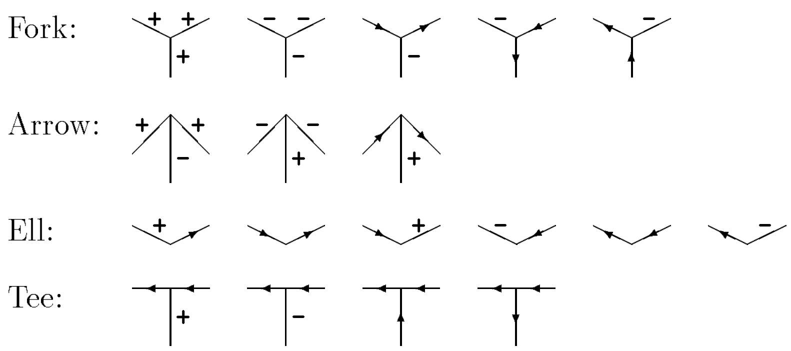

Figure 1.4: Huffman-Clowes junction labelings

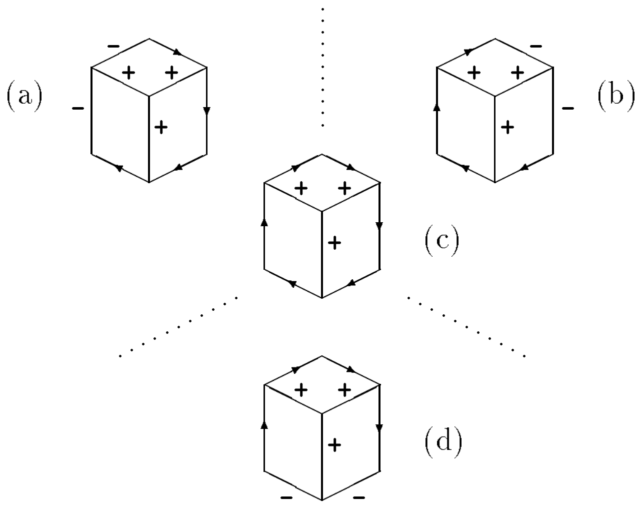

Figure 1.5: Solutions: (a) stuck on left wall, (b) stuck on right wall, (c) suspended in mid-air, (d) resting on floor.

receive the same label. The possible consistent labelings of a cube are presented in Figure 1.5. One formulation of this problem casts the lines in the figure as the variables, with the domains being the possible labelings $\{+, -, \rightarrow, \leftarrow\}$. The constraints will restrict the labelings of adjacent lines in a junction according to the junction types.

1.2 Mathematical foundations

The formalization of constraint networks relies upon concepts drawn from the related areas of discrete mathematics, logic, the theory of relational databases, and graph theory. This section is a compilation of the mathematical knowledge needed for understanding the formalization of constraint networks and the analyses presented in subsequent chapters. Here we present the basic notations and definitions for sets, relations, operations on relations, and graphs. For those readers already familiar with these topics, a skim of this material will suffice to ensure your understanding of notation used in this book. Additionally, some parts of this section (e.g., induced-width, cycle-cutset) may be re-read as they become relevant in later chapters.

1.2.1 Sets, domains, and tuples

A *set* is a collection of distinguishable objects, and an object in the collection is called a *member* or an *element* of the set. A set cannot contain the same object more than once, and its elements are not ordered. If an object x is a member of set A, we write $x \in A$; if an object x is not a member of set A, we write $x \notin A$. A set can be defined explicitly by listing the members of the set or implicitly by stating a property satisfied by elements of the set. For example, $A = \{1, 2, 3\}$ and $A = \{x \mid x \text{ an integer and } 1 \leq x \leq 3\}$ both represent the same set with three members: 1, 2, and 3. If each element of a set A is also an element of set B, then we write $A \subseteq B$ and say that A is *subset* of B. Set A is a *proper subset* of B, written $A \subset B$, if $A \subseteq B$ but $A \neq B$.

Given two sets A and B, we can also define new sets by applying set operations: the *intersection* of two sets A and B is the set $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$, the *union* of two sets A and B is the set $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$, and the *difference* of two sets A and B is the set $A - B = \{x \mid x \in A \text{ and } x \notin B\}$. A set containing no members is called an *empty set* and is denoted $\emptyset$. The number of elements in a set A is called the *size* (or *cardinality*) of the set and is denoted $|A|$. Two sets A and B are *disjoint* if they have no elements in common; that is, if $A \cap B = \emptyset$, which is also to say that $|A \cap B| = 0$.

The *domain* of a variable is simply a set that lists all of the possible objects that a variable can denote or all of the possible values that a variable can be assigned. A *k-tuple* (or simply a *tuple*) is an ordered sequence of k not necessarily distinct objects denoted by $(a_1, \ldots, a_k)$, and an object in the sequence is called a *component*. The *Cartesian product* (or simply the *product*) of a list of domains $D_1, \ldots, D_k$, written $D_1 \times \cdots \times D_k$, is the set of all k-tuples $(a_1, \ldots, a_k)$ such that $a_1 \in D_1$, $a_2 \in D_2$, and so on.

Example 1.1 Let $D_1 = \{\text{black, green}\}$ and $D_2 = \{\text{apple juice, coffee, tea}\}$. The Cartesian product $D_1 \times D_2$ is the set of tuples {(black, apple juice), (black, coffee), (black, tea), (green, apple juice), (green, coffee), (green, tea)}.

1.2.2 Relations

Given a set of variables $X = \{x_1, \ldots, x_k\}$, each associated respectively with a domain $D_1, \ldots, D_k$, a *relation* R on the set of variables is any subset of the Cartesian product of their domains. The set of variables on which a relation is defined is called the *scope* of the relation, denoted $scope(R)$. We will frequently denote a relation defined on a scope S by R_S. Each relation that is a subset of some product $D_1 \times \cdots \times D_k$ of k domains is said to have *arity* k. If $k = 1$, 2, or 3, then the relation is called a *unary*, *binary*, or *ternary* relation, respectively. If $R = D_1 \times \cdots \times D_k$, then R is called a *universal* relation.

Example 1.2 Let $D_1 = \{\text{black, green}\}$ be the domain of variable x_1, and let $D_2 = \{\text{apple juice, coffee, tea}\}$ be the domain of variable x_2. The set of tuples {(black, coffee), (black, tea), (green, tea)} is a relation on $\{x_1, x_2\}$, since the tuples are a subset of the product of D_1 and D_2. The scope of this relation is $\{x_1, x_2\}$.

The empty set is another example of a relation. The scope of the empty relation is itself the empty set.

Representing relations

Relations are sets of tuples defined over the same scope, and, as discussed above for sets, they may be defined either explicitly or implicitly. For example, let R be a relation on the set of variables $\{x_1, x_2\}$, where $D_1 = \{\text{black, green}\}$ and $D_2 = \{\text{apple juice, coffee, tea}\}$. Then the relation R_1 on the scope $\{x_1, x_2\}$ given by $R_1 = \{$(black, coffee), (black, tea), (green, tea)$\}$ and $R_2 = \{(x_1, x_2) \mid x_1 \in D_1, x_2 \in D_2, \text{ and } x_1 \text{ is before } x_2 \text{ in dictionary ordering}\}$ both represent the same relation.

Two additional ways to express a relation are to make use of table or a (0,1)-matrix. In the table representation, each row represents a tuple from the relation, and each column corresponds to one component position and is associated with one variable. (In the database community, the names of columns are called attributes). The ordering of the columns is inconsequential; two relations that differ only in the ordering of their columns are considered to be the same relation.

In the (0,1)-matrix representation, a binary relation between two variables x_i and x_j is represented as a matrix with $|D_i|$ rows and $|D_j|$ columns, where an ordering is imposed on the domains of the variables. A 0-entry at row a, column b means that the pair consisting of the ath element of D_i and the bth element of D_j is not included in the relation; a 1-entry means that the pair is included. The (0,1)-matrix representation can be generalized to nonbinary relations by increasing the number of dimensions of the matrix.

Figure 1.6 shows the table and the (0,1)-matrix representations of a binary relation.

x_1	x_2
black	coffee
black	tea
green	tea

(a) table

$$\begin{array}{ccc} & & \underline{x_2} \\ & & \begin{array}{ccc} \text{apple juice} & \text{coffee} & \text{tea} \end{array} \\ \underline{x_1} & \begin{array}{c} \text{black} \\ \text{green} \end{array} & \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

(b) (0,1)-matrix

Figure 1.6: Two alternative views of relation $R = \{$(black, coffee), (black, tea), (green, tea)$\}$ on the scope $\{x_1, x_2\}$, where $D_1 = \{\text{black, green}\}$ and $D_2 = \{\text{apple juice, coffee, tea}\}$.

Operations on relations

Having introduced the mathematical notion of a relation, we can now consider operations on relations. First we consider how operations defined for all sets apply to relations, and then we discuss three operations specific to relations: selection, projection, and join.

Intersection, union, and difference. Given two relations, R and R', on the same scope, the intersection of R and R', denoted $R \cap R'$, is the relation containing all tuples that are in both R and R'; the union $R \cup R'$ is the relation containing all the tuples that are in either R or R' or both, and the difference $R - R'$ is the relation containing those tuples that are in R but not in R'. The scope of the resulting relations is the same as the scope of the relations R and R'.

x_1	x_2	x_3
a	b	c
b	b	c
c	b	c
c	b	s

(a) Relation R

x_1	x_2	x_3
b	b	c
c	b	c
c	n	n

(b) Relation R'

x_2	x_3	x_4
a	a	1
b	c	2
b	c	3

(c) Relation R''

Figure 1.7: Three relations.

Example 1.3 Let the relations R, R', and R'' be as shown in Figure 1.7. The relations R and R' have the same scopes $\{x_1, x_2, x_3\}$, so the set operations intersection, union, and difference are well defined for these sets. Since the scope of R'' is not the same, none of these three operations is well defined for R'' and either of the other two relations. Figure 1.8 shows the relations $R \cap R'$, $R \cup R'$, and $R - R'$.

x_1	x_2	x_3
b	b	c
c	b	c

(a) $R \cap R'$

x_1	x_2	x_3
a	b	c
b	b	c
c	b	c
c	b	s
c	n	n

(b) $R \cup R'$

x_1	x_2	x_3
a	b	c
c	b	s

(b) $R - R'$

Figure 1.8: Example of set operations intersection, union, and difference applied to relations.

Selection. Let us now consider those operations specific to relations. A selection takes a relation R and yields a new relation which is the subset of tuples of R that have specified values on specified components. That is to say, in a table representation of a relation, selection chooses a subset of the rows of the relation. The scope of the resulting relation is the same as the scope of R. Let R be a relation, let $x_1, \ldots, x_k$ be variables in the scope of R, and let a_i be an element of D_i, the domain of x_i. We use the notation $\sigma_{x_1=a_1,\ldots,x_k=a_k}(R)$ to denote the selection of those tuples in R that have the value a_1 for variable x_1, the value a_2 for variable x_2, and so on. If $Y = (x_1, \ldots, x_k)$ and $t = (a_1, \ldots, a_k)$, then $\sigma_{Y=t}(R)$ is an equivalent alternative and more succinct notation for $\sigma_{x_1=a_1,\ldots,x_k=a_k}(R)$.

Projection. Projection takes a relation R and yields a new relation that consists of the tuples of R with the components in certain positions removed. In the table representation of a relation, projection chooses a subset of the columns of a relation. Let R be a relation, and let $Y = \{x_1, \ldots, x_k\}$ be a subset of the variables in the scope of R. We use the notation $\pi_Y(R)$ to denote the projection of R onto Y, that is, the set of tuples obtained by taking in turn each tuple in R and forming from it a new, smaller tuple, keeping only those components associated with variables in Y. Projection specifies a subset of the variables of a relation, and the scope of the resulting relation is that subset of variables.

Join. The join operator takes two relations, R_S and R_T, and yields a new relation that consists of the tuples of R_S and R_T combined on all their common variables. Let R_S be a relation with scope S and R_T be a relation with scope T. A tuple r is in the join of R_S and R_T, denoted $R_S \bowtie R_T$, if it can be constructed according to the following steps: (i) take a tuple s from R_S, (ii) select a tuple t from R_T such that the components of s and t agree on the variables that R_S and R_T have in common (that is, on the variables in $S \cap T$), and (iii) form a new tuple r by combining the components of s and t, keeping only one copy of those components corresponding to variables in $S \cap T$. The scope of the resulting relation is the union of the scopes of S and T, that is, $S \cup T$. We can see that a join of two relations with the same scopes is equivalent to the intersection of the two relations.

Example 1.4 Let the relations R' and R'' be as shown in Figure 1.7. Figure 1.9 shows examples of the selection, projection, and join operations applied to these relations. The relation $\sigma_{x_3=c}(R')$ consists of those tuples in R' that have the value c for variable x_3. The relation $\pi_{\{x_2,x_3\}}(R')$ consists of the tuples in R', each with the component that corresponds to the variable x_1 removed; only the components that correspond to variables x_2 and x_3 are kept. Duplicate entries are removed, since a relation is a set, and sets do not contain duplicate objects. The relation $R' \bowtie R''$ consists of tuples that are combinations of pairs of tuples from R' and R'' which agree on their common variables $\{x_2, x_3\}$. To construct $R' \bowtie R''$, we consider each tuple in R', match it up with all possible tuples in R'' that agree with it on the common variables, and delete duplicate components associated with these variables. For example, the tuple (b, b, c) in R' agrees with the tuples (b, c, 2) and (b, c, 3) in R'', resulting in the tuples (b, b, c, 2) and (b, b, c, 3). Similarly, the tuple (c, b, c) in R' results in the tuples (c, b, c, 2) and (c, b, c, 3). The tuple (c, n, n) in R' does not agree with any tuple in R'' on variables x_2 and x_3, so no additional tuples are added to $R' \bowtie R''$.

x_1	x_2	x_3
b	b	c
c	b	c

(a) $\sigma_{x_3=c}(R')$

x_2	x_3
b	c
n	n

(b) $\pi_{\{x_2,x_3\}}(R')$

x_1	x_2	x_3	x_4
b	b	c	2
b	b	c	3
c	b	c	2
c	b	c	3

(c) $R' \bowtie R''$

Figure 1.9: Example of selection, projection, and join operations on relations.