Chapter 2

Constraint Networks

In this chapter we begin formally modeling constraint satisfaction problems as constraint networks. The initial formal work on constraint networks introduced by Montanari [214] was restricted to binary constraints, defined on pairs of variables only. Much of the early research, experiments in particular, were limited to the binary case. Indeed, it is possible to show that any set of constraints can be mapped to the binary case. Nevertheless, we will always assume the general case, where constraints are defined on sets of variables of arbitrary size, and will explicitly refer to the special case of binary constraints when appropriate.

2.1 Constraint networks and constraint satisfaction

2.1.1 The basics of the framework

A *constraint network* $\mathcal{R}$ consists of a finite set of *variables* $X = \{x_1, \ldots, x_n\}$, with respective *domains* $D = \{D_1, \ldots, D_n\}$ (which list the possible values for each variable $D_i = \{v_1, \ldots, v_k\}$), and a set of *constraints* $C = \{C_1, \ldots, C_t\}$. Thus, a constraint network can be viewed as a triple (X, D, C).

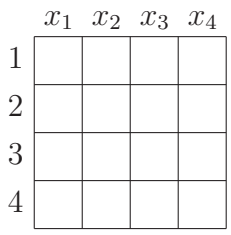
A constraint C_i is a relation R_i defined on a subset of variables S_i, $S_i \subseteq X$. The relation denotes the variables' simultaneous legal value assignments. S_i is called the *scope* of R_i. If $S_i = \{x_{i_1}, ...x_{i_r}\}$, then R_i is a subset of the Cartesian product $D_{i_1} \times ... \times D_{i_r}$. Thus, a constraint can also be viewed as a pair $C_i =< S_i, R_i >$. When the scopes identity is clear, we will often identify the constraint C_i with its relation R_i. Alternatively, for clarity, a constraint relation may be denoted R_{S_i}. A scope will be denoted explicitly as $\{x, y, z\}$, or, if there is no possibility of confusion, as xyz. For example, we can denote as R_{xyz}, (or $< xyz, R >$) the constraint defined over the variables x, y and z whose relation is R. The set of scopes $S = \{S_1, \ldots, S_i, \ldots, S_t\}$ is called the network *scheme*. Without loss

of generality, we assume for relational constraints that only a single constraint is defined over a subset S_i in S. Namely, if $i \neq j$ then $S_i \neq S_j$. This assumption may be removed when discussing algebraic or Boolean constraints.

The *arity* of a constraint refers to the cardinality, or size, of its scope. A *unary constraint* is defined on a single variable, a *binary constraint* on two variables. A *binary constraint network* has only unary and binary constraints.

Formulating the n-queens problem

Lets return to the n-queens problem and explore a possible constraint network formulation. We can think of the columns of the chess board as the variables, $x_1, \ldots, x_n$, and the possible row positions, $D_i = \{1, \ldots, n\}$, as domains of the variables. Assigning a value $j \in D_i$ to a variable x_i means to place a queen in row j on column x_i of the board. The constraints, binary in this case, are that no two queens should attack one another, namely that no two queens should be placed on the same row or diagonal. Figure 2.1(a) shows the chess-board for the 4-queens problem, and Figure 2.1(b) shows all of the constraints in relational form. For example, the tuple (1,3) is in the relation R_{12} defined over $\{x_1, x_2\}$, which means that simultaneously assigning the value 1 to variable x_1 and the value 3 to variable x_2 (a queen on row 1 of column 1 and on row 3 of column 2) is allowed by the constraint. The complete definition of the 4-queen problem is $\mathcal{R} = (X, D, C)$, where $X = \{x_1, x_2, x_3, x_4\}$, and for every i $D_i = \{1, 2, 3, 4\}$. There are six constraints: $C_1 = R_{12}$, $C_2 = R_{13}$, $C_3 = R_{14}$, $C_4 = R_{23}$, $C_5 = R_{24}$ and $C_6 = R_{34}$.



(a)

$R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$
$R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$
$R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4)$
$(4,2), (4,3)\}$
$R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$
$R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$
$R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$

(b)

Figure 2.1: The 4-queens constraint network. The network has four variables, all with domains $D_i = \{1, 2, 3, 4\}$. (a) The labeled chess board. (b) The constraints between variables.

2.1.2 Solution of a constraint network

Instantiation. When a variable is assigned a value from its domain, we say that the variable has been *instantiated*. An *instantiation* of a subset of variables is an assignment from its domain to each variable in the subset. Formally, an instantiation of a set of variables $\{x_{i_1}, \ldots, x_{i_k}\}$ is a tuple of ordered pairs $(< x_{i_1}, a_{i_1} >, \ldots, < x_{i_k}, a_{i_k} >)$, where each pair $< x, a >$ represents an assignment of the value a to the variable x, and where a is in the domain of x. We also use the notation $(x_1 = a_1 \ldots, x_i = a_i)$. For simplicity we often abbreviate $(< x_1, a_1 >, \ldots, < x_i, a_i >)$ to $\bar{a} = (a_1, \ldots, a_i)$, and understand an instantiation or tuple as a relation over some scope having a single tuple.

Satisfying a constraint. An instantiation or tuple $\bar{a}$, satisfies a constraint $< S, R >$ iff it is defined over all the variables in S and the components of the tuple $\bar{a}$ associated with S are present in the relation R. For example, let $R_{xyz} = \{(1,1,1), (1,0,1), (0,0,0)\}$. Then the instantiation $\bar{a}$, whose scope is $\{x, y, z, t\}$, given by $\bar{a} = (< x, 1 >, < y, 1 >, < z, 1 >, < t, 0 >)$, satisfies R_{xyz} because its projection on $\{x, y, z\}$ is $(1, 1, 1)$, which is an element of R_{xyz}. However, the instantiation $(< x, 1 >, < y, 0 >, < z, 0 >, < t, 0 >)$ violates R_{xyz} because $(1, 0, 0)$ is not a member of R_{xyz}. Formally,

A consistent partial instantiation. A partial instantiation is *consistent* if it satisfies all of the constraints whose scopes have no un-instantiated variables. A *projection* of a tuple $\bar{a}$ on a subset of its scope S is denoted also by $\bar{a}[S_i]$ (as well as $\pi_{S_i}(\bar{a})$). Using this notation, $\bar{a}$ over S is consistent relative to network $\mathcal{R}$ iff for all S_i in the scheme of $\mathcal{R}$, s.t., $S_i \subseteq S$, $\bar{a}[S_i] \in R_{S_i}$.

Solution. A *solution* of a constraint network $\mathcal{R} = (X, D, C)$, where $X = \{x_1, ..., x_n\}$, is an instantiation of all its variables that satisfies all the constraints. The solution relation $sol(\mathcal{R})$, also denoted ρ_X, is hence defined by:

$$sol(\mathcal{R}) = \{\bar{a} = (a_1, ..., a_n) | a_i \in D_i, \forall S_i \in \ scheme\ of\ \mathcal{R},\ \bar{a}[S_i] \in R_i\}.$$

A network of constraints is said to *express* or *represent* the relation of all its solutions. If we have a constraint network $\mathcal{R}$ over X and a subset of variables $A \subseteq X$, $sol(A)$ or ρ_A is the set of all consistent instantiations over A.

Consider again the constraint network for the 4-queens problem. As there is a constraint between every pair of variables, the scheme of the network is given by $\{\{x_1, x_2\}, \{x_1, x_3\}, \ldots, \{x_3, x_4\}\}$. Consider the set of variables $Y = \{x_1, x_2, x_3\}$. The instantiation $\bar{a} = (< x_1, 1 >, < x_2, 4 >, < x_3, 2 >)$ shown in Figure 2.2(a) is consistent, since

$$\begin{array}{lll} \bar{a}[\{x_1, x_2\}] = (1, 4) & \text{and} & (1, 4) \in R_{12} \\ \bar{a}[\{x_1, x_3\}] = (1, 2) & \text{and} & (1, 2) \in R_{13} \\ \bar{a}[\{x_2, x_3\}] = (4, 2) & \text{and} & (4, 2) \in R_{23} \end{array}$$

although $\bar{a}$ is not part of any full solution. There are only two full solutions to the 4-queens problem, as shown in Figure 2.2(b) and (c). The 4-queens problem represents the relation $\rho_{1234} = \{(2,4,1,3)(3,1,4,2)\}$.

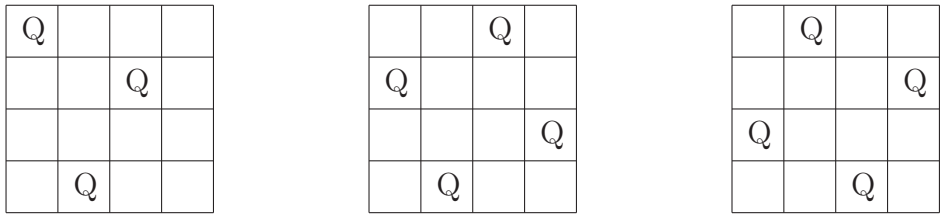


Figure 2.2: Not all consistent instantiations are part of a solution: (a) A consistent instantiation that is not part of a solution. (b) The placement of the queens corresponding to the solution $(2,4,1,3)$. (c) The placement of the queens corresponding to the solution $(3,1,4,2)$.

The crossword puzzle

We return now to our crossword puzzle example from Chapter 1 (Figure 2.3.). A formal constraint network specification is: there is a variable for each square that can hold a character, $x_1, \ldots, x_{13}$, the domains of the variables are the alphabet letters, and the constraints are the possible words. For this example, the constraints are given by:

$R_{\{1,2,3,4,5\}}$ = {(H,O,S,E,S), (L,A,S,E,R), (S,H,E,E,T), (S,N,A,I,L), (S,T,E,E,R)}
$R_{\{3,6,9,12\}}$ = {(A,L,S,O), (E,A,R,N),(H,I,K,E), (I,R,O,N), (S,A,M,E)}
$R_{\{5,7,11\}}$ = {(E,A,T), (L,E,T), (R,U,N), (S,U,N), (T,E,N), (Y,E,S)}

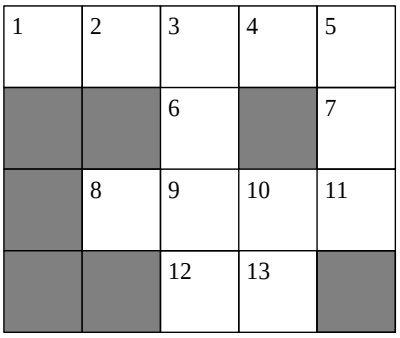


Figure 2.3: A crossword puzzle

$R_{\{8,9,10,11\}}$= $R_{\{3,6,9,12\}}$
$R_{\{10,13\}}$ = {(B,E), (I,T), (N,O), (U,S)}
$R_{\{12,13\}}$ = $R_{\{10,13\}}$.

The reader should verify that the consistent partial assignments over the set of variables $\{x_1, x_2, x_3, x_4, x_5, x_6, x_9, x_{12}\}$ are:

$$\rho_{\{1,2,3,4,5,6,9,12\}} = \{(H, O, S, E, S, A, M, E), (L, A, S, E, R, A, M, E),$$

$$(S, H, E, E, T, A, R, N), (S, N, A, I, L, L, S, O), (S, T, E, E, R, A, R, N)\}.$$

An alternative constraint formulation of the crossword puzzle associates each digit that can start a word with a variable and its possible word assignments as domain values. In this case, the variables are x_1 (1, horizontal), x_2 (3, vertical), x_3 (5, vertical), x_4 (8, horizontal), x_5 (12, horizontal), and x_6 (10, vertical). The scheme of this problem is $\{\{x_1, x_2\}, \{x_1, x_3\}, \{x_4, x_2\}, \{x_4, x_3\}, \{x_5, x_2\}, \{x_6, x_4\}, \{x_6, x_5\}\}$ because there is a constraint between x_1 (1, horizontal) and x_2 (3, vertical), and so on. The domains are: $D_1 = \{hoses, laser, sheet, snail, steer\}$, $D_2 = D_4 = \{also, earn, hike, iron, same\}$, $D_3 = \{eat, let, run, sun, ten, yes\}$ and $D_5 = D_6 = \{be, it, no, us\}$. The constraint between x_1 and x_2, is given by

$$R_{12} = \{(hoses, same)(laser, same), (sheet, earn)(snail, also), (steer, earn)\}.$$

A partial consistent tuple in this case is : $(x_1 = sheet, x_2 = earn, x_3 = ten, x_4 = iron, x_5 = no)$. Note that this formulation of the problem is binary, namely, involving constraints on pairs of variables only. Note also that the network has no solutions.

2.1.3 Constraint Graphs

Primal constraint graphs

A constraint network can be represented by a graph called a *primal constraint graph* or just a *constraint graph*, where each node represents a variable and the arcs connect all nodes whose variables are included in a constraint scope of the problem. The absence of an arc between two nodes indicates that there is no direct constraint – the one specified in the input – between the corresponding variables. For binary constraints a missing arc means that there is implicitly a *universal binary relation* allowing every pair of values. Figure 2.4(a) shows the constraint graph associated with our second formulation of the crossword puzzle, and Figure 2.4(b) shows the constraint graph of the 4-queens problem, which is complete.

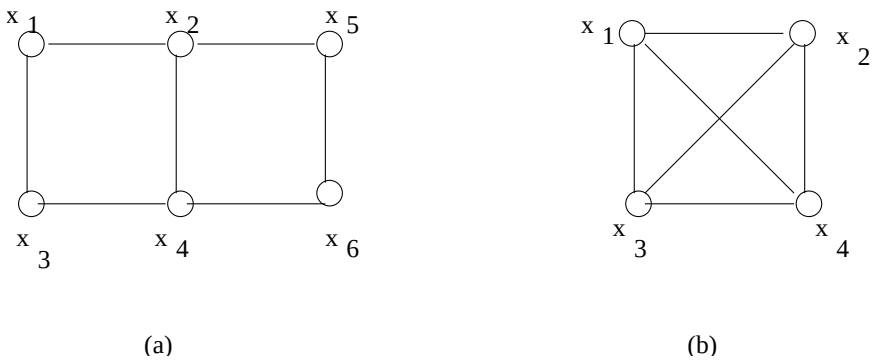


Figure 2.4: Constraint graphs of (a) the crossword puzzle and (b) the 4-queens problem.

A scheduling example

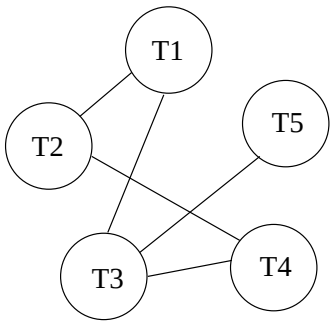
The constraint framework is useful for expressing and solving scheduling problems. Consider the problem of scheduling five tasks, T1, T2, T3, T4, T5, each of which takes one hour to complete. The tasks may start at 1:00, 2:00 or 3:00. Tasks can be executed simultaneously, subject to the restrictions that T1 must start after T3, T3 must start before T4 and after T5, T2 cannot execute at the same time as T1 or T4, and T4 cannot start at 2:00.

Given our five tasks and three time slots, we can model the scheduling problem by creating five variables, one for each task, and giving each variable the domain {1:00, 2:00, 3:00}. Another possible approach is to create three variables, one for each starting time, and to assign each of these variables a domain which is the powerset of {T1, T2, T3, T4, T5} denoting tasks that start at that time point. This second approach is somewhat oakward and is left as an exercise at the end of the chapter. The problem's constraint graph for the first formulation is shown in Figure 2.5. The constraint relations are shown beside the figure.

Dual constraint graphs and hypergraphs

The primal constraint graph is well defined for both binary and non-binary constraints. However, a hypergraph representation more accurately maintains the association between arcs and constraints in the non-binary case.

Definition 2.1.1 (hypergraph) *A hypergraph is a structure* $\mathcal{H} = (V, S)$ *that consists of vertices* $V = \{v_1, .., v_n\}$ *and a set of subsets of these vertices* $S = \{S_1, ..., S_l\}$, $S_i \subseteq V$, *called hyperedges. The hyperedges differ from regular edges in that they "connect" (or are defined over) one or two variables, or more.*



Unary constraint
D_{T4} = {1:00, 3:00}
Binary constraints
$R_{\{T1,T2\}}$: {(1:00,2:00), (1:00,3:00), (2:00,1:00), (2:00,3:00), (3:00,1:00), (3:00,2:00)}
$R_{\{T1,T3\}}$: {(2:00,1:00), (3:00,1:00), (3:00,2:00)}
$R_{\{T2,T4\}}$: {(1:00,2:00), (1:00,3:00), (2:00,1:00), (2:00,3:00), (3:00,1:00), (3:00,2:00)}
$R_{\{T3,T4\}}$: {(1:00,2:00), (1:00,3:00), (2:00,3:00)}
$R_{\{T3,T5\}}$: {(2:00,1:00), (3:00,1:00), (3:00,2:00)}

Figure 2.5: The constraint graph and constraint relations of the scheduling problem in Example 1.

In the *constraint hypergraph* representation of a constraint problem, nodes represent the variables, and *hyperarcs* (drawn as regions) are the scopes of constraints. The hyperarcs group those variables that belong to the same scope. A related representation is the *dual constraint graph*. A *dual constraint graph* represents each constraint scope by a node and associates a labeled arc with any two nodes whose *constraint scopes* share variables. The arcs are labeled by the shared variables. Figure 2.6 depicts the *primal* and the *dual* graphs of our first formulation of the crossword puzzle.

Notice that the structure of dual graph of the first formulation (Figure 2.6b) is identical to the structure of the primal constraint graph of the second formulation (Figure 2.4a). Indeed, the *dual* constraint graph suggests a transformation of a nonbinary network into a special type of *binary* network called the *dual problem*, where the constraints themselves (of the primary problem) are the variables, denoted *c-variables*. The domain of each c-variable ranges over all possible value combinations permitted by the corresponding constraint. There exists a constraint between any two adjacent c-variables enforcing the restriction that their shared (original) variables must be assigned the same values (i.e., the c-variables are bound by equality constraints). In this way, any set of constraints can be transformed into a *binary* constraint network and solved by binary network techniques. For the crossword puzzle, both the dual and the primal representations are intuitive; people will frequently come up with either of these formulations as their first representation of the problem.

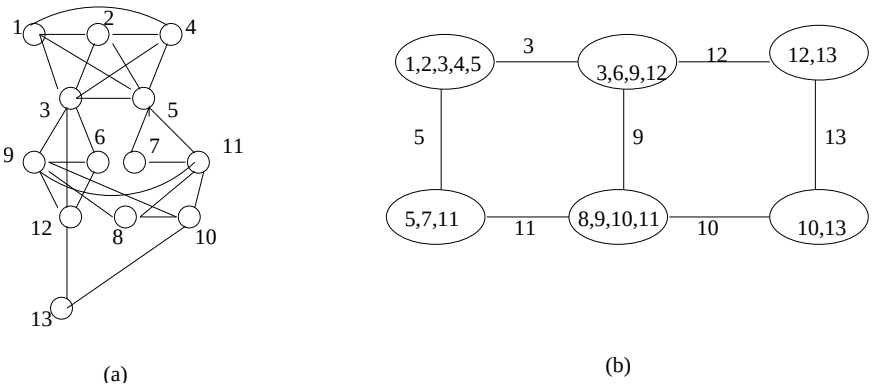


Figure 2.6: Constraint graphs of the crossword problem: (a) primal and (b) dual.

The Radio link frequency assignment problem

Radio link frequency assignment problems are communication problems where the goal is to assign frequencies to a set of radio links in such a way that all the links may operate together without noticeable interference. The French "Centre d'lectronique de l'Armement" (CELAR) has made available a set of Radio Link Frequency Assignment benchmark problems (RLFAP) built from a real network, with simplified data. The constraints are all binary non linear and the variables have finite domains. These are real-world size problems, the larger instances having around 1000 variables and more than 5000 constraints. All these instances have been built from a unique real instance of 916 links and 5744 constraints in 11 connected components. Each radio link is represented by a variable whose domain is the set of all frequencies available for that link. The essential constraints involve two variables F_1 and F_2: $|F1 - F2| > k_{12}$. The two variables represent two radio links and the constant k_{12} depends on the position of the two links and also on the physical environment. For each pair of sites, two frequencies must be assigned, one for the communications from A to B, and the other for the communications from B to A.

Figure 2.7 show three constraint graphs of instances of the radio link frequency assignment problems from the CELAR benchmark, showing the diversity in structure for some problems.

The Huffman-Claws labeling

The Huffman-Claws junction labellings are another example of intuitive dual-primal formulation. In one formulation of the problem as a constraint network, the various junctions in the problem instance are variables, their domains are the possible label combinations on junction types, and the constraints are expressed using (0,1)-matrices as in Figure 2.9.

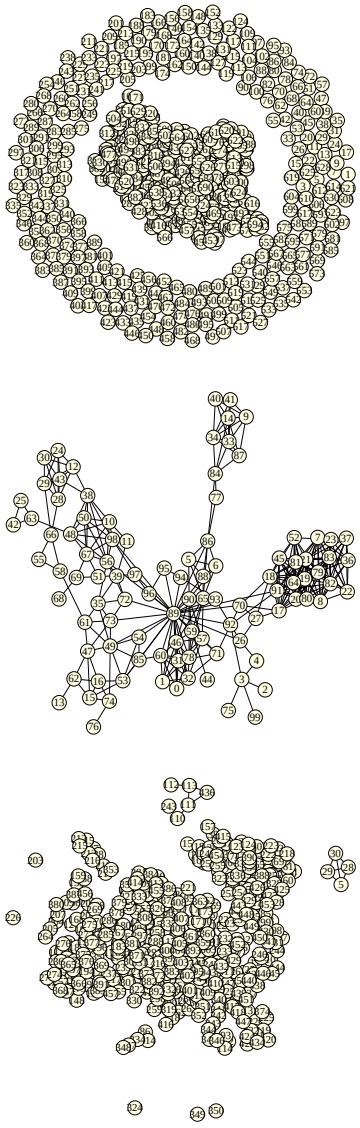


Figure 2.7: Constraint graphs of 3 instances of the Radio frequency assignment problem in CELAR's benchmark

For example, x_i denotes junction i in the cube. Since 1 is a Fork junction, the values of x_1 are the label combinations that can be assigned such a junction. The label assignments are constrained by the requirement that if an edge connects two junctions, it must get the same label from each junction. The $(0,1)$-matrix constraints representation assumes that the domains of each junction variable (in Figure 2.8) are ordered from left to right. In this formulation, all the constraints are binary and the constraint graph is identical to the input problem graph as shown in Figure 2.9. The set of solutions to this problem is given in Figure 1.5.

The above formulation may also be perceived as the dual graph of a nonbinary formulation that views the edges as the variables, each having the domains of $\{+, -, \rightarrow, \leftarrow\}$, and the constraints as binary or ternary depending on the junction types. For example the edges (1,3),(1,5) (1,2) are three variables in a scope of a single constraint in the second formulation. In the primal constraint graph of this second formulation, two nodes (representing edges) will be connected if they participate in the same junction (see exercises).

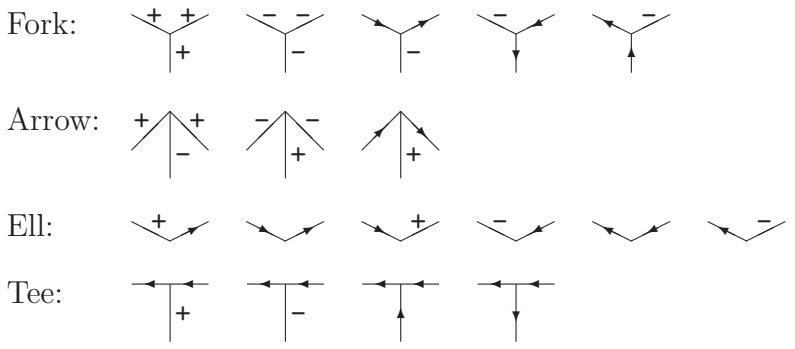
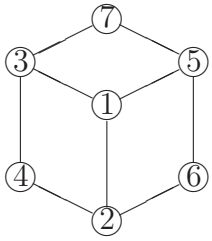


Figure 2.8: Huffman-Clawes junction labelings

2.2 Numeric and Boolean constraints

By explicitly specifying the allowed tuples via the syntax of relations, we do a good job of describing the underlying meaning of a constraint. However, as we saw in Section 2.1b, this method of specification is sometimes tedious and unwieldy. In many instances, mathematical conventions can describe the relationships between objects more concisely or conveniently than relational descriptions. Arithmetic constraints and boolean constraints are examples of alternative languages for describing constraints. The first allows for more concise expression while the second, in addition to being restricted to bi-valued domains, expresses the forbidden tuples rather then the legal ones.

$$R_{21} = \begin{bmatrix} 0\,1\,1\,0\,0 \\ 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{bmatrix} \quad R_{31} = \begin{bmatrix} 0\,1\,0\,1\,0 \\ 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{bmatrix} \quad R_{51} = \begin{bmatrix} 0\,1\,0\,0\,1 \\ 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{bmatrix}$$



$$R_{24} = R_{37} = R_{56} = \begin{bmatrix} 0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1 \\ 0\,0\,0\,1\,1\,0 \end{bmatrix}$$

$$R_{26} = R_{34} = R_{57} = \begin{bmatrix} 1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,1 \end{bmatrix}$$

Figure 2.9: Scene labeling constraint network

2.2.1 Numeric Constraints

Numeric constraints express constraints by arithmetic expressions. Consider expressing our 4-queens problem with numeric constraints. Instead of describing the constraints by enumerating the allowed elements of each relation, we can quite succinctly state that every two variables x_i and x_j should satisfy: $\forall i, j, \;\; x_i \neq x_j, \; and \; |x_i - x_j| \neq |i - j|$ defining the relation

$$R_{ij} = \{(x_i, x_j) \mid x_i \in D_i, x_j \in D_j, x_i \neq x_j, \text{ and } |x_i - x_j| \neq |i - j|\}.$$

In another example, let the domains of the variables be finite subsets of the integers, and let a binary constraint between two variables be a conjunction of linear inequalities of the form $ax_i - bx_j = c$, $ax_i - bx_j < c$, or $ax_i - bx_j \leq c$, where a, b, and c are integer constants. For example, the conjunction

$$(3x_i + 2x_j \leq 3) \wedge (-4x_i + 5x_j < 1)$$

is a legitimate constraint between variables x_i and x_j. A network with constraints of this form can be formulated as "an integer linear program" where each constraint is defined over two variables, and the domains of the variables are restricted to being finite subsets of the integers. Linear constraints are a special subclass of numeric constraints widely applicable in the areas of scheduling and temporal and spatial reasoning.

Cryptarithmetic puzzles

One class of toy problems that can be easily formulated with linear constraints are the *cryptarithmetic puzzles*, such as: SEND + MORE = MONEY. Here we are asked to

replace each letter by a different digit so that the above equation is correct. The constraint formulation of these puzzles will associate each letter with a variable whose domains are the digits $\{0..9\}$. The exact formulation of this problem is left to you as an exercise at the end of the chapter.

2.2.2 Boolean constraints and propositional cnf

When the variables of a constraint problem range over two values, we frequently use a boolean propositional language to describe the various relationships. Assume that you would like to invite your friends Alex, Bill, and Chris to a party. Let A, B, and C denote the propositions "Alex comes", "Bill comes" and "Chris comes", respectively. You know that if Alex comes to the party, Bill will come as well, and if Chris comes, then Alex will too. This can be expressed in propositional calculus as $(A \rightarrow B) \wedge (C \rightarrow A)$, or equivalently as $(\neg A \vee B) \wedge (\neg C \vee A)$. Assume now that Chris came to the party; should you expect to see Bill? Or, in propositional logic, does the propositional theory $\varphi = C \wedge (A \rightarrow B) \wedge (C \rightarrow A)$ entail B? A common way to answer this query is to assume that Bill will not come and check whether this is a plausible situation (i.e., decide if $\varphi\prime = \varphi \wedge \neg B$ is satisfiable). If $\varphi\prime$ is unsatisfiable, we can conclude that φ entails B. We next provide the formal definitions.

Propositional variables take only two values: $\{true, false\}$ or "1" and "0." We denote propositional *variables* by uppercase letters $P, Q, R, \ldots$, propositional literals (i.e., $P, \neg P$) stand for $P =$ "$true$" or $P =$ "$false$," and disjunctions of literals, or *clauses*, are denoted by $\alpha, \beta, \ldots$. For instance, $\alpha = (P \vee Q \vee R)$ is a clause. We will sometime denote by $\{P, Q, R\}$ the clause $(P \vee Q \vee R)$. A *unit clause* is a clause of size 1. The notation $(\alpha \vee T)$, when $\alpha = (P \vee Q \vee R)$, is shorthand for the disjunction $(P \vee Q \vee R \vee T)$. $\alpha \vee \beta$ denotes the clause whose literal appears in either α or β. The *resolution* operation over two clauses $(\alpha \vee Q)$ and $(\beta \vee \neg Q)$ results in a clause $(\alpha \vee \beta)$, thus eliminating Q. A formula φ in conjunctive normal form (*CNF*), referred to as a *theory*, is a set of clauses $\varphi = \{\alpha_1, \ldots, \alpha_t\}$ that denotes their conjunction. The set of *models* or *solutions* of a theory φ is the set of all truth assignments to all variables that do not violate any clause.

In general, a CNF theory is a constraint network whose variables are the propositions, the domains have two values $\{true, false\}$ or $\{0, 1\}$. Each clause is a constraint on the corresponding propositional variables. For instance, the CNF theory $\varphi = (A \vee B) \wedge (C \vee \neg B)$ has three variables and two constraints. The constraint $A \vee B$ expresses the relation $R_{AB} = \{(0, 1), (1, 0), (1, 1)\}$. Note that we may have more than one clause defined on the same set of propositional variables.

The propositional satisfiability problem (SAT) is to decide whether a given cnf theory has a *model*, (that is, a truth assignment that does not violate any clause), alternatively, weather the associated constraint problem is consistent. The structure of a proposi-

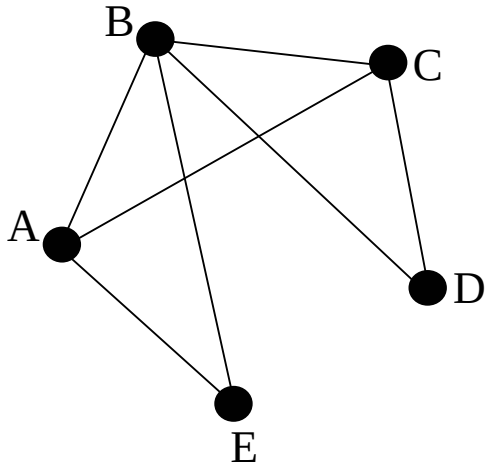


Figure 2.10: The interaction graph of theory $\varphi_1 = \{(\neg C), (A \vee B \vee C), (\neg A \vee B \vee E), (\neg B \vee C \vee D)\}$

tional theory can be described by an *interaction graph* which corresponds to the (primal) constraint graph. The interaction graph of a propositional theory φ, denoted $G(\varphi)$, is an undirected graph that contains a node for each propositional variable and an edge for each pair of nodes that correspond to variables appearing in the same clause. For example, the interaction graph of theory $\varphi_1 = \{(\neg C), (A \vee B \vee C), (\neg A \vee B \vee E), (\neg B \vee C \vee D)\}$ is shown in Figure 2.10.

2.2.3 Combinatorial circuits diagnosis

Boolean propositional languages are often used to express combinatorial circuits built out of AND, OR and XOR gates (see Figure 2.11). The diagnosis task over such circuits can be formulated as a constraint satisfaction problem. In *circuit diagnosis* we are given a description of the circuit, composed of combined inputs through boolean gates to their respective outputs. If the expected output is different from the observed output, our task is to identify a subset of the gates that, if assumed to be faulty, explains the observed output.

One method of formulating the diagnosis task within a constraint network is to associate each gate, as well as its inputs and outputs, with a variable and then to describe by Boolean constraints the relationship between the input and output of each gate under the assumption that the gate is either proper or faulty. Once we observe an output fault, the task of explaining the circuit behavior is completed by identifying a set of faulty gates. (Often the minimal such set is required). Such an explanation amounts to finding a consistent solution for the formulated network.

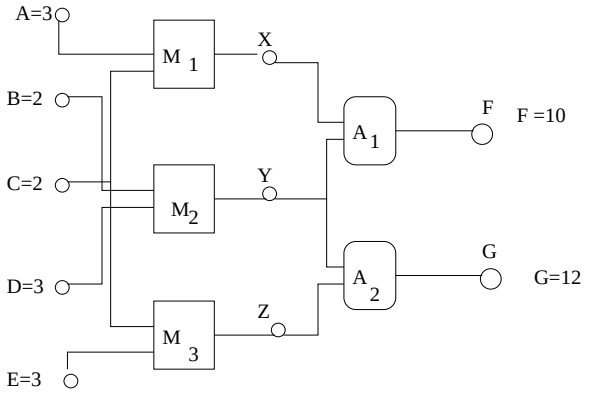


Figure 2.11: A combinatorial circuit: M is a multiplier, A is an adder

In the example in Figure 2.11 there are three multipliers and two adders. The constraint formulation will have a variable for each input $\{A, B, C, D, E\}$, output $\{F, G\}$ and intermediate output $\{X, Y, Z\}$, and for each component M_1, M_2, M_3, A_1, A_2. The domains of the input and output variables are any number (integer or Boolean if we so choose to restrict the problem). The domain of the components are $\{0, 1\}$, where 1 indicates a faulty behavior of the component and 0 a correct behavior. A constraint is associated with each component variable, its input variables and output variables, yielding 5 constraints, each defined over 4 variables.

If the inputs and outputs are Boolean variables, the constraints can be expressed by Boolean expression. For example, if M_1 is an AND gate, its associated constraint can be: $M_1 \rightarrow (A \wedge C \rightarrow X)$. You will be asked to formulate a detailed description of the constraints in the exercises.

2.3 Properties of binary constraint networks

Many constraint processing concepts were initially introduced for binary networks. It is helpful to discuss binary networks separately in this section because many important concepts are more easily digested in this restricted case. Understanding concepts such as minimal network and decomposability, will allow deep understanding of issues that underly the general theory of constraints.

2.3.1 Equivalence and deduction with constraints

A central concept of constraint processing is constraint deduction or *constraint inference*. New constraints can be inferred from an initial set of constraints. These newly inferred

constraints might take the form of constraints between variables that were not initially constrained or might be tightenings of existing constraints. For instance, from the algebraic constraints $x \geq y$ and $y \geq z$ we can infer that $x \geq z$. Note that adding the inferred constraint $x \geq z$ yields an equivalent constraint network. The next example demonstrates constraint inference using relational representation.

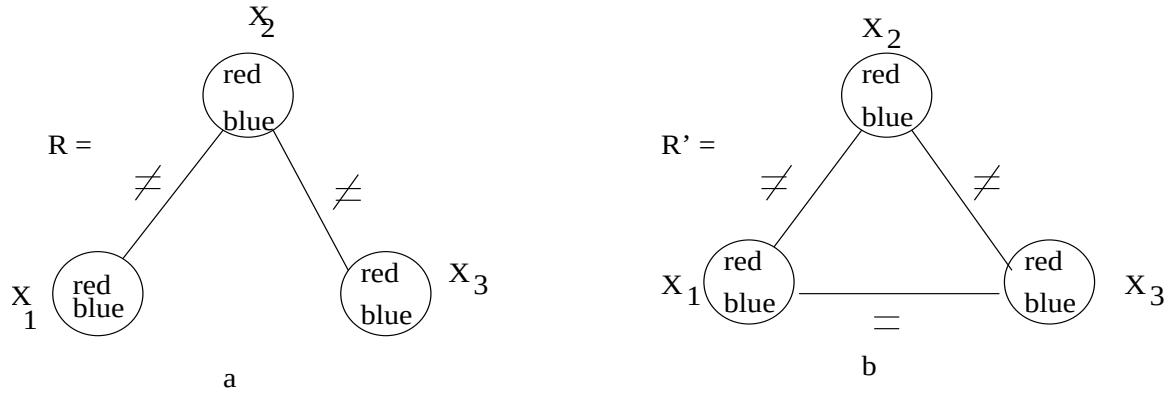


Figure 2.12: (a) A graph $\mathcal{R}$ to be colored by two colors, (b) an equivalent representation $\mathcal{R}'$ having a newly inferred constraint between x_1 and x_3.

Example 2.3.1 Consider the graph-coloring problem in Figure 2.12a. The problem can be described by a constraint network with three variables, x_1, x_2, and x_3, one for each node, all defined on the same domain values $\{red, blue\}$, and the not-equal constraints: $R_{21} = R_{32} = \{(blue, red), (red, blue)\}$. The lack of an arc between x_1 and x_3 represents the universal relation $R_{13} = \{(red, blue), (blue, red), (red, red), (blue, blue)\}$. The problem has two solutions: $\rho_{123} = \{(red, blue, red)(blue, red, blue)\}$. Assume now that we tighten the constraint between x_1 and x_3, disallowing the pair $(< x_1, red >, < x_3, blue >)$. Since this pair does not participate in any of the original problem's solutions, this restriction does not alter the set of solutions. In fact, if we add the constraint $R'_{13} = \{(red, red)(blue, blue)\}$, we get a new constraint network $\mathcal{R}'$ having the same set of solutions (see Figure 2.12b). Indeed, the constraint R'_{13} which enforces $x_1 = x_3$ can be *inferred* from the original network $\mathcal{R}$. The two networks $\mathcal{R}$ and $\mathcal{R}'$ are said to be *equivalent.* □

An inferred constraint can also be viewed as *redundant* relative to a constraint network since its deletion from the network will not change the set of all solutions. In $\mathcal{R}'$, for instance, the not-equal constraint between x_1 and x_2 is redundant since its deletion does not change the set of solutions. The same redundancy exists for each of the other two constraints. However, once R'_{12} is removed from $\mathcal{R}'$, the other constraints are no longer redundant. In summary, two constraint networks are equivalent if they are defined on the

same set of variables and express the same set of solutions. A constraint R_{ij} is redundant relative to $\mathcal{R}'$ iff $\mathcal{R}'$ is equivalent to $\mathcal{R}$ when R_{ij} is removed.

Constraint deduction can be accomplished through the *composition* operation.

Definition 2.3.2 (composition) *Given two binary or unary constraints R_{xy} and R_{yz}, the composition $R_{xy} \cdot R_{yz}$ generates the binary relation R_{xz} defined by:*

$$R_{xz} = \{(a,b) | a \in D_x,\ b \in D_z\ , \exists c \in D_y \text{ s.t. } (a,c) \in R_{xy} \text{ and } (c,b) \in R_{yz}\}$$

An alternative, operational definition of composition is formulated in terms of the join and projection operators:

$$R_{xz} = R_{xy} \cdot R_{yz} = \pi_{\{x,z\}}(R_{xy} \bowtie R_{yz}).$$

In Figure 2.12a, we deduced that $R'_{13} = \pi_{\{x_1,x_3\}}(R_{12} \bowtie R_{23}) = \{(red, red), (blue, blue)\}$, thus yielding the equivalent network in Figure 2.12b. The composition operation can also be described via boolean matrix multiplication when binary relations are expressed using $(0,1)$ matrices.

Example 2.3.3 Continuing with our simple graph-coloring example, the two inequality constraints can be expressed as 2×2 matrices having zeros along the main diagonal.

$$R_{12} = \left(\begin{array}{c|cc} & red & blue \\ \hline red & 0 & 1 \\ blue & 1 & 0 \end{array}\right) \qquad R_{23} = \left(\begin{array}{c|cc} & red & blue \\ \hline red & 0 & 1 \\ blue & 1 & 0 \end{array}\right)$$

Multiplying two such matrices yields the following two-dimensional identity matrix:

$$R_{12} \cdot R_{23} = R_{13} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \left(\begin{array}{c|cc} & red & blue \\ \hline red & 1 & 0 \\ blue & 0 & 1 \end{array}\right)$$

□

Note that composition is not generally distributive with respect to intersection. Namely: $R_{12} \cdot (R_{23} \cap R'_{23}) \neq (R_{12} \cdot R_{23}) \cap (R_{12} \cdot R'_{23})$.

2.3.2 The minimal and the projection networks

In his seminal paper [214], Montanari considers the expressive power of binary networks. The question of interest is whether an arbitrary relation can be a set of solutions to some underlying binary constraint network having the same set of variables. In other words, can a relation be represented as a binary constraint network?

In fact, most relations cannot be represented by a collection of binary constraints, since there are many more different relations possible on n variables than there are networks of binary constraints on n variables. Given n variables each having a domain of size k, cardinality arguments dictate that the number of different relations on n variables is 2^{k^n}, which is far greater than the number of different binary constraint networks, $2^{k^2n^2}$. The proof of this argument is left to you as an exercise. Accordingly, those special cases where a relation is expressible by a binary constraint network, are highly desirable, since they may require less space to specify.

A relation that cannot be expressed by a binary network may still be approximated by one. Let's consider the approximation of a relation by its binary *projection network*.

Definition 2.3.4 (projection network) *The projection network of a relation ρ is obtained by projecting ρ onto each pair of its variables. Formally, if ρ is a relation over $X = \{x_1, ..., x_n\}$, its projection network, $P(\rho)$ is defined by the network $\mathcal{P} = (X, D, P)$ where $D = \{D_i\}$, $D_i = \pi_i(\rho)$ and $P = \{P_{ij}\}$, where $P_{ij} = \pi_{x_i,x_j}(\rho)$.*

Example 2.3.5 Let $\rho_{123} = \{(1,1,2)(1,2,2)(1,2,1)\}$. The projection network $P(\rho)$ includes the constraints $P_{12} = \{(1,1)(1,2)\}$, $P_{13} = \{(1,2)(1,1)\}$, and $P_{23} = \{(1,2)(2,2)(2,1)\}$. Generating all solutions of $P(\rho)$ yields $sol(P(\rho)) = \{(1,1,2)(1,2,2)(1,2,1)\}$. □

What is the relationship between the original relation and its approximation, between ρ and $P(\rho)$? Generating all solutions of $P(\rho)$, in Example 2.3.5, yields back the relation ρ. But this cannot be characteristic of the general case, else we would have proven that every relation has a binary network representation by its projection network, and we know this to be false due to the cardinality argument presented earlier. What does hold in general is that the projection network is the best *upper bound network approximation* of a relation. That is, for every relation ρ, the solution set of $P(\rho)$ contains ρ. Moreover, as we will show, any other upper bound network will express a solution set that includes the projection network's solutions. The following example illustrates this useful feature of the projection network.

Example 2.3.6 Consider a slightly different relation ρ:

x_1	x_2	x_3
1	1	2
1	2	2
2	1	3
2	2	2

The projection network $P(\rho)$ has the following constraints: $P_{12} = \{(1,1)(1,2)(2,1)(2,2)\}$, $P_{23} = \{(1,2)(2,2)(1,3)\}$ and $P_{13} = \{(1,2)(2,3)(2,2)\}$. The set of solutions to $P(\rho)$, $sol(P(\rho))$, are:

x_1	x_2	x_3
1	1	2
1	2	2
2	1	2
2	1	3
2	2	2

□

We see that all the tuples of ρ appear in the solution set of $P(\rho)$, while some additional solutions to $P(\rho)$ are not in ρ. In general:

Theorem 2.3.7 *For every relation ρ, $\rho \subseteq sol(P(\rho))$.*

Proof: Let $t \in \rho$. We have to show only that $t \in sol(P(\rho))$, namely, that it satisfies every binary constraint in $P(\rho)$. This is clearly true, since, by its definition, every pair of values of t was included, by projection, in the corresponding constraint of $P(\rho)$. □

Is there another binary network of constraints $\mathcal{R}'$, whose solution set contains ρ but is smaller than $P(\rho)$? Might we discover such a network by tightening the constraints of $P = P(\rho)$? Let's try to eliminate the superfluous tuple $(2,1,2)$ from $sol(P)$ in Example 2.3.6 by deleting the pair $(2,1)$ from the projection constraint P_{12}. Unfortunately, when we do this, we also exclude the tuple $(2,1,3)$, which is in ρ. Similarly, when we try to exclude $(2,1,2)$ by deleting the pair $(1,2)$ from P_{23}, we eliminate $(1,1,2)$, which is also in ρ. And, if we exclude $(2,2)$ from P_{13}, we eliminate $(2,2,2)$. Indeed, $P(\rho)$ cannot be tightened any further if its solutions are to include all the tuples in ρ. In fact,

Theorem 2.3.8 *The projection network $P(\rho)$ is the tightest upper bound binary network representation of ρ; there is no binary network $\mathcal{R}'$, s.t. $\rho \subseteq sol(R') \subset sol(P(\rho))$.*

Consequently, if a relation is not expressible by its projection network, it cannot be expressed by any binary network of constraints.

Theorem 2.3.8 shows that the projection network is the most accurate binary network upper bound for a relation. But is it also the tightest explicit description of relation $sol(P(\rho))$? This question leads us to the notion of a partial order of tightness among binary constraint networks. As we have seen in Figure 2.12a and b, the networks $\mathcal{R}$ and $\mathcal{R}'$ are semantically equivalent since they represent the same set of solutions, yet $\mathcal{R}'$ is tighter than $\mathcal{R}$ using a pairwise comparison of their binary relations.

Definition 2.3.9 ('tighter than' network relationship) *Given two binary networks, $\mathcal{R}'$ and $\mathcal{R}$, on the same set of variables $x_1, ..., x_n$, $\mathcal{R}'$ is* at least as tight as *$\mathcal{R}$ iff for every i and j, $R'_{ij} \subseteq R_{ij}$.*

Obviously, if $\mathcal{R}'$ is tighter than $\mathcal{R}$, then the solution set of $\mathcal{R}'$ is contained in the solution set of $\mathcal{R}$. Oftentimes, however, the tighter network still expresses the same set of solutions. Moreover, if we take the intersection of two equivalent networks (by intersecting constraints pairwise), we get yet another equivalent network that is tighter than either.

Definition 2.3.10 (intersection on networks) *The intersection of two networks $\mathcal{R}$ and $\mathcal{R}'$, denoted $\mathcal{R} \cap \mathcal{R}'$, is the binary network obtained by pairwise intersection of the corresponding constraints in the two networks.*

Clearly,

Proposition 2.3.11 *If $\mathcal{R}$ and $\mathcal{R}'$ are two equivalent networks, then $\mathcal{R} \cap \mathcal{R}'$ is equivalent to and is at least as tight as both.*

The proof is left as an exercise.

Example 2.3.12 Consider the network in Figure 2.12a and the network in Figure 2.12b without R_{23}. Intersecting the two networks yield the equivalent network in Figure 2.12b that is tighter than either. □

There exists, therefore, a partial order of tightness amongst all equivalent networks. If we intersect all these networks, we get one unique network that is both equivalent to all the networks and at least as tight as all networks. This network is called the *minimal network*. The minimal network $M(\mathcal{R})$ of a binary network $\mathcal{R}$ is the tightest network equivalent to $\mathcal{R}$. $M(\mathcal{R})$ is also denoted $M(\rho)$ when ρ is the set of solutions to $\mathcal{R}$.

Definition 2.3.13 (minimal network) *Let $\{\mathcal{R}_1, ...\mathcal{R}_l\}$ be the set of all networks equivalent to $\mathcal{R}_0$ and let $\rho = sol(\mathcal{R}_0)$. Then the minimal network M of $\mathcal{R}_0$ or of ρ is defined by $M(\mathcal{R}_0) = M(\rho) = \cap_{i=1}^{l} \mathcal{R}_i$.*

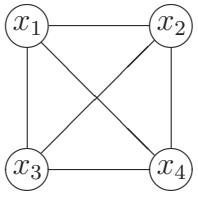
Finally, it is possible to show that the minimal network is identical to the projection network of the minimal network's set of solutions.

Theorem 2.3.14 *For every binary network $\mathcal{R}$ s.t. $\rho = sol(\mathcal{R})$, $M(\rho) = P(\rho)$.*

Proof: Left as an exercise

Figure 2.13 shows the constraint graph and the binary minimal constraints of the 4-queens problem. We denote the binary constraints in the minimal network by M_{ij}. The unary constraints are the reduced domains. (Compare this network to the equivalent one in Figure 2.1.) The minimal network is perfectly explicit for unary and binary constraints. That is to say, if a pair of values is permitted by the minimal network, then it is guaranteed to appear in at least one solution. Indeed, it immediately follows from Theorem 2.3.14 that,

Proposition 2.3.15 *If $(a, b) \in M_{ij}$ then $\exists\ t \in sol(M)$ s.t. $t[i] = a$ and $t[j] = b$.*



(a)

$M_{12} = \{(2,4), (3,1)\}$
$M_{13} = \{(2,1), (3,4)\}$
$M_{14} = \{(2,3), (3,2)\}$
$M_{23} = \{(1,4), (4,1)\}$
$M_{24} = \{(1,2), (4,3)\}$
$M_{34} = \{(1,3), (4,2)\}$

(b)

$D_1 = \{1,3\}$
$D_2 = \{1,4\}$
$D_3 = \{1,4\ \}$
$D_4 = \{1,3\}$

(c)

Figure 2.13: The 4-queens constraint network. (a) The constraint graph. (b) The minimal binary constraints. (c) The minimal unary constraints (the domains).

We should note here that finding a single solution of a minimal network of constraints is not guaranteed to be easy. In fact, deciding whether ρ can be representable by its projection network is NP-hard. It is still not clear, however, whether or not generating a single solution of a minimal network is hard. Empirical experience shows that generating a single solution using the minimal network is normally easy. Nevertheless, we do speculate that generating a single solution from the minimal network is hard and we leave this as an exercise.

2.3.3 Decomposable networks

While we can always easily find a partial solution of size 2 that is part of a full solution when given a minimal network, we cannot guarantee that we can extend a two-variable solution to a third variable unless the minimal network is also *decomposable*.

We know by now that a relation has a binary network representation iff it is equivalent to its projection network. We also know that the projection network is the relation's most explicit form. It turns out, however, that a relation may be representable by a binary network even if many of its projections are not.

Example 2.3.16 Consider the relations

$$\rho = \left(\begin{array}{cccc} x & y & z & t \\ \hline a & a & a & a \\ a & b & b & b \\ b & b & a & c \end{array}\right) \qquad \pi_{xyz}\rho = \left(\begin{array}{ccc} x & y & z \\ \hline a & a & a \\ a & b & b \\ b & b & a \end{array}\right)$$

You can easily verify that ρ is representable by a binary constraint network. However, the projected relation $\pi_{xyz}\rho$ is not expressible by binary networks because

$$\pi_{xyz}\rho \subset sol(P(\pi_{xyz}\rho))$$

□

Definition 2.3.17 *A relation is* decomposable *if it is expressible by a network of binary constraints and iff each of its projected relations is also expressible by a binary network of constraints.*

If a relation is decomposable, then the projection network expresses the relation and all its projections. In other words, if ρ is a decomposable relation and S is a subset of the variables, then $\pi_S\rho$ is expressible by the subnetwork of $P(\rho)$ whose variables are restricted to the variables in S (see exercises).

2.4 Summary

In this chapter we provide a formal presentation of constraint networks, their solutions and their graph representations, and have demonstrated these concepts through examples. We end it by focusing on some formal properties of binary constraint networks. We defined concepts such as *inferred constraint, redundant constraint*, partial-order of tightness between constraints, the projection network and the minimal network. We showed that those two latter notions coincide. Finally we defined the concept of decomposable networks.

2.5 Chapter Notes

Graphical properties of constraint networks were initially investigated through the class of binary constraint networks. Montanari [214] was the first to formally define these networks. Montanari also introduced most of the concepts mentioned in the second part of this chapter, including minimal network, projection network and decomposable network. He also discussed important notions of constraint propagations, which are the focus of the next chapter.

Exercises

1. Nadel [216] proposes a variant of n-queens called *confused n-queens*. The problem is to find all ways to place n-queens on an $n \times n$ chess board, one queen per column, so that all pairs of queens *do* attack each other. Propose a formulation of the problem as a constraint network. Identify variables, domains and constraints.

2. (Introductory Combinatorics, R. A. Brunaldi, 1977.) A Latin Square of order n is defined to be an $n \times n$ array made out of n distinct symbols (usually the integers $1, 2, \ldots, n$) with the defining characteristic that each of the n symbols occurs exactly once in each row of the array and exactly once in each column. For example,

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$

   Orthogonal Latin Squares: Let $\mathbf{A}$ and $\mathbf{B}$ be Latin squares of order n and let the entry in the i^{th} row and the j^{th} column of $\mathbf{A}$ and $\mathbf{B}$ be denoted as a_{ij} and b_{ij}, respectively, with $i, j = 1, 2, \ldots, n$. $\mathbf{A}$ and $\mathbf{B}$ are *orthogonal* if the n^2 order pairs (a_{ij}, b_{ij}) are all distinct. For example, the following juxtaposed Latin Squares are orthogonal:

$$\begin{bmatrix} (3,2) & (2,3) & (1,1) \\ (2,1) & (1,2) & (3,3) \\ (1,3) & (3,1) & (2,2) \end{bmatrix}$$

   Propose a formulation of the problem as a constraint network. Identify variables, domains and constraints.

3. Using the starting times as variables, formulate the scheduling problem in Figure 2.5 as a constraint network problem.

4. Formulate the Huffman-Claws labelling problem as a constraint network where each arc in the drawing is a variable. Draw the primal constraint graph and the dual constraint graph of such formulation of the cube 3-dimensional interpretation.

5. Formulate the Zebra Problem below as a constraint network. Provide the variables, domains and constraints. Draw its primal constraint graph.

   The Zebra Problem: There are five houses in a row, each of a different color, inhabited by women of different nationalities. The owner of each house owns a different pet, serves different drinks, and smokes different cigarettes from the other owners. The following facts are also known:

   The Englishwoman lives in the red house
   The Spaniard owns a dog
   Coffee is drunk in the green house
   The Ukrainian drinks tea
   The green house is immediately to the right of the ivory house
   The Oldgold smoker owns the snail
   Kools are smoked in the yellow house
   Milk is drunk in the middle house
   The Norwegian lives in the first house on the left
   The Chesterfield smoker lives next to the fox owner
   The yellow house is next to the horse owner
   The Lucky Strike smoker drinks orange juice
   The Japanese smokes Parliament
   The Norwegian lives next to the blue house

   The Question: who drinks water and who owns the zebra?

6. Provide two formulations for each of the Cryptarithmetic problems below as a constraint network. Provide the variables, domains and constraints. Draw the primal and the dual constraint graphs. Discuss which formulation is superior, in your opinion.

   (a)
   $$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline & M & O & N & E & Y \end{array}$$

   (b) HOCUS +POCUS = PRESTO

   (c) GERALD + DONALD = ROBERT

7. Provide a detailed formulation of the circuit diagnosis problem in Figure 2.11.

8. Provide a detailed formulation of the design problem presented in Figure 1.3.

9. Provide a formal definition for the Radio link frequency assignment problem. The goal is to assign frequencies to set of radio links in a way that all the links can work together with no interference.

10. Formulate the 3x3 magic square as a constraint problem. Draw its primal and dual graphs.

    A magic square of order n is an an $n \times n$ array of the integers $1, 2, \ldots, n^2$ arranged so that the sum of every row, column, and the two main diagonals is the same. Since

    $$\sum_{i=1}^{n^2} = \frac{1}{2} n^2 (n^2 + 1),$$

    the sum must be $\frac{1}{2} n(n^2 + 1)$. For example,

    $$\begin{bmatrix} 1 & 15 & 24 & 8 & 17 \\ 23 & 7 & 16 & 5 & 14 \\ 20 & 4 & 13 & 22 & 6 \\ 12 & 21 & 10 & 19 & 3 \\ 9 & 18 & 2 & 11 & 25 \end{bmatrix}$$

    is a magic square of order 5, each row, column, and main diagonal add up to $\frac{1}{2} 5(5^2 + 1) = 65$.

11. Find the minimal network of the crossword puzzle (Figure 2.3) when the problem is formulated as a set of binary constraints.

12. Consider the following relation ρ on variables x,y,z,t.

    $$\rho_{xyzt} = \{(a, a, a, a)(a, b, b, b)(b, b, a, c)\}$$

    (a) Find the projection network $P(\rho)$. Is ρ representable by a network of binary constraints? Justify your answer.

    (b) Is the projection $\pi_{xyz}(\rho)$ representable by a network of binary constraints? Is ρ decomposable?

    (c) A search space is backtrack-free along an order of its variables, d, if any partial solution along this order can be extended to a full solution. Can you find an ordering such that $P(\rho)$ is backtrack-free?

(d) Is there a binary network representation of ρ that is backtrack-free in the order x, y, z, t? Is there a binary network representing ρ that is decomposable?

13. Prove that if $\mathcal{R}$ and $\mathcal{R}'$ are two equivalent networks, then $\mathcal{R} \cap \mathcal{R}'$ is equivalent to, and is at least as tight, as both.

14. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two binary networks on the same n variables and the same domains. Prove that if $\mathcal{R}_1$ is tighter than $\mathcal{R}_2$, then $sol(\mathcal{R}_1) \subseteq sol(\mathcal{R}_2)$.

15. Prove: for every binary network $\mathcal{R}$ whose set of solutions is ρ, $M(\rho) = P(\rho)$.

16. *Decomposability*

    (a) Is the set of solutions of the 4-queen problem binary decomposable?

    (b) Prove that if ρ is a binary decomposable relation and S is a subset of its variables, then $\pi_S(\rho)$ is expressible by the subnetwork restricted to variables in S of the projection network $P(\rho)$.

    (c) Is the minimal network always decomposable? Prove or show a counterexample.

17. Prove that the number of relations over n variables with domain size k is 2^{k^n}, while the number of binary constraint networks is $2^{k^2n^2}$.

18. Prove that composition is not generally distributive with respect to intersection. Namely: $R_{12} \cdot (R_{23} \cap R'_{23}) \neq (R_{12} \cdot R_{23}) \cap (R_{12} \cdot R'_{23})$.