FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE
Exercise 3: CSP – Problems Winter Semester 2022/23
Sebastian Mair

1 Presented Problems

Problem 3.1: Turning n-ary constraints into binary constraints

(from *Russell & Norvig 3ed.* q. 7.6) Suppose that we have $CSP = (X, D, E^1)$ with

$$\begin{aligned} X &= \{A, B, C\}, \\ D &= \{\text{dom}(A), \text{dom}(B), \text{dom}(C)\}, \\ E &= \{\langle (A, B, C),\ A + B = C\rangle\}, \end{aligned}$$

where $\text{dom}(A)$, $\text{dom}(B)$, and $\text{dom}(C)$ denote the domain of variable A, B, and C, respectively, and each domain can be $\{0, 1, \ldots, 9\}$ for example.

Problem 3.1.1: Draw the constraint hypergraph for the CSP. In this case, a hypergraph is a graph with two types of nodes. The first type of node represents the *variables*, depicted by ◯, and the second type of node represents the constraint, depicted by □. Based on the number of variables involved, what is the type of the constraint?

Problem 3.1.2: We can eliminate the higher-order constraint in E by replacing the constraint node □ with a new variable node Z. (We denote this new CSP as CSP'.) What is the domain for variable Z? (Hint: The domain for variable Z can be ordered pairs of other values.) What is the new constraint set E' after introducing the new variable Z?

Problem 3.1.3: Modify CSP' such that it only contains binary constraints and formally express the new $CSP'' = (X'', D'', E'')$.

Problem 3.1.4: Taking inspiration from previous solutions, how can you generally turn a n-ary constraint into binary constraints?

Problem 3.2: Arc consistency and backtracking search for binary constraints

Consider the constraint satisfaction problem in Fig. 1. According to the picture, we have $CSP = (X, D, E)$ with

$$\begin{aligned} X &= v_1, v_2, v_3, v_4, v_5, \\ D &= \{\text{dom}(v_1), \text{dom}(v_2), \text{dom}(v_3), \text{dom}(v_4), \text{dom}(v_5)\}, \\ E &= \{\langle (v_1, v_2),\ v_2 = v_1 + 1\rangle, \\ &\quad \langle (v_1, v_3),\ v_1 \neq v_3\rangle \\ &\quad \langle (v_2, v_3),\ v_2 \neq v_3\rangle, \\ &\quad \langle (v_3, v_4),\ v_3 \neq v_4\rangle, \\ &\quad \langle (v_3, v_5),\ v_3 \neq v_5\rangle, \\ &\quad \langle (v_4, v_5),\ v_4 \neq v_5\rangle, \\ &\quad \langle (v_1, v_5),\ v_1 \neq v_5\rangle\}, \end{aligned}$$

where $\text{dom}(v_1)$, $\text{dom}(v_2)$, $\text{dom}(v_3)$, $\text{dom}(v_4)$ and $\text{dom}(v_5)$ denote the domain of variable v_1, v_2, v_3, v_4 and v_5, respectively, and each domain is initially $\{2, 3, 4\}$. Note that all constraints in the graph are binary constraints.

Problem 3.2.1: Sort the variables once by their domain size (i.e. number of remaining values) and once by their degree (i.e. number of constraints on other unassigned variables).

[1] the symbol E is taken from German word *Einschränkung*.

Problem 3.1: Turning n-ary constraints into binary constraints

(from *Russell & Norvig 3ed.* q. 7.6) Suppose that we have $CSP = (X, D, E)$ with

$$\begin{aligned} X &= \{A, B, C\}, \\ D &= \{\text{dom}(A), \text{dom}(B), \text{dom}(C)\}, \\ E &= \{\langle (A, B, C),\ A + B = C \rangle\}, \end{aligned}$$

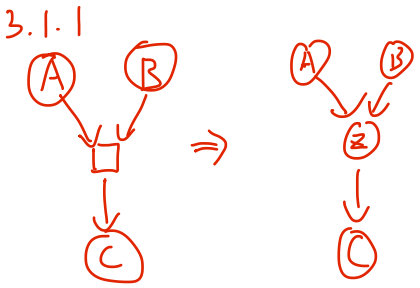
where $\text{dom}(A)$, $\text{dom}(B)$, and $\text{dom}(C)$ denote the domain of variable A, B, and C, respectively, and each domain can be $\{0, 1, \ldots, 9\}$ for example.

Problem 3.1.1: Draw the constraint hypergraph for the CSP. In this case, a hypergraph is a graph with two types of nodes. The first type of node represents the *variables*, depicted by ○, and the second type of node represents the constraint, depicted by □. Based on the number of variables involved, what is the type of the constraint?

Problem 3.1.2: We can eliminate the higher-order constraint in E by replacing the constraint node □ with a new variable node Z. (We denote this new CSP as CSP'.) What is the domain for variable Z? (Hint: The domain for variable Z can be ordered pairs of other values.) What is the new constraint set E' after introducing the new variable Z?

Problem 3.1.3: Modify CSP' such that it only contains binary constraints and formally express the new $CSP'' = (X'', D'', E'')$.

Problem 3.1.4: Taking inspiration from previous solutions, how can you generally turn a n-ary constraint into binary constraints?



3.1.2

$dom(Z) = \{(z_1, z_2, z_3) \mid z_1 \in dom(A) \wedge z_2 \in dom(B) \wedge z_3 \in dom(C)\}$

$E' = \{z_1 = A,\ z_2 = B,\ z_3 = C\} \cup E$

middle node

3.1.3

$X'' = \{A, B, C, Z\}$

$D'' = \{dom(A), dom(B), dom(C), dom(Z)\}$

$dom(Z) = \{(z_1, z_2, z_3) \mid z_1 \in dom(A) \wedge z_2 \in dom(B) \wedge z_3 \in dom(C) \wedge z_3 = z_1 + z_2\}$

$E'' = \{z_1 = A,\ z_2 = B,\ z_1 = C\}$

Problem 3.2: Arc consistency and backtracking search for binary constraints

Consider the constraint satisfaction problem in Fig. 1. According to the picture, we have $CSP = (X, D, E)$ with

$$\begin{aligned} X &= v_1, v_2, v_3, v_4, v_5, \\ D &= \{\text{dom}(v_1), \text{dom}(v_2), \text{dom}(v_3), \text{dom}(v_4), \text{dom}(v_5)\}, \\ E &= \{\langle (v_1, v_2),\ v_2 = v_1 + 1 \rangle, \\ &\quad \langle (v_1, v_3),\ v_1 \neq v_3 \rangle \\ &\quad \langle (v_2, v_3),\ v_2 \neq v_3 \rangle, \\ &\quad \langle (v_3, v_4),\ v_3 \neq v_4 \rangle, \\ &\quad \langle (v_3, v_5),\ v_3 \neq v_5 \rangle, \\ &\quad \langle (v_4, v_5),\ v_4 \neq v_5 \rangle, \\ &\quad \langle (v_1, v_5),\ v_1 \neq v_5 \rangle\}, \end{aligned}$$

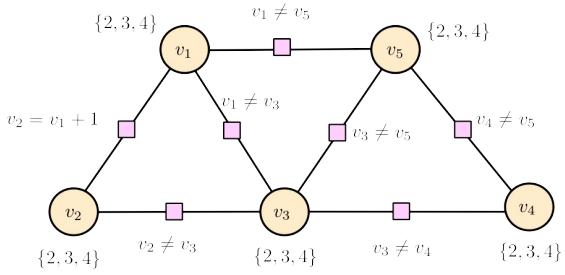


Figure 1: Constraint graph for Problem 3.2

where $\text{dom}(v_1)$, $\text{dom}(v_2)$, $\text{dom}(v_3)$, $\text{dom}(v_4)$ and $\text{dom}(v_5)$ denote the domain of variable v_1, v_2, v_3, v_4 and v_5, respectively, and each domain is initially $\{2, 3, 4\}$. Note that all constraints in the graph are binary constraints.

Problem 3.2.1: Sort the variables once by their domain size (i.e. number of remaining values) and once by their degree (i.e. number of constraints on other unassigned variables).

domain size	
3	v_1 v_2 v_3 v_4 v_5

degree	
4	v_3
3	v_1 v_5
2	v_2 v_4

Problem 3.2.2: Perform backtracking search by hand to solve the CSP problem: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose the variable with the smaller lower index. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, set the value to 3; if this is not possible choose the lowest value. After each assignment, perform forward checking as inference. Backtrack if you find an inconsistency.

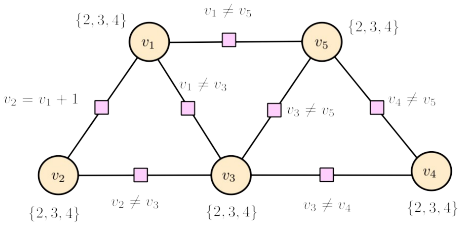
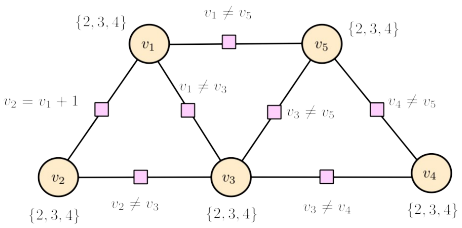


Figure 1: Constraint graph for Problem 3.2

MRV → degree heuristics → ↓ index
LCV → 3 → ↓ value

step	assign	current domains v_1	v_2	v_3	v_4	v_5	degree v_1	v_2	v_3	v_4	v_5	backtrack to
0	$\emptyset$	234	234	234	234	234	3	2	4	2	3	
1	$V_3=3$	24	24	—	24	24	2	1	—	1	2	
2	$V_1=2$	—	$\emptyset$	—	24	4	—	0	—	0	2	2
2	$V_1=4$	—	$\emptyset$	—	24	2	—	0	—	0	2	1
1	$V_3=2$	34	34	—	34	34	2	1	—	1	2	
2	$V_1=3$	—	4	—	34	4	—	0	—	1	1	
3	$V_5=4$	—	4	—	3	—	—	0	—	0	—	
4	$V_2=4$	—	—	—	3	—	—	—	—	0	—	
5	$V_4=3$	—	—	—	—	—	—	—	—	—	—	

Problem 3.2.3: Perform backtracking search again, but with a different inference: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose the variable with the smaller lower index. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, set the value to 3; if this is not possible choose the lowest value. After each assignment, perform the arc consistency algorithm. Backtrack if you find an inconsistency.



MRV → degree heuristics → ↓ index
LCV → 3 → ↓ value

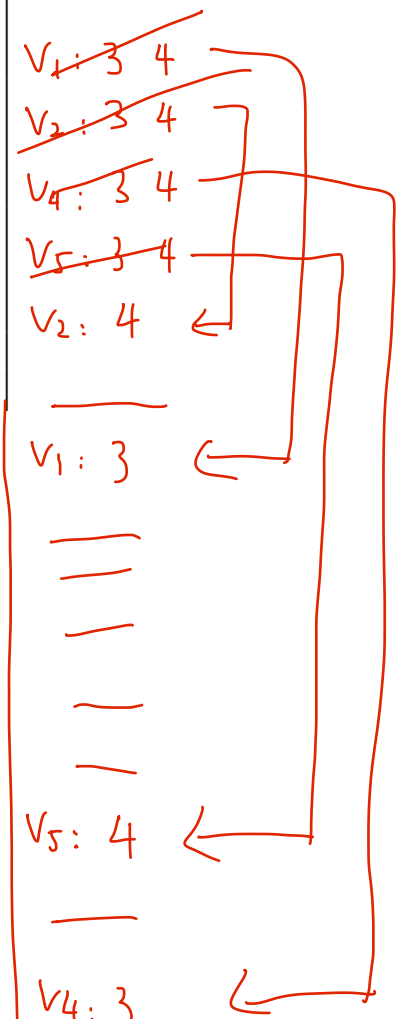
step	assign	current domains v_1	v_2	v_3	v_4	v_5	degree v_1	v_2	v_3	v_4	v_5	backtrack to
0	$\emptyset$	234	234	234	234	234	3	2	4	2	3	
1	$V_3=3$	24	$\emptyset$	—	24	24	2	1	—	1	2	1
1	$V_3=2$	3	4	—	3	4	2	1	—	1	2	
2	$V_1=3$	—	4	—	3	4	—	0	—	1	1	
3	$V_4=3$	—	4	—	—	4	—	0	—	—	0	
4	$V_2=4$	—	—	—	—	4	—	—	—	—	0	
5	$V_5=4$	—	—	—	—	—	—	—	—	—	—	

① $V_3 = 3$

Arc	Queue	Domain Change
	(V_1,V_3) (V_2,V_3) (V_4,V_3) (V_5,V_3)	$V_1: 2\ 4$
(V_1,V_3)	(V_2,V_1) (V_5,V_1) (V_1,V_2) (V_5,V_4)	$V_2: 2\ 4$
(V_2,V_3)	(V_1,V_5) (V_4,V_5)	$V_4: 2\ 4$
(V_4,V_3)		$V_5: 2\ 4$
(V_5,V_3)		$V_2: \emptyset$
(V_2,V_1)		

② $V_3 = 2$

Arc	Queue	Domain Change
	(V_1,V_3) (V_2,V_3) (V_4,V_3) (V_5,V_3)	
(V_1,V_3)	(V_2,V_1) (V_5,V_1)	$V_1: 3\ 4$
(V_2,V_3)	(V_1,V_2)	$V_2: 3\ 4$
(V_4,V_3)	(V_5,V_4)	$V_4: 3\ 4$
(V_5,V_3)	(V_1,V_5) (V_4,V_5)	$V_5: 3\ 4$
(V_2,V_1)	(V_3,V_2)	$V_2: 4$
(V_5,V_1)	—	—
(V_1,V_2)	(V_3,V_1) (V_5,V_1)	$V_1: 3$
(V_5,V_4)	—	—
(V_1,V_5)	—	—
(V_4,V_5)	—	—
(V_3,V_2)	—	—
(V_3,V_1)	—	—
(V_5,V_1)	(V_3,V_5) (V_4,V_5)	$V_5: 4$
(V_2,V_5)	—	—
(V_4,V_5)	(V_3,V_4)	$V_4: 3$
(V_3,V_4)	—	—
	$\emptyset$	



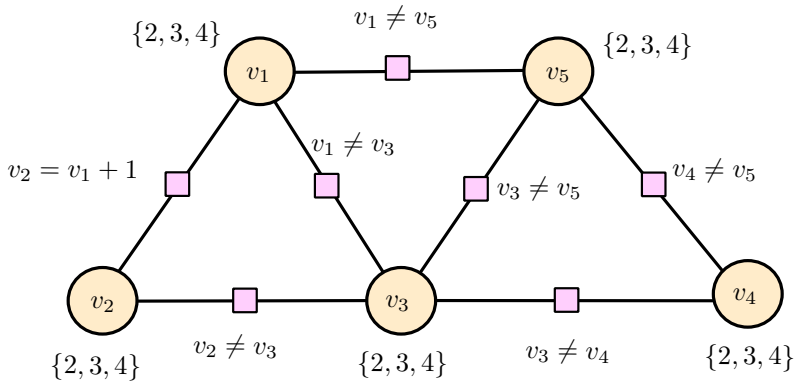


Figure 1: Constraint graph for Problem 3.2

Problem 3.2.2: Perform backtracking search by hand to solve the CSP problem: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose the variable with the smaller lower index. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, set the value to 3; if this is not possible choose the lowest value. After each assignment, perform forward checking as inference. Backtrack if you find an inconsistency.

Problem 3.2.3: Perform backtracking search again, but with a different inference: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose the variable with the smaller lower index. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, set the value to 3; if this is not possible choose the lowest value. After each assignment, perform the arc consistency algorithm. Backtrack if you find an inconsistency.

Problem 3.2.4: Consider the CSP in Fig. 1 at its initial state. Is the CSP arc consistent? Is this a convenient initial condition if we plan to apply backtracking search? Apply the arc consistency algorithm to the CSP as a preprocessing step: Initialize the queue with all arcs of the CSP. Is the CSP arc consistent now?

Problem 3.2.5: Perform backtracking search after the preprocessing step of the previous task: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose one randomly. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, choose one randomly. After each assignment, perform arc consistency algorithm. Backtrack if you find an inconsistency. Assume the data structure of the queue is a set, i.e., if we add an element to the queue which is already in the queue, the element will not be added a second time (each element is unique).

Problem 3.2.6: For each of the previous performances of backtracking search with a different inference (forward checking, arc consistency, arc consistency after preprocessing), compare the number of iterations and the number of times you needed to backtrack.

2 Additional Problems

Problem 3.3: Arc consistency and backtracking search for binary constraints

Consider again the constraint satisfaction problem from Problem 3.2 described in Fig. 1.

Problem 3.3.1: Perform backtracking search by hand with forward checking as inference after preprocessing with the arc consistency algorithm: Apply the arc consistency algorithm to the CSP initializing the queue with all arcs, then start backtracking search. Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose one randomly. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, choose randomly. After each assignment, perform forward checking as inference. Backtrack if you find an inconsistency.

Problem 3.2.4: Consider the CSP in Fig. 1 at its initial state. Is the CSP arc consistent? Is this a convenient initial condition if we plan to apply backtracking search? Apply the arc consistency algorithm to the CSP as a preprocessing step: Initialize the queue with all arcs of the CSP. Is the CSP arc consistent now?

⇒ No, not arc consistent

$V_1: \{2,3\}$ $V_2: \{3,4\}$

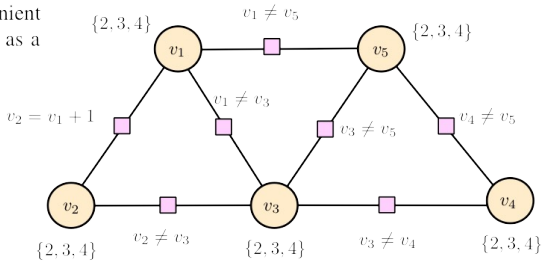


Figure 1: Constraint graph for Problem 3.2

Problem 3.2.5: Perform backtracking search after the preprocessing step of the previous task: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose one randomly. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, choose one randomly. After each assignment, perform arc consistency algorithm. Backtrack if you find an inconsistency. Assume the data structure of the queue is a set, i.e., if we add an element to the queue which is already in the queue, the element will not be added a second time (each element is unique).

MRV → degree heuristic → R

LCV → R

		current domains					degree					*backtrack to*
step	assign	v_1	v_2	v_3	v_4	v_5	v_1	v_2	v_3	v_4	v_5	
0	$\emptyset$	2 3	3 4	234	234	234	3	2	4	2	3	
1	$V_1 = 2$	~	3	4	2	3	–	1	3	2	2	
2	$V_3 = 4$	—	3	—	2	3	–	0	–	1	1	
3	$V_4 = 2$	—	3	—	–	3	–	0	–	–	0	
4	$V_2 = 3$	–	—	–	–	3	–	–	–	–	0	
5	$V_5 = 3$	—	—	–	–	–	–	–	–	–	–	

$V_1 = 2$

Arc	Queue	Domain Change
	~~(V_2,V_1)~~ ~~(V_3,V_1)~~ ~~(V_5,V_1)~~	
(V_2,V_1)	~~(V_1,V_2)~~	$V_2: 3$
(V_3,V_1)	~~(V_2,V_3)~~ ~~(V_4,V_3)~~ ~~(V_5,V_3)~~	~~$V_3: 3\,4$~~
(V_5,V_1)	~~(V_3,V_5)~~ ~~(V_4,V_5)~~	~~$V_5: 3\,4$~~
(V_3,V_2)	~~(V_1,V_3)~~ ~~(V_4,V_3)~~ ~~(V_5,V_3)~~	$V_3: 4$
(V_2,V_3)	—	—
(V_4,V_3)	~~(V_3,V_4)~~	~~$V_4: 2\,3$~~
(V_5,V_3)	~~(V_1,V_5)~~ ~~(V_4,V_5)~~	$V_5: 3$
(V_3,V_5)	—	—
(V_4,V_5)	~~(V_3,V_4)~~	$V_4: 2$
(V_1,V_3)	—	—
(V_4,V_3)	—	—
(V_5,V_3)	—	—
(V_5,V_4)	—	—
(V_1,V_5)	—	—
(V_4,V_5)	—	—
(V_3,V_4)	— $\emptyset$	—

Problem 3.3.1: Perform backtracking search by hand with forward checking as inference after preprocessing with the arc consistency algorithm: Apply the arc consistency algorithm to the CSP initializing the queue with all arcs, then start backtracking search. Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose one randomly. Use least-constraining-value heuristics to decide which value to assign; if there is a tie, choose randomly. After each assignment, perform forward checking as inference. Backtrack if you find an inconsistency.

MRV → degree heuristic → R

LCV → R

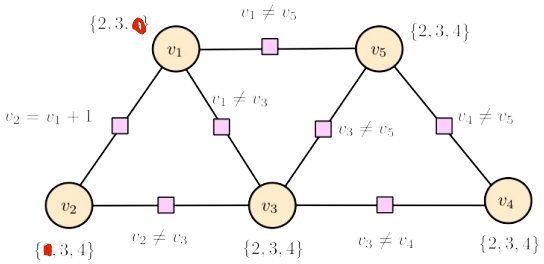
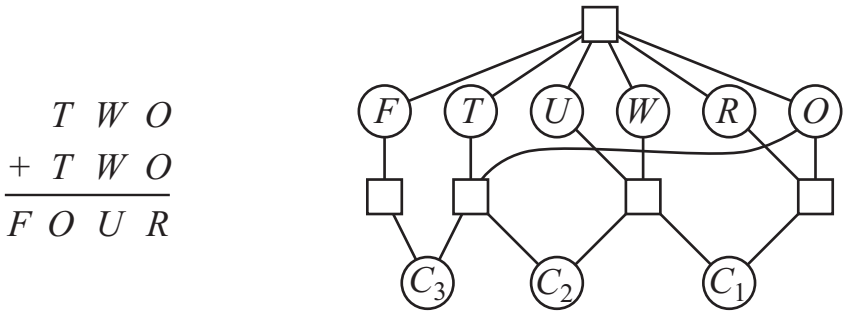


Figure 1: Constraint graph for Problem 3.2

		current domains					degree					*backtrack to*
step	assign	v_1	v_2	v_3	v_4	v_5	v_1	v_2	v_3	v_4	v_5	
0	$\emptyset$	23	34	234	234	234	3	2	4	2	3	
1	$v_1 = 2$	–	4	34	234	34	–	1	3	2	2	
2	$v_2 = 4$	–	–	3	234	34	–	–	2	2	2	
3	$v_3 = 3$	–	–	–	24	4	–	–	–	1	1	
4	$v_5 = 4$	–	–	–	2	–	–	–	–	0	–	
5	$v_4 = 2$	–	–	–	–	–	–	–	–	–	–	

Problem 3.4: Solving a CSP by hand performing backtracking search with minimum-remaining-values (MRV) and degree heuristics, least-constraining-value heuristics, and forward checking

(from *Russell & Norvig 3ed.* q. 7.5) Suppose that we have the cryptarithmetic problem as shown in Fig. 2.



(a) (b)

Figure 2: (a) A cryptarithmetic problem. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct. (b) The constraint hypergraph for the cryptarithmetic problem, showing the `Alldiff` constraint (square box at the top) as well as the column addition constraints (four square boxes in the middle). The variables $C1$, $C2$, and $C3$ represent the carry digits for the three columns.

We model the cryptarithmetic problem as $CSP = (X, D, C)$ with

$$\begin{aligned} X &= \{F, T, U, W, R, O, C_1, C_2, C_3\} \\ D &= \{\text{numbers}, \ldots, \text{numbers}, \text{binary}, \text{binary}, \text{binary}\} \\ C &= \{\langle (O, R, C_1), \quad O + O = R + 10 \cdot C_1 \rangle, \\ &\quad \langle (U, W, C_1, C_2), \quad C_1 + W + W = U + 10 \cdot C_2 \rangle, \\ &\quad \langle (O, T, C_2, C_3), \quad C_2 + T + T = O + 10 \cdot C_3 \rangle, \\ &\quad \langle (C_3, F), \quad C_3 = F \rangle, \\ &\quad \langle (F, T, U, W, R, O), \quad \texttt{Alldiff}(F, T, U, W, R, O) \rangle\}, \end{aligned}$$

where numbers $= \{0, 1, 2, \ldots, 9\}$ and binary $= \{0, 1\}$.

Problem 3.4.1: Replace all boxes which correspond to higher-order constraints by binary constraints. Use the approach of Problem 3.1 and introduce variables such as X_1, X_2, and X_3, etc.

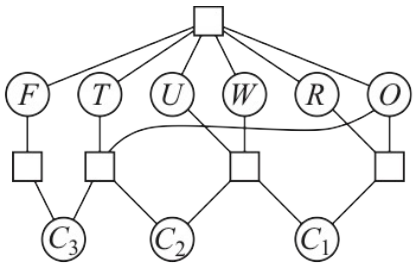
Problem 3.4.2: Sort the variables once by their domain size (i.e. number of remaining values) and once by their degree (i.e. number of constraints on other unassigned variables).

Problem 3.4.3: Perform backtracking search to solve the cryptarithmetic problem: Determine which variable to expand next by applying the minimum-remaining-values (MRV) heuristic; if there is a tie, use degree heuristics; if there is a tie again, choose one randomly. Use least-constraining-value heuristics to decide which value to assign. After each assignment, perform forward checking. Backtrack if you find an inconsistency.

We model the cryptarithmetic problem as $CSP = (X, D, C)$ with

$$
\begin{array}{rcl}
X & = & \{F, T, U, W, R, O, C_1, C_2, C_3\} \\
D & = & \{\text{numbers}, \ldots, \text{numbers}, \text{binary}, \text{binary}, \text{binary}\} \\
C & = & \{\langle (O, R, C_1), \quad O + O = R + 10 \cdot C_1 \rangle, \\
 & & \langle (U, W, C_1, C_2), \quad C_1 + W + W = U + 10 \cdot C_2 \rangle, \\
 & & \langle (O, T, C_2, C_3), \quad C_2 + T + T = O + 10 \cdot C_3 \rangle, \\
 & & \langle (C_3, F), \quad C_3 = F \rangle, \\
 & & \langle (F, T, U, W, R, O), \quad \texttt{Alldiff}(F, T, U, W, R, O) \rangle \}.
\end{array}
$$

$$
\begin{array}{r}
\ T\ W\ O \\
+\ T\ W\ O \\
\hline
F\ O\ U\ R
\end{array}
$$



(a) (b)

where numbers $= \{0, 1, 2, \ldots, 9\}$ and binary $= \{0, 1\}$.

Problem 3.4.1: Replace all boxes which correspond to higher-order constraints by binary constraints. Use the approach of Problem 3.1 and introduce variables such as X_1, X_2, and X_3, etc.

$X_1 = \{(O, R, C_1) \mid O \in \text{numbers} \wedge R \in \text{numbers} \wedge C_1 \in \text{binary} \wedge O+O = R+10 \cdot C_1\}$

$X_2 = \{(U, W, C_1, C_2) \mid U \in \text{numbers} \wedge W \in \text{numbers} \wedge C_1 \in \text{binary} \wedge C_2 \in \text{binary} \wedge C_1+W+W = U+10 \cdot C_2\}$

$X_3 = \{(O, T, C_2, C_3) \mid O \in \text{numbers} \wedge T \in \text{numbers} \wedge C_2 \in \text{binary} \wedge C_3 \in \text{binary} \wedge C_2+T+T = O+10 \cdot C_3\}$

$X_4 = \{(F, T, U, W, R, O) \mid F, T, U, W, R, O \in \text{numbers} \wedge \text{Alldiff}(F, T, U, W, R, O)\}$

$C = \{\langle (O, X_1), fst(X_1) = O \rangle, \langle (R, X_1), snd(X_1) = R \rangle, \ldots\}$

Problem 3.4.2: Sort the variables once by their domain size (i.e. number of remaining values) and once by their degree (i.e. number of constraints on other unassigned variables).

domain size	
10	F T U W R O
2	C_1 C_2 C_3
10	X_1
20	X_2 X_3
$p(10,6)$	X_4

degree	
6	X_4
4	X_2 X_3
3	X_1
3	O
2	I = T U R W
2	C_1 C_2 C_3