checking

- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, \quad , \quad \}$

  $D_C = \{1, \quad , 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$



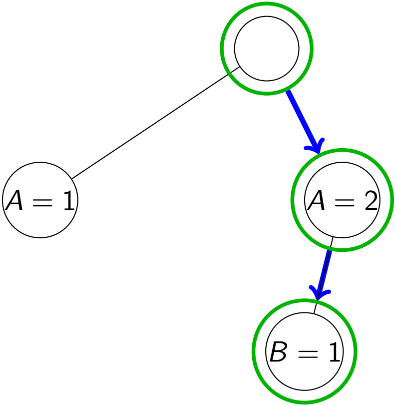
Example: forward checking

- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, \quad , \quad \}$

  $D_C = \{1, \quad , 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$



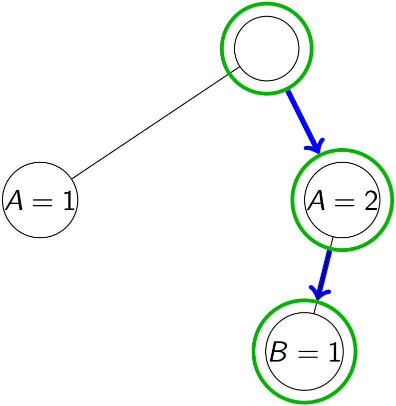
Example: forward checking

- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, \ , \ \}$

  $D_C = \{\ , \ , 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$



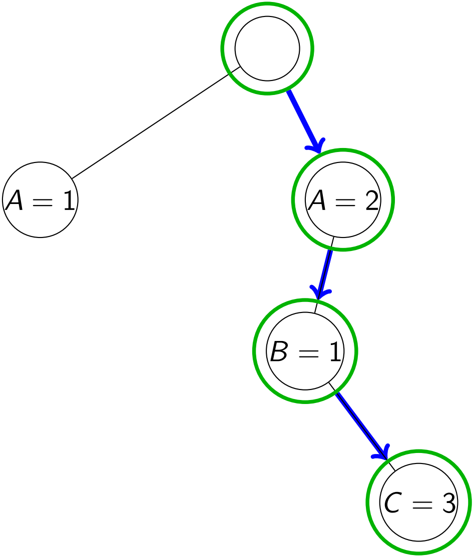
Example: forward checking

- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, \ , \ \}$

  $D_C = \{ \ , \ , 3\}$
- Constraints:
- $A > B$
- $B \neq C$
- $A \neq C$



AC-3

- Turn each binary constraint into two arcs
  ($A < B$ becomes $A < B$ and $B > A$)
- Add all arcs to queue Q
- Repeat until Q is empty:
  - remove an arc (X_i, X_j) from Q and check if for every value of X_i there is a value in X_j
  - remove any inconsistent values from X_i
  - if the domain of X_i changed, add all arcs (X_k, X_i) to Q

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, 2, 3\}$

  $D_C = \{1, 2, 3\}$
- Constraints:

  $A > B$

  $B = C$

Q

Arcs:

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{1, 2, 3\}$

  $D_B = \{1, 2, 3\}$

  $D_C = \{1, 2, 3\}$
- Constraints:

  $A > B$

  $B = C$

Q

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:
  $D_A = \{1, 2, 3\}$
  $D_B = \{1, 2, 3\}$
  $D_C = \{1, 2, 3\}$
- Constraints:
  $A > B$
  $B = C$

Q

$A > B \leftarrow$

$B < A$

$B = C$

$C = B$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2, 3\}$

  $D_C = \{1, 2, 3\}$
- Constraints:

  $A > B$

  $B = C$

Q

$A > B \leftarrow$

$B < A$

$B = C$

$C = B$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:
  $D_A = \{\ \ , 2, 3\}$
  $D_B = \{1, 2, 3\}$
  $D_C = \{1, 2, 3\}$
- Constraints:
  $A > B$
  $B = C$

Q

$B < A \leftarrow$

$B = C$

$C = B$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2, \ \ \}$

  $D_C = \{1, 2, 3\}$
- Constraints:

  $A > B$

  $B = C$

Q

$B < A \leftarrow$

$B = C$

$C = B$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2,\ \ \}$

  $D_C = \{1, 2, 3\}$
- Constraints:

  $A > B$

  $B = C$

Q

$B < A \leftarrow$

$B = C$

$C = B$

$A > B$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2, \ \ \}$

  $D_C = \{1, 2, 3\}$
- Constraints:

  $A > B$

  $B = C$

Q

$B = C \ \leftarrow$

$C = B$

$A > B$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2,\ \ \}$

  $D_C = \{1, 2, 3\}$
- Constraints:

  $A > B$

  $B = C$

Q

$C = B \leftarrow$

$A > B$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2,\ \ \}$

  $D_C = \{1, 2,\ \ \}$
- Constraints:

  $A > B$

  $B = C$

Q

$C = B \leftarrow$

$A > B$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:
  $D_A = \{\ \ , 2, 3\}$
  $D_B = \{1, 2,\ \ \}$
  $D_C = \{1, 2,\ \ \}$
- Constraints:
  $A > B$
  $B = C$

Q

$C = B \leftarrow$

$A > B$

$B = C$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2,\ \ \}$

  $D_C = \{1, 2,\ \ \}$
- Constraints:

  $A > B$

  $B = C$

Q

$A > B \leftarrow$

$B = C$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2,\ \ \}$

  $D_C = \{1, 2,\ \ \}$
- Constraints:

  $A > B$

  $B = C$

Q

$B = C \ \leftarrow$

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2, \ \ \}$

  $D_C = \{1, 2, \ \ \}$
- Constraints:

  $A > B$

  $B = C$

Q

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Example: AC-3

- Variables: $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2, \ \ \}$

  $D_C = \{1, 2, \ \ \}$
- Constraints:

  $A > B$

  $B = C$

Q

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Empty Q - the problem is arc consistent.

Example: AC-3

- Variables:
  $\{A, B, C\}$
- Domains:

  $D_A = \{\ \ , 2, 3\}$

  $D_B = \{1, 2,\ \ \}$

  $D_C = \{1, 2,\ \ \}$
- Constraints:

  $A > B$

  $B = C$

Q

Arcs:

$A > B$

$B < A$

$B = C$

$C = B$

Will the algorithm always terminate?

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount
- The required profit is at least 30 dollars

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount
- The required profit is at least 30 dollars
- What should be placed in the knapsack?

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount
- The required profit is at least 30 dollars
- What should be placed in the knapsack?

- Variables: $Goods = \{W, P, C\}$ (amounts of types of goods)

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount
- The required profit is at least 30 dollars
- What should be placed in the knapsack?

- Variables: $Goods = \{W, P, C\}$ (amounts of types of goods)
- Domains: $D_i = \{0, \ldots, 4\}$, $i = w, p, c$

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount
- The required profit is at least 30 dollars
- What should be placed in the knapsack?

- Variables: $Goods = \{W, P, C\}$ (amounts of types of goods)
- Domains: $D_i = \{0, \ldots, 4\}$, $i = w, p, c$
- Constraints:

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount
- The required profit is at least 30 dollars
- What should be placed in the knapsack?

- Variables: $Goods = \{W, P, C\}$ (amounts of types of goods)
- Domains: $D_i = \{0, \ldots, 4\}$, $i = w, p, c$
- Constraints:

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

- whisky (takes 4 capacity units, profit is 15 dollars),
- perfume (3 units, 10 dollars)
- cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount
- The required profit is at least 30 dollars
- What should be placed in the knapsack?

- Variables: $Goods = \{W, P, C\}$ (amounts of types of goods)
- Domains: $D_i = \{0, \ldots, 4\}$, $i = w, p, c$
- Constraints:
  - $4W + 3P + 2C \leq 9$

Modelling example: knapsack

Knapsack

The smuggler has a knapsack of capacity 9 units. It can be filled with:

  - whisky (takes 4 capacity units, profit is 15 dollars),
  - perfume (3 units, 10 dollars)
  - cigarettes (2 units, 7 dollars)

- Any combination of the goods can be used in any amount
- The required profit is at least 30 dollars
- What should be placed in the knapsack?

- Variables: $Goods = \{W, P, C\}$ (amounts of types of goods)
- Domains: $D_i = \{0, \ldots, 4\}$, $i = w, p, c$
- Constraints:
  - $4W + 3P + 2C \leq 9$
  - $15W + 10P + 7C \geq 30$

Modelling example: sudoku

formulate (model) the sudoku problem as a CSP

Modelling example: sudoku

formulate (model) the sudoku problem as a CSP

- Variables: $X = \{x_{ij} : 1 \leq i,j \leq 9\}$
- Domains: $D_{ij} = \{1,\ldots,9\}$, $1 \leq i,j \leq 9$

Modelling example: sudoku

formulate (model) the sudoku problem as a CSP

- Variables: $X = \{x_{ij} : 1 \leq i, j \leq 9\}$
- Domains: $D_{ij} = \{1, \ldots, 9\}$, $1 \leq i, j \leq 9$
- Variable subsets: $\{C_1, \ldots, C_9, R_1, \ldots, R_9, B_{1,1}, B_{1,2}, \ldots, B_{3,3}\}$ defined by

  $\forall x_{ij} \in X : x_{ij} \in C_j, x_{ij} \in R_i, x_{ij} \in B_{(i-1)/3+1,(j-1)/3+1}$

Modelling example: sudoku

formulate (model) the sudoku problem as a CSP

- Variables: $X = \{x_{ij} : 1 \leq i, j \leq 9\}$
- Domains: $D_{ij} = \{1, \dots, 9\}$, $1 \leq i, j \leq 9$
- Variable subsets: $\{C_1, \dots, C_9, R_1, \dots, R_9, B_{1,1}, B_{1,2}, \dots, B_{3,3}\}$

  defined by

  $\forall x_{ij} \in X : x_{ij} \in C_j, x_{ij} \in R_i, x_{ij} \in B_{(i-1)/3+1,(j-1)/3+1}$
- Constraints:

Modelling example: sudoku

formulate (model) the sudoku problem as a CSP

- Variables: $X = \{x_{ij} : 1 \leq i, j \leq 9\}$
- Domains: $D_{ij} = \{1, \ldots, 9\}$, $1 \leq i, j \leq 9$
- Variable subsets: $\{C_1, \ldots, C_9, R_1, \ldots, R_9, B_{1,1}, B_{1,2}, \ldots, B_{3,3}\}$ defined by

  $\forall x_{ij} \in X : x_{ij} \in C_j, x_{ij} \in R_i, x_{ij} \in B_{(i-1)/3+1,(j-1)/3+1}$
- Constraints:
  - $\forall j \in \{1, \ldots, 9\}$: all_different(C_j)
  - $\forall i \in \{1, \ldots, 9\}$: all_different(R_i)
  - $\forall k, k' \in \{1, \ldots, 3\}$: all_different($B_{kk'}$)

Modelling example: sudoku

formulate (model) the sudoku problem as a CSP

- Variables: $X = \{x_{ij} : 1 \leq i, j \leq 9\}$
- Domains: $D_{ij} = \{1, \ldots, 9\}$, $1 \leq i, j \leq 9$
- Variable subsets: $\{C_1, \ldots, C_9, R_1, \ldots, R_9, B_{1,1}, B_{1,2}, \ldots, B_{3,3}\}$

  defined by

  $\forall x_{ij} \in X : x_{ij} \in C_j, x_{ij} \in R_i, x_{ij} \in B_{(i-1)/3+1,(j-1)/3+1}$
- Constraints:
  - $\forall j \in \{1, \ldots, 9\}$: all_different(C_j)
  - $\forall i \in \{1, \ldots, 9\}$: all_different(R_i)
  - $\forall k, k' \in \{1, \ldots, 3\}$: all_different($B_{kk'}$)
  - for each clue $h_{ij} = v$ add $x_{ij} = v$

Modelling example: sudoku

formulate (model) the sudoku problem as a CSP

- Variables: $X = \{x_{ij} : 1 \leq i, j \leq 9\}$
- Domains: $D_{ij} = \{1, \ldots, 9\}$, $1 \leq i, j \leq 9$
- Variable subsets: $\{C_1, \ldots, C_9, R_1, \ldots, R_9, B_{1,1}, B_{1,2}, \ldots, B_{3,3}\}$ defined by

  $\forall x_{ij} \in X : x_{ij} \in C_j, x_{ij} \in R_i, x_{ij} \in B_{(i-1)/3+1,(j-1)/3+1}$
- Constraints:
  - $\forall j \in \{1, \ldots, 9\}$: all_different(C_j)
  - $\forall i \in \{1, \ldots, 9\}$: all_different(R_i)
  - $\forall k, k' \in \{1, \ldots, 3\}$: all_different($B_{kk'}$)
  - for each clue $h_{ij} = v$ add $x_{ij} = v$
- Each variable evaluation satisfying all constraints corresponds to a solution to a given sudoku instance and vice versa

Modelling example: Perfect matching

Perfect matching

A <u>perfect matching</u> in a graph $G = (V, E)$ is a subset $M \subseteq E$ in which every vertex in V is incident to exactly one edge in M.

Formulate (model) perfect matching as a CSP

Modelling example: Perfect matching

Perfect matching

A perfect matching in a graph $G = (V, E)$ is a subset $M \subseteq E$ in which every vertex in V is incident to exactly one edge in M.

Formulate (model) perfect matching as a CSP

- Variables: $\{x_f \in \{0, 1\} : f \in E\}$

Modelling example: Perfect matching

Perfect matching

A perfect matching in a graph $G = (V, E)$ is a subset $M \subseteq E$ in which every vertex in V is incident to exactly one edge in M.

Formulate (model) perfect matching as a CSP

- Variables: $\{x_f \in \{0, 1\} : f \in E\}$
- Constraints:
  - for every $u \in V$ of degree k $(uv_1, \ldots, uv_k)$: $(x_{uv_1}, \ldots, x_{uv_k}) \in \{e_i : i \in \{1, \ldots, k\}\}$

Modelling example: Perfect matching

Perfect matching

A perfect matching in a graph $G = (V, E)$ is a subset $M \subseteq E$ in which every vertex in V is incident to exactly one edge in M.

Formulate (model) perfect matching as a CSP

- Variables: $\{x_f \in \{0, 1\} : f \in E\}$
- Constraints:
  - for every $u \in V$ of degree k $(uv_1, \ldots, uv_k)$: $(x_{uv_1}, \ldots, x_{uv_k}) \in \{e_i : i \in \{1, \ldots, k\}\}$
  - common notation: $e_1 = (1, 0, 0, 0)$, $e_3 = (0, 0, 1, 0)$ for $k = 4$

Modelling example: Perfect matching

Perfect matching

A perfect matching in a graph $G = (V, E)$ is a subset $M \subseteq E$ in which every vertex in V is incident to exactly one edge in M.

Formulate (model) perfect matching as a CSP

- Variables: $\{x_f \in \{0, 1\} : f \in E\}$
- Constraints:
  - for every $u \in V$ of degree k $(uv_1, \dots, uv_k)$: $(x_{uv_1}, \dots, x_{uv_k}) \in \{e_i : i \in \{1, \dots, k\}\}$
  - common notation: $e_1 = (1, 0, 0, 0)$, $e_3 = (0, 0, 1, 0)$ for $k = 4$

Formulation for bipartite graphs

Modelling example: Perfect matching

Perfect matching

A perfect matching in a graph $G = (V, E)$ is a subset $M \subseteq E$ in which every vertex in V is incident to exactly one edge in M.

Formulate (model) perfect matching as a CSP

- Variables: $\{x_f \in \{0, 1\} : f \in E\}$
- Constraints:
  - for every $u \in V$ of degree k $(uv_1, \ldots, uv_k)$:
    $(x_{uv_1}, \ldots, x_{uv_k}) \in \{e_i : i \in \{1, \ldots, k\}\}$
  - common notation: $e_1 = (1, 0, 0, 0)$, $e_3 = (0, 0, 1, 0)$ for $k = 4$

Formulation for bipartite graphs

- Bipartite graph with node sets A and B with $|A| = |B|$

Modelling example: Perfect matching

Perfect matching

A perfect matching in a graph $G = (V, E)$ is a subset $M \subseteq E$ in which every vertex in V is incident to exactly one edge in M.

Formulate (model) perfect matching as a CSP

- Variables: $\{x_f \in \{0, 1\} : f \in E\}$
- Constraints:
  - for every $u \in V$ of degree k $(uv_1, \ldots, uv_k)$: $(x_{uv_1}, \ldots, x_{uv_k}) \in \{e_i : i \in \{1, \ldots, k\}\}$
  - common notation: $e_1 = (1, 0, 0, 0)$, $e_3 = (0, 0, 1, 0)$ for $k = 4$

Formulation for bipartite graphs

- Bipartite graph with node sets A and B with $|A| = |B|$
- Variables: $x_a \in N(a)$ denotes the node from B matched with a

  $N(a) \subseteq B$ are nodes adjacent to a

Modelling example: Perfect matching

Perfect matching

A perfect matching in a graph $G = (V, E)$ is a subset $M \subseteq E$ in which every vertex in V is incident to exactly one edge in M.

Formulate (model) perfect matching as a CSP

- Variables: $\{x_f \in \{0, 1\} : f \in E\}$
- Constraints:
  - for every $u \in V$ of degree k $(uv_1, \ldots, uv_k)$: $(x_{uv_1}, \ldots, x_{uv_k}) \in \{e_i : i \in \{1, \ldots, k\}\}$
  - common notation: $e_1 = (1, 0, 0, 0)$, $e_3 = (0, 0, 1, 0)$ for $k = 4$

Formulation for bipartite graphs

- Bipartite graph with node sets A and B with $|A| = |B|$
- Variables: $x_a \in N(a)$ denotes the node from B matched with a

  $N(a) \subseteq B$ are nodes adjacent to a
- Constraints: $\forall a, a' \in A$ such that $a \neq a'$: $x_a \neq x'_a$