Chapter 1

Background Material

Peter Jipsen, Chris Brink[1], *Gunther Schmidt*

This chapter serves the rest of the book: all later chapters presuppose it. It introduces the calculus of binary relations, and relates it to basic concepts and results from lattice theory, universal algebra, category theory and logic. It also fixes the notation and terminology to be used in the rest of the book. Our aim here is to write in a way accessible to readers who desire a gentle introduction to the subject of relational methods. Other readers may prefer to go on to further chapters, only referring back to Chapt. 1 as needed.

1.1 The calculus of sets

Our approach to set theory is, for the most part, informal. We use the capital letters X, Y, Z to denote sets, and lower case x, y, z to denote elements. The symbols "$x \in X$" express the fact that x is an *element* (or *member*) of the *set* X. The *empty set* is the unique set $\emptyset$ that contains no elements. *Singletons* are sets with one element, *unordered pairs* are sets that contain exactly two elements and *unordered n-tuples* $\{x_0, \ldots, x_{n-1}\}$ contain n distinct elements. When convenient, a universal set U may be defined to represent the largest collection under consideration.

Given a set X and a property P, the set of all elements of X that satisfy P is denoted by

$$\{x \in X : P(x)\} \qquad \text{or} \qquad \{x : x \in X \text{ and } P(x)\}.$$

The property P is often described informally, but is understood to be an abbreviation for a precise expression in some formal language.

There are several obvious relations and operations defined on sets:

- *equality:* $X = Y$ if X and Y contain the same elements,
- *inclusion:* $X \subseteq Y$ if every element of X is also an element of Y,
- *union:* $X \cup Y =$ the set of elements in X or Y,
- *intersection:* $X \cap Y =$ the set of elements in both X and Y,
- *difference:* $X - Y =$ the set of elements in X that are not in Y,

[1]Chris Brink gratefully acknowledges the longstanding financial support of the South African Foundation for Research Development.

- *complement:* $\overline{X} = U - X$ provided that a universal set U has been fixed,
- *powerset:* $\mathcal{P}(X) =$ the set of all subsets of X.

Two sets are said to be *disjoint* if their intersection is the empty set.

The expression "calculus of sets" refers to the many interactions between these relations and operations. The main ones are captured by the following observations: For any universal set U and $X, Y, Z \subseteq U$

- $X \subseteq X$ (reflexivity of $\subseteq$),
- $X \subseteq Y$ and $Y \subseteq Z$ imply $X \subseteq Z$ (transitivity of $\subseteq$),
- $X \subseteq Y$ and $Y \subseteq X$ imply $X = Y$ (antisymmetry of $\subseteq$),
- $X \cup Y = Y \cup X$ and $X \cap Y = Y \cap X$ (commutativity of $\cup$, $\cap$),
- $(X \cup Y) \cup Z = X \cup (Y \cup Z)$ and $(X \cap Y) \cap Z = X \cap (Y \cap Z)$ (associativity of $\cup$, $\cap$),
- $X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$ and $X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$ (distributivity of $\cup$, $\cap$) and
- $X \cup \overline{X} = U$ and $X \cap \overline{X} = \emptyset$.

Readers familiar with mathematical terminology may note that the first three properties show $\subseteq$ is a partial order on $\mathcal{P}(U)$, and the remaining ones imply that $(\mathcal{P}(U), \cup, \cap, \overline{}, \emptyset, U)$ is a Boolean algebra.

If I is any set, and $\mathcal{X} = \{X_i : i \in I\}$ is a collection of sets indexed by I then the union and intersection of $\mathcal{X}$ are defined by

$$\bigcup \mathcal{X} \triangleq \bigcup_{i \in I} X_i \triangleq \{x : x \in X_i \text{ for some } i \in I\}^2,$$
$$\bigcap \mathcal{X} \triangleq \bigcap_{i \in I} X_i \triangleq \{x : x \in X_i \text{ for all } i \in I\}.$$

From objects x and y we obtain the *ordered pair* or simply *pair* (x, y) with the characteristic property that $(x, y) = (x', y')$ if and only if $x = x'$ and $y = y'$. *Sequences of length* n or n*-tuples* $(x_0, \ldots, x_{n-1})$ are characterized by the analogous property. The *Cartesian product* of sets $X_0, \ldots, X_{n-1}$ is given by

$$X_0 \times \ldots \times X_{n-1} = \prod_{i=0}^{n-1} X_i \triangleq \{(x_0, \ldots, x_{n-1}) : x_i \in X_i \text{ for all } i < n\}.$$

If all the sets X_i are equal to X, we write X^n instead of $\prod_{i=0}^{n-1} X$.

We also fix the following notation for various standard sets:

- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$, the set of natural numbers.
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$, the set of integers.
- $\mathbb{Q} = \{\frac{m}{n} : m \in \mathbb{Z} \text{ and } 0 \neq n \in \mathbb{N}\}$, the set of rationals.
- $\mathbb{R}$ the set of reals.
- $\mathbb{B} = \{\mathsf{true}, \mathsf{false}\}$, the set of booleans.

[2]The symbol $\triangleq$ signifies that the equalities hold by definition.

1.2 The calculus of binary relations

Informally, a binary relation is simply a collection of ordered pairs. More precisely, a *binary relation* R *from a set* X *to a set* Y is a subset of the set of all pairs (x, y) where $x \in X$ and $y \in Y$. In symbols

$$R \subseteq X \times Y.$$

If $X = Y$, we say that R is a *relation over* X. Instead of $(x, y) \in R$, we usually write xRy. By virtue of being sets, binary relations are partially ordered by inclusion. The smallest relation is just the empty set $\emptyset$. Depending on the context, we may also fix a largest relation, called the *universal relation*, which is denoted by V (for example $V = U^2$). Another special relation defined for each set X is the *identity relation*

$$I_X = \{(x, x) : x \in X\}.$$

Since relations are sets, all the set-theoretic operations apply. However, relations are more than just sets, and the structure of their elements allows the definition of many other operations. The most common ones are

- *domain:* $\mathit{dom}R = \{x : \text{there exists } y \text{ such that } xRy\}$,
- *range:* $\mathit{ran}R = \{y : \text{there exists } x \text{ such that } xRy\}$,
- *converse:* $R^{\smile} = \{(x, y) : yRx\}$,
- *composition:* $R;S = \{(x, y) : \text{there exists } z \text{ such that } xRz \text{ and } zSy\}$,
- *right residual:* $R \backslash S = \{(x, y) : \text{for all } z,\ zRx \text{ implies } zSy\}$,
- *left residual:* $R/S = \{(x, y) : \text{for all } z,\ ySz \text{ implies } xRz\}$,
- *Peirce product:* $R : Y = \{x : \text{there exists } y \in Y \text{ such that } xRy\}$,
- *image set:* $R(x) = \{y : xRy\}$,
- *exponentiation:* $R^0 = I_U$ where U is a fixed universal set, and $R^{n+1} = R;R^n$.

One of the strengths of the relational calculus is that many properties can be expressed very compactly. This is demonstrated throughout this book, beginning with the list below of common conditions used to classify relations. A relation R is said to be

- *reflexive* if $I_U \subseteq R$ (i.e., xRx for all $x \in U$),
- *transitive* if $R;R \subseteq R$ (i.e., xRy and yRz imply xRz for all $x, y, z \in U$),
- *symmetric* if $R = R^{\smile}$ (i.e., xRy implies yRx for all $x, y \in U$),
- *antisymmetric* if $R \cap R^{\smile} \subseteq I_U$ (i.e., xRy and yRx imply $x = y$),
- a *preorder* if it is reflexive and transitive (also called a *quasi-order*),
- an *equivalence relation* if it is a symmetric preorder,
- a *partial order* if it is an antisymmetric preorder.

Since the intersection of transitive relations is again transitive, any relation R is contained in a smallest transitive relation, called the *transitive closure* of a relation

R. This relation can be defined directly by $R^+ = \bigcap\{T : R \subseteq T \text{ and } T;T \subseteq T\}$, and it is easy to prove that

$$R^+ = \bigcup_{i \in \mathbb{N}} R^i = R \cup R^2 \cup R^3 \cup \cdots$$

The closely related *reflexive transitive closure* of R is defined as $R^* = I_U \cup R^+$. Both operations are of particular interest to computer science since they can model the behaviour of programs with loops.

When several operations appear in the same expression, parentheses are used to indicate the order in which the operations are performed. To avoid proliferation of parentheses the following convention is adopted:

Priority of operations: Unary (superscript) operations ($\overline{\ \ }$, $\breve{\ }$, $^+$, *, ^{n}) are performed first, followed by the binary relation operations ($;$, $:$, $/$, $\backslash$) and finally the binary set operations ($\cup$, $\cap$).

Algebraic properties of relation operations

The most obvious properties of operations defined on relations are listed below.

- Composition is *associative*: $(R;S);T = R;(S;T)$.
- I_U is an *identity* for the composition of relations on U: $I_U;R = R = R;I_U$.
- Composition *distributes* over union: $R;(S \cup T) = R;S \cup R;T$ and $(R \cup S);T = R;T \cup S;T$.
- Conversion *distributes* over union: $(R \cup S)^\smile = R^\smile \cup S^\smile$.
- Conversion is an *involution*: $(R^\smile)^\smile = R$.
- Conversion *antidistributes* over composition: $(R;S)^\smile = S^\smile;R^\smile$.
- Composition satisfies what are known as the *Schröder equivalences* (which will later be related to *right-* and *left conjugates*):

$$R;S \cap T = \emptyset \iff R^\smile;T \cap S = \emptyset \iff T;S^\smile \cap R = \emptyset\ .$$

- The reflexive transitive closure satisfies

$$R^* = I_U \cup R;R^* \qquad \text{and} \qquad R;S \subseteq S \implies R^*;S \subseteq S\ .$$

Many other relationships hold, but the ones mentioned above have the distinction that they form the basis for an abstract treatment of relations.

Viewing relations as sets of ordered pairs is appealing in its simplicity, but for many applications in computer science it is more useful to "type" a relation $R \subseteq X \times Y$ by explicitly recording its *source* X and *target* Y. So we define a *typed relation from* X *to* Y to be a triple (R, X, Y), where $R \subseteq X \times Y$. The set of all typed relations from X to Y is denoted by $[X \leftrightarrow Y]$, and instead of $T \in [X \leftrightarrow Y]$ we also write $T : X \leftrightarrow Y$. When working only with typed relations, the adjective "typed" is usually omitted. All the operations and properties of (untyped) relations apply here as well, with the understanding that conversion interchanges the source and target, and the binary operations are only defined when the sources and targets of the respective relations are compatible. For the operations $\cup$, $\cap$ the relations must have the same sources and the same targets, and for the other operations, if $R : X \leftrightarrow Y$, $S : Y \leftrightarrow Z$ and $T : X \leftrightarrow Z$ then

$$R^\smile : Y \leftrightarrow X, \quad R;S : X \leftrightarrow Z, \quad R\backslash T : Y \leftrightarrow Z \text{ and } T/S : X \leftrightarrow Y.$$

The additional structure of typed relations leads to further useful definitions. A relation $R : X \leftrightarrow Y$ is said to be

- *univalent* if $R^\smile;R \subseteq I_Y$ (i.e. xRy and xRy' imply $y = y'$ for all x, y, y'),
- *total* if $I_X \subseteq R;R^\smile$ (i.e. for all $x \in X$ there exists $y \in Y$ such that xRy),
- a *function* if it is univalent and total,
- *injective* if $R;R^\smile \subseteq I_X$ (i.e. xRy and $x'Ry$ imply $x = x'$ for all x, x', y),
- *surjective* if $I_Y \subseteq R^\smile;R$ (i.e. for all $y \in Y$ there exists $x \in X$ such that xRy),
- a *bijection* if it is an injective and surjective function (i.e. $R;R^\smile = I_X$ and $R^\smile;R = I_Y$).

If $R : X \leftrightarrow Y$ is a function, we write $R : X \to Y$. If R is not total but univalent then it is called a *partial function.*

In either case we will normally use the symbols f, g, h instead of relational symbols, and the notation "$y = f(x)$" instead of "xfy". For two functions $f : X \to Y$ and $g : Y \to Z$, the *composite function* $g \circ f$ is defined by $g \circ f = f;g$, with the result that $(g \circ f)(x) = g(f(x))$.

The notion of "binary relation" can obviously be generalized to higher dimensions: an n*-ary relation based on sets* $X_0, \ldots, X_{n-1}$ is simply a subset of the n-ary Cartesian product $X_0 \times \ldots \times X_{n-1}$. Although less pervasive than binary relations, these generalizations arise naturally in logic, algebra, database theory and many other areas. In the above definition, the elements of an n-ary relation are n-tuples indexed by the set $H = \{0, \ldots, n-1\}$. In applications to computer science it is often convenient to allow arbitrary index sets. For example, if the 3-tuple $t =$ (Smith, Jane, 1234567) represents an entry in a phonebook, a suitable index set would be $H = \{\text{lastname, firstname, phonenumber}\}$. Such descriptive indices are also called *attributes* in database theory. Associated with each index i is a *domain of values* $\mathcal{D}(i)$, e.g. $\mathcal{D}(\text{lastname})$ would be the collection of lastnames valid for a phonebook.

An H*-tuple* t is now simply a function from H to $\bigcup_{i \in H} \mathcal{D}(i)$ such that $t(i) \in \mathcal{D}(i)$, and the set of all H-tuples is the Cartesian product $\prod_{i \in H} \mathcal{D}(i)$. A relation with index set H is a subset of this Cartesian product. Note that if $H = \{0, \ldots, n-1\}$ then an H-tuple is just an n-tuple and a relation with index set H is an n-ary relation.

Given a Cartesian product $\prod_{i \in H} X_i$ there is for each $i \in H$ a *projection function*

$$\pi_i : \prod_{i \in H} X_i \to X_i \quad \text{defined by} \quad \pi_i(t) \triangleq t(i).$$

For a subset J of H, each H-tuple t gives rise to a J-tuple $t[J]$ defined by restricting the function t to values from J. This provides a generalized projection function $\pi_J : \prod_{i \in H} X_i \to \prod_{j \in J} X_j$ given by $\pi_J(t) \triangleq t[J]$.

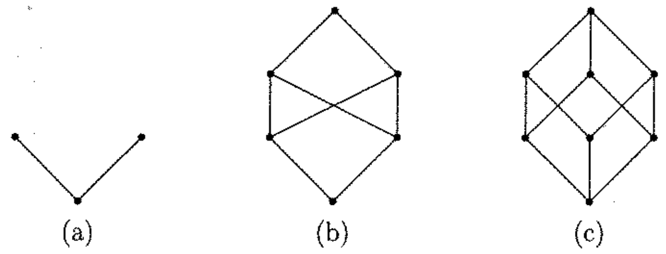

Fig. 1.1 Some posets and their Hasse diagrams

Using the above notions, any relation over an index set H can be decomposed into a binary relation. A particular decomposition is determined by a subset J of H, and each H-tuple t is mapped to the pair $(t[J], t[\overline{J}])$, where complementation is with respect to H. Conversely, given disjoint sets J, K and a binary relation in which all pairs have J-tuples as first component and K-tuples as second component, each pair (x, y) can be mapped to a unique $J \cup K$-tuple t defined by $t[J] = x$ and $t[K] = y$.

1.3 Partially ordered structures

As mentioned in Sect. 1.2, a binary relation R on a set X is a *partial order* if it is reflexive, transitive and antisymmetric. The pair (X, R) is called a *partially ordered set* or *poset*. Usually "R" is replaced with a more suggestive symbol "$\sqsubseteq$" (read "less-or-equal"), in which case $R^{\smile}$ is written $\sqsupseteq$ (read "greater-or-equal"). If X has only a few elements then it is quite instructive to represent $(X, \sqsubseteq)$ by a *Hasse diagram* where an element x is connected by a straight line to a distinct element y higher up on the page if and only if $x \sqsubseteq y$ and there exists no $z \in X$ distinct from x and y with $x \sqsubseteq z \sqsubseteq y$. Some examples of posets and their Hasse diagrams are given in Fig. 1.1.

In a poset $(X, \sqsubseteq)$ an element x is an *upper bound* of a subset Y of X if $x \sqsupseteq y$ for all $y \in Y$. The *least upper bound* of Y, also called the *join* of Y and denoted by $\bigsqcup Y$, is the (unique) upper bound that is less than every other upper bound of Y. Of course upper bounds and least upper bounds don't necessarily exist for all subsets (see Fig. 1.1 (a), (b)). In particular, since every element is an upper bound of the empty set, $\bigsqcup \emptyset$ exists if and only if $(X, \sqsubseteq)$ has a least element, denoted by $\bot$ (read "bottom"). The largest element of a poset, if it exists, is denoted by $\top$ (read "top").

A nonempty subset C of X is a *chain* if it is linearly ordered, i.e. $x \sqsubseteq y$ or $y \sqsubseteq x$ for all $x, y \in C$. This definition allows us to define a *complete partial order* (*cpo* for short) as a poset $(X, \sqsubseteq)$ in which every chain has a least upper bound. In computer science cpos are mainly used as models of denotational semantics for programs. A poset endowed with extra structure is naturally thought of as a *partially ordered structure*. Algebraic versions of the calculus of sets and the calculus of relations are in a natural way represented as such structures.

Lattices

The *greatest lower bound* or *meet* of a subset Y in a poset $(X, \sqsubseteq)$ is denoted by $\bigsqcap Y$, and can be defined as the least upper bound of Y in the *dual poset* $(X, \sqsupseteq)$. A *lattice* is a nonempty poset in which every two-element subset $\{x, y\}$ has a join $x \sqcup y = \bigsqcup\{x, y\}$ and a meet $x \sqcap y = \bigsqcap\{x, y\}$. These two operations are

- *associative:* $(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$ and $(x \sqcap y) \sqcap z = x \sqcap (y \sqcap z)$,
- *commutative:* $x \sqcup y = y \sqcup x$ and $x \sqcap y = y \sqcap x$,
- *idempotent:* $x \sqcup x = x$ and $x \sqcap x = x$, and satisfy the
- *absorptive laws:* $(x \sqcup y) \sqcap x = x$ and $(x \sqcap y) \sqcup x = x$.

The equational properties listed above characterize lattices in the following sense: Given a set X with two binary operations $\sqcup$ and $\sqcap$ that satisfy the above equations for all $x, y, z \in X$, define a relation $\sqsubseteq$ on X by

$$x \sqsubseteq y \iff x \sqcup y = y.$$

Then $(X, \sqsubseteq)$ turns out to be a lattice in which the join and meet operations agree with $\sqcup$ and $\sqcap$. Hence a lattice may be viewed as either a poset $(X, \sqsubseteq)$ or an algebraic structure $(X, \sqcup, \sqcap)$. It is easy to give examples of a set of sets closed under union and intersection; this would be an example of a lattice.

In a lattice the least upper bound of any finite nonempty subset can be found by repeated application of the join operation. A lattice $(X, \sqsubseteq)$ is said to be

- *bounded* if it has a smallest element $\bot$ $(= \bigsqcup \emptyset = \bigsqcap X)$ and a largest element $\top$ $(= \bigsqcup X = \bigsqcap \emptyset)$,
- *complete* if every subset Y has a join $\bigsqcup Y$. In that case Y also has a meet $\bigsqcap Y = \bigsqcup\{\text{all lower bounds of } Y\}$,
- *complemented* if it is bounded and every $x \in X$ has a *complement* $\overline{x}$ such that $x \sqcup \overline{x} = \top$ and $x \sqcap \overline{x} = \bot$,
- *distributive* if $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$ and $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$ for all $x, y, z \in X$.

We note that in a distributive lattice the complement of an element is unique (whenever it exists).

Boolean algebras

With the above terminology, a *Boolean algebra* is defined as a complemented distributive lattice $\mathcal{A} = (A, \sqcup, \sqcap, \overline{}, \bot, \top)$. Since it can be proved that Boolean algebras satisfy *De Morgan's laws* $\overline{x \sqcup y} = \overline{x} \sqcap \overline{y}$ and $\overline{x \sqcap y} = \overline{x} \sqcup \overline{y}$, it is actually enough to list only the operations $\sqcup$, $\overline{}$ and $\bot$. The others are recovered as

$$x \sqcap y = \overline{\overline{x} \sqcup \overline{y}} \qquad \text{and} \qquad \top = \overline{\bot}.$$

The theory of Boolean algebras is an abstract algebraic version of the calculus of sets (Sect. 1.1) and propositional logic (Sect. 1.6). Standard examples of Boolean algebras are the collection of all subsets of a universal set U:

$$\mathcal{B} = (\mathcal{P}(U), \cup, \overline{}, \emptyset).$$

Figure 1.1 (c) shows the Hasse diagram of such an algebra when U is a 3-element set. Other examples are obtained by considering subcollections $C \subseteq \mathcal{P}(U)$ with the property that $\emptyset \in C$ and for all $X, Y \in C$ we also have both $X \cup Y \in C$ and $\overline{X} \in C$. Then $\mathcal{C} = (C, \cup, \overline{}, \emptyset)$ is also a Boolean algebra[3], referred to as a *subalgebra* of $\mathcal{B}$. In fact, by a fundamental result of Stone, dating back to 1935, every Boolean algebra can be obtained in this way (up to isomorphism).

To make this precise, we need a few definitions. An *atom* in a Boolean algebra is a minimal non-$\perp\!\!\!\perp$ element. In the algebra $\mathcal{B}$ above, the atoms are the singleton subsets of U. A subset F of a Boolean algebra is

- *meet-closed* if $x, y \in F$ implies $x \sqcap y \in F$,
- an *upset* if $x \in F$ and $x \sqsubseteq y$ imply $y \in F$,
- a *filter* if it is a meet-closed upset,
- an *ultrafilter* if it is a filter, $\perp\!\!\!\perp \notin F$ and for all x in the algebra either $x \in F$ or $\overline{x} \in F$.

The last condition is equivalent to saying that F is a maximal proper filter. For example, given an atom a in a Boolean algebra, there is exactly one ultrafilter F containing a, namely $F = \{x : a \sqsubseteq x\}$. Now the ultrafilters of a Boolean algebra $\mathcal{A}$ can serve as atoms of another algebra.

Theorem 1.3.1 (*Stone's Representation Theorem*) Let $\mathcal{U}_\mathcal{A}$ be the set of all ultrafilters of $\mathcal{A}$. The function $\sigma : \mathcal{A} \to \mathcal{P}(\mathcal{U}_\mathcal{A})$ defined by

$$\sigma(x) \triangleq \{F \in \mathcal{U}_\mathcal{A} : x \in F\}$$

is a Boolean algebra embedding, i.e. $\sigma(x \sqcup y) = \sigma(x) \cup \sigma(y)$, $\sigma(\overline{x}) = \overline{\sigma(x)}$, $\sigma(\perp\!\!\!\perp) = \emptyset$ and σ is injective. □

Relation algebras

Here we define abstract algebraic structures that capture many of the algebraic properties of relations. They are based on Boolean algebras augmented by the operations ; (binary *relation composition*) and $\breve{}$ (unary *converse*), and a distinguished element $\mathbb{I}$ (*identity*).

An *(abstract) relation algebra* is of the form $(A, \sqcup, \overline{}, \perp\!\!\!\perp, ;, \breve{}, \mathbb{I})$ where

- $(A, \sqcup, \overline{}, \perp\!\!\!\perp)$ is a Boolean algebra,
- ; is associative and distributes over $\sqcup$,
- $\breve{}$ is an involution, distributes over $\sqcup$ and antidistributes over ;,
- $\mathbb{I}$ is an identity for ; and
- $x ; \overline{x^{\smile} ; \overline{y}} \sqsubseteq y$ for all $x, y \in A$.

The last-mentioned inequality states that $z = \overline{x^{\smile} ; \overline{y}}$ is a solution of $x ; z \sqsubseteq y$. In the presence of the first three properties, this is in fact the largest solution. Moreover, the inequality is equivalent to the claim that, for all $x, y, z \in A$:

$$x ; y \sqcap z = \perp\!\!\!\perp \iff x^{\smile} ; z \sqcap y = \perp\!\!\!\perp \iff z ; y^{\smile} \sqcap x = \perp\!\!\!\perp .$$

[3]Since the axioms for lattices, distributivity and complementation are equational.

These are exactly the Schröder equivalences, and hence the properties listed here for abstract relations are the same as for relations in Sect. 1.2 (under the subheading "Algebraic properties of binary relations"). A standard example of a relation algebra is the set of all relations on a universe U, called the *full relation algebra over* U:

$$\mathit{Rel}(U) = (\mathcal{P}(U^2), \cup, \overline{}, \emptyset, ;, \breve{}, I_U) .$$

Boolean modules and Peirce algebras

Observe that in a relation algebra the nonboolean operations ;, $\breve{}$ and $\mathbb{I}$ distribute in each argument over the Boolean join $\sqcup$.[4] Such operations are called operators, and algebras of this type are collectively referred to as *Boolean algebras with operators* or BAOs for short. An operator is said to be *normal* if it has value $\perp\!\!\!\perp$ whenever one of the arguments is $\perp\!\!\!\perp$. A *normal* BAO is one in which every operator is normal. It is easy to show that relation algebras are normal BAOs.

Let $\mathcal{A}$ be a relation algebra. A (left) *Boolean $\mathcal{A}$-module* is a Boolean algebra $\mathcal{B} = (B, \sqcup, \overline{}, \perp\!\!\!\perp)$ together with a mapping $f : A \times B \to B$, where $f(r, x)$ is written $r : x$ and called the *Peirce product*, such that for all $r, s \in A$ and $x, y \in B$:

- $r : (x \sqcup y) = r : x \sqcup r : y$,
- $(r \sqcup s) : x = r : x \sqcup s : x$,
- $(r ; s) : x = r : (s : x)$,
- $\mathbb{I} : x = x$,
- $\perp\!\!\!\perp : x = \perp\!\!\!\perp$,
- $r^{\smile} : \overline{r : x} \sqsubseteq \overline{x}$.

To see that Boolean modules are a type of BAO, one can define for each $r \in A$ a unary operation $f_r : B \to B$ by $f_r(x) = r : x$. Then the first equation above states that each f_r is an operator. For a concrete example of a Boolean module, consider $\mathcal{A} = \mathit{Rel}(U)$, $\mathcal{B} = (\mathcal{P}(U), \cup, \overline{}, \emptyset)$ and $R : X$ is the Peirce product defined in Sect. 1.2. Thus, whereas relation algebras are an algebraic version of relations acting on each other, Boolean modules are an algebraic version of relations acting on sets.

It is occasionally useful to consider also relation-forming operations on sets. One such is the *cylindrification*, which associates with any set X contained in some universe U the relation $X^c = \{(x, u) : x \in X \text{ and } u \in U\}$. A *Peirce algebra* $(\mathcal{A}, \mathcal{B})$ is a Boolean $\mathcal{A}$-module $\mathcal{B}$ with an additional unary (postfix) operation ${}^c : \mathcal{B} \to \mathcal{A}$, such that for all $r \in \mathcal{A}$ and $x \in \mathcal{B}$

$$x^c : \mathbb{T} = x \quad \text{and} \quad (r : \mathbb{T})^c = r ; \mathbb{T} .$$

The concrete example above can be expanded to a Peirce algebra by defining X^c to be the relation $X \times U$.

[4]The constant $\mathbb{I}$ does so vacuously, since it has no arguments.

Residuation and conjugation

When dealing with operations on partial orders, there are several recurring concepts that are simple, but important.

Let $(X, \sqsubseteq)$ and $(Y, \sqsubseteq)$ be two partially ordered sets. A map $f : X \to Y$ is

- *order-preserving* (or *isotonic*) if $x \sqsubseteq x'$ implies $f(x) \sqsubseteq f(x')$,
- *join-preserving* if $f(x \sqcup y) = f(x) \sqcup f(y)$ whenever $x \sqcup y$ exists in X,
- *completely join-preserving* if $f(\bigsqcup S) = \bigsqcup\{f(s) : s \in S\}$ whenever $\bigsqcup S$ exists in X,
- *residuated* if there is a map $g : Y \to X$, called the *residual of* f, such that

$$f(x) \sqsubseteq y \iff x \sqsubseteq g(y) \qquad \text{for all } x \in X,\ y \in Y.$$

A residuated map f and its residual g are also referred to as a *Galois connection* or *adjunction*. In the latter case f is the *lower adjoint* (of g), and g is the *upper adjoint* (of f)[5].

A simple example of a residuated map is the function $f(x) = x^\smile$, defined on any relation algebra. The residual of f is in fact f itself, since the statements $x^\smile \sqsubseteq y$ and $x \sqsubseteq y^\smile$ are equivalent. (For residuals of relations see below.)

The notion of a *(completely) meet-preserving* map is defined dually by interchanging $\sqcup$ with $\sqcap$, and $\bigsqcup$ with $\bigsqcap$. The following result implies that the properties above are listed in increasing order of strength.

Theorem 1.3.2 Let $(X, \sqsubseteq)$ and $(Y, \sqsubseteq)$ be posets. If $f : X \to Y$ is residuated, then it is completely join-preserving and hence order-preserving. Furthermore, the residual g is unique, completely meet-preserving, and is given by

$$g(y) = \bigsqcup\{x \in X : f(x) \sqsubseteq y\}.$$

For a partial converse, if $(X, \sqsubseteq)$ is a complete lattice and f is completely join-preserving then f is residuated. □

For a map $f : A \to B$, where $(A, \sqcup, \overline{}, \perp\!\!\!\perp)$ and $(B, \sqcup, \overline{}, \perp\!\!\!\perp)$ are Boolean algebras, the *dual of* f is defined by $f^d(x) = \overline{f(\overline{x})}$. Furthermore, f is said to be *conjugated* if there is a map $h : B \to A$, called the *conjugate of* f, such that

$$f(x) \sqcap y = \perp\!\!\!\perp \iff x \sqcap h(y) = \perp\!\!\!\perp \qquad \text{for all } x \in X,\ y \in Y.$$

(Or, equivalently,

$$f(x) \sqsubseteq \overline{y} \iff x \sqsubseteq \overline{h(y)} \qquad \text{for all } x \in X,\ y \in Y.$$

In this form it may be compared to the definition above of a residuated map.) A natural example of a conjugated map is the function $f_a(x) = a ; x$, defined for any element a in a relation algebra. The conjugate map is $h_a(y) = a^\smile ; y$, since the expressions $a ; x \sqcap y = \perp\!\!\!\perp$ and $a^\smile ; y \sqcap x = \perp\!\!\!\perp$ are equivalent. In fact, in this case the conjugate map is usually called the *right* conjugate. If we reverse the order of the composition, and define the map $(f')_a(x) = x ; a$, then the conjugate map (which will now be called the *left* conjugate) is $(h')_a(y) = y ; a^\smile$, since the expressions $x ; a \sqcap y = \perp\!\!\!\perp$ and $y ; a^\smile \sqcap x = \perp\!\!\!\perp$ are equivalent. This explains the use of the word "conjugate" in connection with the Schröder equivalences (under "algebraic properties of relation operations" in Sect. 1.2).

For maps between Boolean algebras, residuals and conjugates are duals of each other (i.e. $g = h^d$), so a map is conjugated if and only if it is residuated. (However, conjugation has the advantage of being a symmetric property: if h is the conjugate of f, then f is the conjugate of h.) In particular the residual g_a of the function f_a above is given by $(h_a)^d(y) = \overline{h_a(\overline{y})} = \overline{a^\smile ; \overline{y}}$. It is easy to check that this satisfies the definition of a residual: the expressions $f_a(x) \sqsubseteq y$ and $x \sqsubseteq g_a(y)$ are equivalent. Again, in this case the residual is usually called the *right* residual, and the *left* residual will be given by the dual of the left conjugate: $((h')_a)^d(y) = \overline{(h')_a(\overline{y})} = \overline{\overline{y} ; a^\smile}$. It is easy to verify that for binary relations these characterisations of right and left residual coincide with those given in Sect. 1.2.

[5]In categorical terms f is the left adjoint (of g), and g is the right adjoint (of f). To avoid confusion with left and right residuals of relations, we refrain from this terminology.

Fixed points of order-preserving maps

Given a recursively defined function such as the factorial function g, defined by $g(0) = 1$ and $g(n+1) = (n+1) \cdot g(n)$, one can legitimately ask what object represents g. One solution is to view the recursion at a higher level: for every (partial) function $x : \mathrm{N} \to \mathrm{N}$, define a (partial) function $f(x)$ (often called a *functional* since it maps functions to functions) by the (nonrecursive) equations

$$f(x)(0) = 1 \quad \text{and} \quad f(x)(n+1) = (n+1) \cdot x(n)$$

and look for a solution of the equation $f(x) = x$. Any such solution is called a *fixed point* of f. In this example it can be shown that there is exactly one solution, namely $x(n) = n!$, but for other recursive equations there may be many or no solutions.

Recursively defined functions are prominent in computer science and, as in the above example, the meaning of a recursive definition can be viewed as a fixed point of a related function(al). So it is important to find general conditions under which a function is guaranteed to have fixed points.

Knaster and Tarski (1927) proved that, for any universe U, an inclusion-preserving function $f : \mathcal{P}(U) \to \mathcal{P}(U)$ has at least one fixed point. This result was generalized by Tarski to arbitrary complete lattices.

Theorem 1.3.3 (*Tarski's Fixed Point Theorem, 1955*) Let $(X, \sqsubseteq)$ be a complete lattice and suppose $f : X \to X$ is order-preserving. Then the set $F = \{x \in X : f(x) = x\}$ is nonempty. In fact $(F, \sqsubseteq)$ is a complete lattice[6] with least element $\bigsqcap F = \bigsqcap\{x \in X : f(x) \sqsubseteq x\}$ and largest element $\bigsqcup F = \bigsqcup\{x \in X : x \sqsubseteq f(x)\}$. □

A similar result holds for complete partial orders. The least element of $(F, \sqsubseteq)$ is (by definition) the *least fixed point* of f, denoted by μf. The *greatest fixed point* of f is $\nu f = \bigsqcup F$. When a recursive definition has several solutions, it is often the least fixed point that is of interest. The preceding result implies that μf is characterized by the conditions

[6]Note that in general $(F, \sqsubseteq)$ is not a sublattice of $(X, \sqsubseteq)$.

- $f(\mu f) = \mu f$ (computation) and
- $f(x) \sqsubseteq x \implies \mu f \sqsubseteq x$ (induction).

1.4 Relational structures and algebras

Posets and Boolean algebras are specific examples of relational structures and (universal) algebras. To define these concepts in general, we first need the notion of a *(similarity) type*, which is a function $\tau : \mathcal{F}_\tau \cup \mathcal{R}_\tau \to \{0, 1, 2, \ldots\}$ where $\mathcal{F}_\tau$ is a set of *function symbols* and $\mathcal{R}_\tau$ is a disjoint set of *relation symbols* or *predicate symbols*. For a symbol $s \in \mathcal{F}_\tau \cup \mathcal{R}_\tau$ we say that s has *arity* $\tau(s)$. A relational structure of type τ is of the form $\mathcal{U} = (U, (f^{\mathcal{U}})_{f \in \mathcal{F}_\tau}, (R^{\mathcal{U}})_{R \in \mathcal{R}_\tau})$, where

- U is a set called the *universe* of $\mathcal{U}$,
- $f^{\mathcal{U}}$ is a $\tau(f)$-ary operation on U, i.e. $f^{\mathcal{U}} : U^{\tau(f)} \to U$ and
- $R^{\mathcal{U}}$ is a $\tau(R)$-ary relation on U, i.e. $R^{\mathcal{U}} \subseteq U^{\tau(R)}$.

The distinction between the symbol $s \in \mathcal{F}_\tau \cup \mathcal{R}_\tau$ and its *interpretation* $s^{\mathcal{U}}$ is important, even though the superscript is often omitted in a context where confusion is unlikely. A 0-ary operation $c^{\mathcal{U}}$ from $U^0 = \{\emptyset\}$ to U is also called a *constant*, and $c^{\mathcal{U}}$ is usually identified with the value $c^{\mathcal{U}}(\emptyset)$.

If the set of relation symbols is empty then we say that τ is an *algebraic type*. A *(universal) algebra* is a relational structure of such a type. For example a Boolean algebra has an algebraic type $\tau = \{(\sqcup, 2), (\bar{\ }, 1), (\perp, 0)\}$ and a relation algebra has an algebraic type $\tau' = \tau \cup \{(;, 2), (\breve{\ }, 1), (\mathbb{I}, 0)\}$. Some other algebras mentioned in later chapters are

- *semigroups* $(S, *)$ where $*$ is an associative binary operation,
- *monoids* $(M, *, e)$ where $(M, *)$ is a semigroup and e is an identity element,
- *groups* $(G, *, ^{-1}, e)$ where $(G, *, e)$ is a monoid and $^{-1}$ is a unary inverse operation $(x * x^{-1} = e = x^{-1} * x)$,
- *semilattices* $(S, \sqcup)$, where $\sqcup$ is an associative, commutative and idempotent binary operation.

Given a set V, the set of *terms of type* τ *with variables from* V is defined as the smallest set $T_\tau(V)$ such that

- $V \subseteq T_\tau(V)$ and
- if $t_0, \ldots, t_{n-1} \in T_\tau(V)$, $f \in \mathcal{F}_\tau$ and $n = \tau(f)$ then the (uninterpreted) string of symbols $f(t_0, \ldots, t_{n-1})$ is in $T_\tau(V)$.

This set is the universe of an algebra $\mathcal{T}_\tau(V) = \mathcal{T} = (T_\tau(V), (f^{\mathcal{T}})_{f \in \mathcal{F}_\tau})$, called the *(absolutely free) term algebra of type* τ *generated by* V, with the operations $f^{\mathcal{T}}$ given by

$$f^{\mathcal{T}}(t_0, \ldots, t_{n-1}) = f(t_0, \ldots, t_{n-1}) \quad \text{for } t_i \in T_\tau(V),\ i < n = \tau(f).$$

The terms in $T_\tau(\emptyset)$ are known as *initial (or ground) terms* and $\mathcal{T}_\tau(\emptyset)$ is called the *initial algebra of type* τ.

For example the initial algebra of type $\{(z, 0), (s, 1)\}$ is the Peano algebra on which the arithmetic of the natural numbers is based $(0 = z,\ 1 = s(z),\ 2 = s(s(z)), \ldots)$.

Let $\mathcal{A}, \mathcal{B}, \mathcal{B}_i$ $(i \in I)$ be algebras of type τ.

- $\mathcal{A}$ is a *subalgebra* of $\mathcal{B}$ if $A \subseteq B$ and $f^{\mathcal{A}}(a_0, \ldots, a_{n-1}) = f^{\mathcal{B}}(a_0, \ldots, a_{n-1})$ for all $a_j \in A$, $j < n = \tau(f)$ and all $f \in \mathcal{F}_\tau$.
- $h : \mathcal{A} \to \mathcal{B}$ is a *homomorphism* if h is a function from A to B and $h(f^{\mathcal{A}}(a_0, \ldots, a_{n-1})) = f^{\mathcal{B}}(h(a_0), \ldots, h(a_{n-1}))$ for all $a_j \in A$, $j < n = \tau(f)$ and all $f \in \mathcal{F}_\tau$.
- $\mathcal{B}$ is a *homomorphic image* of $\mathcal{A}$ if there exists a surjective homomorphism from $\mathcal{A}$ to $\mathcal{B}$.
- $\mathcal{A}$ is *isomorphic* to $\mathcal{B}$, in symbols $\mathcal{A} \cong \mathcal{B}$, if there exists a bijective homomorphism from $\mathcal{A}$ to $\mathcal{B}$.
- $\mathcal{A} = \prod_{i \in I} \mathcal{B}_i$, the *product* of algebras $\mathcal{B}_i$, if $A = \prod_{i \in I} B_i$ and $f^{\mathcal{A}}$ is defined coordinatewise by
$$\pi_i(f^{\mathcal{A}}(a_0, \ldots, a_{n-1})) = f^{\mathcal{B}_i}(\pi_i(a_0), \ldots, \pi_i(a_{n-1}))$$
for all $a_j \in A$, $j < n = \tau(f)$ and all $f \in \mathcal{F}_\tau$.
- $\mathcal{A}$ is a *subdirect product* of algebras $\mathcal{B}_i$ $(i \in I)$ if $\mathcal{A}$ is a subalgebra of $\prod_{i \in I} \mathcal{B}_i$ and for each $i \in I$ and each $b \in B_i$ there is an $a \in A$ such that the i-th coordinate of a is b (indicated by writing $a_i = b$).
- $\mathcal{A}$ is *subdirectly irreducible* if whenever $\mathcal{A}$ is isomorphic to a subdirect product of algebras $\mathcal{B}_i$ $(i \in I)$ then $\mathcal{A} \cong \mathcal{B}_i$ for some $i \in I$.[7]

The last two notions are used in the following important result.

Theorem 1.4.1 (*Birkhoff's Subdirect Product Theorem, 1944*) Every algebra is isomorphic to a subdirect product of subdirectly irreducible algebras. □

A *congruence* on an algebra $\mathcal{A}$ is an equivalence relation R on A that is compatible with the operations of $\mathcal{A}$ in the sense that

$$a_0 R b_0 \text{ and } \ldots \text{ and } a_{n-1} R b_{n-1} \text{ implies } f^{\mathcal{A}}(a_0, \ldots, a_{n-1}) R f^{\mathcal{A}}(b_0, \ldots, b_{n-1})$$

for all $a_i, b_i \in A$, $i < n = \tau(f)$ and all $f \in \mathcal{F}_\tau$. The set of all congruences on an algebra $\mathcal{A}$ is denoted by $Con(\mathcal{A})$; it is closed under arbitrary intersections and hence is a complete lattice[8]. Congruences provide a way of describing all homomorphic images of an algebra. Given a congruence R on $\mathcal{A}$, the *quotient algebra* $\mathcal{A}/R$ is defined on the set A/R of equivalence classes ("$[a]_R$" denotes the equivalence class of a) by

$$f^{\mathcal{A}/R}([a_0]_R, \ldots, [a_{n-1}]_R) = [f^{\mathcal{A}}(a_0, \ldots, a_{n-1})]_R$$

for all $a_0, \ldots, a_{n-1} \in A$, $i < n = \tau(f)$ and all $f \in \mathcal{F}_\tau$. The compatibility condition above guarantees that these operations are well-defined. The so-called *canonical*

[7]Note that if $I = \emptyset$ then $\prod_{i \in \emptyset} B_i$ is a one-element algebra, hence one-element algebras are not subdirectly irreducible.

[8]In fact it is an *algebraic lattice* since every congruence is the join of the compact (= finitely generated) congruences that it contains.

map $\pi : \mathcal{A} \to \mathcal{A}/R$ defined by $\pi(a) = [a]_R$ is a surjective homomorphism, so $\mathcal{A}/R$ is a homomorphic image of $\mathcal{A}$. Conversely, given any surjective homomorphism $h : \mathcal{A} \to \mathcal{B}$, the *kernel* of h, defined by

$$\ker(h) = \{(x, y) \in A^2 : h(x) = h(y)\},$$

is a congruence on $\mathcal{A}$, and $\mathcal{A}/\ker(h) \cong \mathcal{B}$. (This is usually called the *Homomorphism Theorem*, or sometimes the *First Isomorphism Theorem*).

Some algebras, particularly those we are interested in, such as Boolean algebras and relation algebras, are entirely defined by equations. We postpone a discussion of these till after we have dealt with equational logic in Sect. 1.6.

1.5 Categories

The language of categories is often used to present a topic at a high level of abstraction. A *category* $\mathbf{C}$ consists of a class[9] of *objects* $\mathsf{Obj}_\mathbf{C}$ and a class of *morphisms* $\mathsf{Mor}_\mathbf{C}$ that satisfy the following conditions.

- For each morphism f there is an object $\mathsf{source}\ f$ from which the morphism originates and an object $\mathsf{target}\ f$ where it ends. If $\mathsf{source}\ f = A$ and $\mathsf{target}\ f = B$ then we write $f : A \to B$.
- For all morphisms $f : A \to B$ and $g : B \to C$ there exists a *composition morphism*[10] $g \circ f : A \to C$, and for any $h : C \to D$ we have associativity $h \circ (g \circ f) = (h \circ g) \circ f$.
- For each object A there is an *identity morphism* $\mathsf{id}_A : A \to A$, and for any $f : A \to B$ and $g : B \to A$ we have $\mathsf{id}_A \circ g = g$ and $f \circ \mathsf{id}_A = f$.

The collection of all morphisms between objects $A, B \in \mathsf{Obj}_\mathbf{C}$ is denoted by $\mathsf{Mor}_\mathbf{C}[A, B]$. Some categories of interest are

- **Set** with sets as objects and functions as morphisms,
- **Rel** with sets as objects and relations as morphisms, and
- **Alg** with algebras as objects and homomorphisms as morphisms.

The language of category theory is well-suited to describing relationships between different areas of mathematics. *Functors* are structure-preserving maps between categories that make such relationships explicit and provide a reliable means for transferring results between different fields. More precisely, a *covariant functor* $\mathbf{F} : \mathbf{C} \to \mathbf{D}$ maps objects to objects and morphisms to morphisms (from $\mathbf{C}$ to $\mathbf{D}$) such that

- $\mathbf{F} f : \mathbf{F} A \to \mathbf{F} B$ for any morphism $f : A \to B$,
- $\mathbf{F}(g \circ f) = (\mathbf{F} g) \circ (\mathbf{F} f)$ for any morphisms $f : A \to B$, $g : B \to C$ and
- $\mathbf{F}(\mathsf{id}_A) = \mathsf{id}_{\mathbf{F} A}$ for any object A.

[9]The distinction between *set* and *class* is not relevant here, so readers unfamiliar with classes may just think of them as sets.

[10]Recall that we have used the small circle rather than the semicolon to indicate functional composition; we do the same for morphisms. It is useful to keep in mind that by convention this inverts the order: $g \circ f = f;g$.

For a *contravariant functor* the third condition is kept, but the first two conditions are changed to

- $\mathbf{F} f : \mathbf{F} B \to \mathbf{F} A$ for any morphism $f : A \to B$,
- $\mathbf{F}(g \circ f) = (\mathbf{F} f) \circ (\mathbf{F} g)$ for any morphisms $f : A \to B$, $g : B \to C$.

A functor $\mathbf{F} : \mathbf{C} \to \mathbf{D}$ is

- *full* if for all $A, B \in \mathsf{Obj}_\mathbf{C}$ and every $g : \mathbf{F} A \to \mathbf{F} B$ there exists $f : A \to B$ such that $\mathbf{F} f = g$, i.e. the restriction of $\mathbf{F}$ to $\mathsf{Mor}_\mathbf{C}[A, B]$ is *surjective* (onto $\mathsf{Mor}_\mathbf{D}[\mathbf{F} A, \mathbf{F} B]$).
- *faithful* if the restriction of $\mathbf{F}$ to $\mathsf{Mor}_\mathbf{C}[A, B]$ is *injective* (one-one) for all $A, B \in \mathsf{Obj}_\mathbf{C}$,
- *dense* if for every $D \in \mathsf{Obj}_\mathbf{D}$ there exists an object $C \in \mathsf{Obj}_\mathbf{C}$ such that $\mathbf{F} C = D$,
- an *equivalence* if $\mathbf{F}$ is covariant, full, faithful and dense,
- a *duality* if $\mathbf{F}$ is contravariant, full, faithful and dense.

Two categories $\mathbf{C}$ and $\mathbf{D}$ are *equivalent* if there exists an equivalence $\mathbf{F} : \mathbf{C} \to \mathbf{D}$, and they are *dual* if there exists a duality $\mathbf{F} : \mathbf{C} \to \mathbf{D}$.

1.6 Logics

There are many general perspectives on what constitutes a logic. Here we take the view that a logic consists of a collection of "similar" theories, defined by the following three items.

Syntax, specifying the *symbols* (variables, relation symbols, function symbols, connectives and/or quantifiers) of the theory, and how to combine them to obtain the properly formed expressions one wants to reason about. These expressions are traditionally called (*well-formed*) *formulae* (or *sentences*) of the logic.

Semantics, defined by a collection of models (i.e. mathematical structures, such as in Sect. 1.4) in which the symbols are interpreted, and a notion of truth of a sentence in a model. For a model M and a sentence φ, the expression 'φ is true in M' is abbreviated $M \models \varphi$.[11] A sentence φ is called a *logical consequence* of a set of sentences Σ, in symbols $\Sigma \models \varphi$ if in every model in which all sentences of Σ are true, φ is also true. Sentences that are logical consequences of the empty set (satisfied by all models under consideration) are called *valid*.

A proof system, which specifies an effective method by which a sentence φ can be "proved" from a set of sentences Σ, in symbols $\Sigma \vdash \varphi$. Sentences deducible from the empty set are called *theorems*.

For a given theory, the proof system is said to be *sound* with respect to the semantics if every proof produces only logical consequences, i.e.

$$\Sigma \vdash \varphi \quad \text{implies} \quad \Sigma \models \varphi$$

[11]$\models$ is called the *satisfaction symbol* and denotes semantical truth.

for all sentences φ and all sets of sentences Σ. The proof system is said to be *complete* if the converse holds, i.e. if every logical consequence has a proof within the logic.

A particular theory is normally identified with its set of logical truths, which coincides with its set of theorems if the proof system is sound and complete. The method for deriving proofs is usually based on

- *axioms:* a set of sentences that are considered theorems (by definition) and
- *proof rules:* a collection of rules specifying how a sentence of a given form can be deduced from finitely many other sentences.

In this case the set of all theorems is the smallest set that contains all the axioms and is closed under application of the proof rules. There are many different styles of proof systems such as natural deduction, Gentzen sequent calculus, tableaux method, resolution, connection method, Hilbert style, Fitch style, etc. Some of these systems are very close to the style of proof used in everyday mathematics, while others are more suitable for automated theorem proving. We will not consider any of these systems in general since, from a logical point of view, the exact nature of the proof system is not important, as long as it is sound and complete. (The situation is similar to a program written in different programming languages, some more suitable for humans, others for machines, but from the user's point of view the correctness of the program should be the primary concern.) Related theories of the same logic are obtained by modifying the semantics (restricting the interpretations) or the proof system (extending the set of axioms and/or proof rules). We now describe some particular logics.

Classical propositional logic

The most basic classical reasoning system restricts itself to a language $\mathcal{L}$ consisting of sentences φ built up from *propositions* $p_0, p_1, p_2, \ldots$ combined with a unary connective $\neg$ (negation) and a binary connective $\vee$ (logical or). The set of all *propositional sentences* is the smallest set $\mathcal{S}$ that contains all the propositions, as well as $\neg\varphi$ and $(\varphi \vee \psi)$ for all $\varphi, \psi \in \mathcal{S}$. The following abbreviations are used to aid readability:

$$\begin{aligned}
&(\varphi \wedge \psi) \triangleq \neg(\neg\varphi \vee \neg\psi),\\
&\varphi \to \psi \triangleq \neg\varphi \vee \psi,\\
&\varphi \leftrightarrow \psi \triangleq (\varphi \to \psi) \wedge (\psi \to \varphi).
\end{aligned}$$

To avoid excessively many parentheses, $\neg$ is given highest priority, followed by $\vee$, $\wedge$ with equal priority and finally $\to$ and $\leftrightarrow$ with lowest priority.

The semantics of a propositional theory is given by a collection of valuations, where a *valuation* is a function from $\{p_0, p_1, p_2, \ldots\}$ to $\{\mathsf{true}, \mathsf{false}\}$. For a sentence φ and a valuation v, the satisfaction relation $\mathsf{v} \models \varphi$ is defined inductively:

$$\begin{aligned}
&\mathsf{v} \models p_i \text{ if } \mathsf{v}(p_i) = \mathsf{true},\\
&\mathsf{v} \models (\varphi \vee \psi) \text{ if at least one of } \mathsf{v} \models \varphi \text{ and } \mathsf{v} \models \psi \text{ holds},\\
&\mathsf{v} \models \neg\varphi \text{ if } \mathsf{v} \not\models \varphi, \quad \text{(i.e. } \mathsf{v} \models \varphi \text{ does not hold).}
\end{aligned}$$

A set Σ of sentences *logically implies* a sentence φ, in symbols $\Sigma \models \varphi$, if every valuation v that satisfies all sentences in Σ also satisfies φ. A sentence φ is called a *logical truth* or *tautology* if $\emptyset \models \varphi$, i.e. if it is satisfied by all valuations. Two sentences φ, ψ are said to be *logically equivalent* ($\varphi \equiv \psi$) if $\varphi \leftrightarrow \psi$ is a logical truth.

There are many different proof systems for propositional logic, in many different styles. Almost any introductory textbook in logic will contain an axiomatisation of propositional logic with a proof of soundness and completeness. For any such system there is a natural Boolean algebra associated with the logic, obtained as follows. Define two formulae φ and ψ to be *provably equivalent*, written $\varphi \approx \psi$, iff $\varphi \leftrightarrow \psi$ is a theorem of the logic, i.e. $\vdash \varphi \leftrightarrow \psi$. This is an equivalence relation, and so we can form an equivalence class $[\varphi] = \{\theta : \varphi \approx \theta\}$ for every formula φ. Denote the set of all these equivalence classes by $\mathcal{L}/\approx$, then define over this *quotient set* operations arising from the logical connectives as follows:

$$\begin{aligned}
&[\varphi] \sqcap [\psi] \triangleq [\varphi \wedge \psi],\\
&[\varphi] \sqcup [\psi] \triangleq [\varphi \vee \psi],\\
&\overline{[\varphi]} \triangleq [\neg\varphi].
\end{aligned}$$

This method yields, for any logic, what is called its *Lindenbaum-Tarski algebra*. In the present case it is easy to check that the Lindenbaum-Tarski algebra of (classical) propositional logic is in fact a Boolean algebra. Since we have assumed the formalisation to be sound and complete, it could be proved further that the equivalence class which is the maximum element in $\mathcal{L}/\approx$ contains all and only the tautologies, and that two formulae φ and ψ are provably equivalent iff they are logically equivalent.

Classical first-order logic

In first-order logic we extend propositional logic by considering also the internal structure of propositions. For a fixed type τ and a set V of variables, an *atomic formula* is a string of symbols of the form $R(t_0, \ldots, t_{n-1})$ such that $R \in \mathcal{R}_\tau$ and $t_i \in T_\tau(V)$ $(i < n = \tau(R))$, where $\mathcal{R}_\tau$ and $T_\tau(V)$ are as in Sect. 1.4. The set Frm_τ of all *first-order formulae of type* τ is the smallest set that contains all atomic formulae as well as the formulae $\exists v\,(\varphi)$, $\neg\varphi$ and $(\varphi \vee \psi)$ for all $v \in V$ and $\varphi, \psi \in Frm_\tau$. The symbol $\exists$ is the *existential quantifier*. The *universally* quantified formula $\forall v\,(\varphi)$ is an abbreviation for $\neg\exists v\,(\neg\varphi)$, and the connectives $\wedge, \to, \leftrightarrow$ are defined as for propositional logic. $\exists v$ and $\forall v$ are considered unary connectives with higher priority than the binary connectives. Associated with each formula φ is a set $\mathsf{free}(\varphi)$ of *free variables*. For an atomic formula, $\mathsf{free}(\varphi)$ is the set of variables that occur in φ, and this is extended inductively by

$$\begin{aligned}
&\mathsf{free}(\exists v\,(\varphi)) = \mathsf{free}(\varphi) - \{v\},\\
&\mathsf{free}(\neg\varphi) = \mathsf{free}(\varphi),\\
&\mathsf{free}(\varphi \vee \psi) = \mathsf{free}(\varphi) \cup \mathsf{free}(\psi).
\end{aligned}$$

A *first-order sentence* is a formula φ that has no free variables ($\mathsf{free}(\varphi) = \emptyset$).

The semantics of a first-order theory of type τ is given by a collection of relational structures of type τ. Each relational structure $\mathcal{U}$ defines an *interpretation* $s \mapsto s^{\mathcal{U}}$ for the symbols $s \in \mathcal{F}_\tau \cup \mathcal{R}_\tau$. This map gives meaning to the function and relation symbols of the language. Observe that a valuation $\mathsf{v} : V \to U$ can be extended to $\bar{\mathsf{v}} : T_\tau(V) \to U$ by defining $\bar{\mathsf{v}}(f(t_0, \ldots, t_{n-1})) = f^{\mathcal{U}}(\bar{\mathsf{v}}(t_0), \ldots, \bar{\mathsf{v}}(t_{n-1}))$. We say that a formula φ is *true in* $\mathcal{U}$ *under the valuation* v, in symbols $\mathcal{U}, \mathsf{v} \models \varphi$ if one of the following conditions holds:

φ is $R(t_0, \ldots, t_{n-1})$ and $(\bar{\mathsf{v}}(t_0), \ldots, \bar{\mathsf{v}}(t_{n-1})) \in R^{\mathcal{U}}$,

φ is $\neg\psi$ and $\mathcal{U}, \mathsf{v} \not\models \psi$,

φ is $\psi \vee \psi'$ and at least one of $\mathcal{U}, \mathsf{v} \models \psi$ or $\mathcal{U}, \mathsf{v} \models \psi'$ holds,

φ is $\exists x\,(\psi)$ and $\mathcal{U}, \mathsf{v}' \models \psi$ for some $\mathsf{v}' : V \to U$ such that v and v' agree on $V - \{x\}$.

Finally, a formula φ is said to be *true in* $\mathcal{U}$ if $\mathcal{U}, \mathsf{v} \models \varphi$ for all valuations v. In this case $\mathcal{U}$ is also called a *model* of φ, in symbols $\mathcal{U} \models \varphi$.

As with syntax, the proof theory of first-order logic extends that of propositional logic, typically by giving some axioms and/or rules for dealing with the quantifiers. The algebraisation, however, of first-order logic is a considerably more complicated matter than for propositional logic. Relation algebras may be considered as an attempt to present in equational form first-order logic with no function symbols and only binary relation symbols. However, as mentioned in Chapt. 2, relation algebras only capture a certain fragment of this logic. Other attempts at presenting algebraic versions of first-order logic are the *cylindric algebras* of [Henkin, Monk+ 1971] and [Henkin, Monk+ 1985], and (earlier) the *polyadic algebras* of [Halmos 1962].

Equational logic

Equational logic can be considered as that fragment of first-order logic in which there is only one relational symbol, "$=$", to be interpreted as equality. It can also be added to a formalisation of first-order logic; the result is then *first-order logic with equality*. However, we consider equational logic separately because it is of interest in its own right: one of the central themes of algebraic logic is to reduce the reasoning of other proof systems to the elegant proof system of equational logic.

The syntax of an equational theory is based on an algebraic type τ and a set V of variables from which one obtains the set of terms $T_\tau(V)$ (Sect. 1.4). The collection of "formulae" is simply the set $T_\tau(V) \times T_\tau(V)$. A pair of terms (s, t) is called an *equation* and is written $s = t$.

The semantics of an equational theory of type τ is given by a collection of universal algebras of type τ. Each algebra $\mathcal{A}$ in this collection determines an interpretation which maps any $f \in \mathcal{F}_\tau$ to an operation $f^{\mathcal{A}}$. This interpretation can be extended inductively from $\mathcal{F}_\tau$ to all terms by considering a term $t \in T_\tau(V)$ as a template for composing the functions $f^{\mathcal{A}}$ that correspond to the function symbols f in t. The composite function defined by a term t is called the *induced term function* and is denoted by $t^{\mathcal{A}}$. The arity of this function is the number of distinct variables that appear in the term. An equation $s = t$ is satisfied in $\mathcal{A}$ if the induced term functions $s^{\mathcal{A}}$ and $t^{\mathcal{A}}$ are identical.

Proof systems for equational theories differ in their choice of axioms, but they all have the following in common:

- *reflexivity axiom:* $\vdash t = t$,
- *symmetry rule:* $s = t \vdash t = s$,
- *transitivity rule:* $r = s,\ s = t \vdash r = t$,
- *congruence rule:* $s_0 = t_0, \ldots, s_{n-1} = t_{n-1} \vdash f(s_0, \ldots, s_{n-1}) = f(t_0, \ldots, t_{n-1})$,
- *substitution rule:* $s = t \vdash s[v/r] = t[v/r]$ (where $s[v/r]$ is derived from s by substituting the variable v with the term r in all instances),

for all terms $r, s, s_0, \ldots, s_{n-1}, t, t_0, \ldots, t_{n-1}$, all $f \in \mathcal{F}_\tau$ and $n = \tau(f)$. The smallest equational theory (of type τ) is given by the set of reflexivity axioms since this set is closed under the above rules. The set of all equations is the largest equational theory, obtained by adding an axiom like $\vdash v = v'$ for distinct variables v, v'.

For a specific example, consider the equational theory of relation algebras, based on the type $\tau = \{(\sqcup, 2), (\overline{}, 1), (\mathbb{L}, 0), (;, 2), (\breve{\ }, 1), (\mathbb{I}, 0)\}$, and axiomatized by the equations (implicit) on Page 8. More generally, given a class $\mathcal{K}$ of algebras of type τ, we denote by $\mathbf{H}(\mathcal{K})$, $\mathbf{S}(\mathcal{K})$, $\mathbf{P}(\mathcal{K})$, $\mathbf{Ps}(\mathcal{K})$, $\mathbf{Si}(\mathcal{K})$ the class of all homomorphic images, all subalgebras, all products, all subdirect products and all subdirectly irreducibles of (members of) $\mathcal{K}$ respectively. For a set Σ of first-order formulae of type τ, let $\mathbf{Mod}(\Sigma) = \{\mathcal{M} : \mathcal{M} \models \Sigma\}$ be the class of all models of type τ that satisfy all formulae in Σ. If Σ contains only equations, $\mathbf{Mod}(\Sigma)$ is called an *equational class* or *variety*. The following is a fundamental result of universal algebra.

Theorem 1.6.1 (*Birkhoff's Preservation Theorem, 1935*) A class $\mathcal{K}$ of algebras of type τ is a variety if and only if $\mathcal{K}$ is closed under $\mathbf{H}$, $\mathbf{S}$ and $\mathbf{P}$ (i.e. $\mathbf{H}(\mathcal{K}) \subseteq \mathcal{K}$, $\mathbf{S}(\mathcal{K}) \subseteq \mathcal{K}$ and $\mathbf{P}(\mathcal{K}) \subseteq \mathcal{K}$).

Since the intersection of any collection of varieties is again a variety, every class of algebras $\mathcal{K}$ is contained in a smallest variety $\mathbf{V}(\mathcal{K})$, called the variety *generated* by $\mathcal{K}$. Based on Birkhoff's result, Tarski (1946) showed that $\mathbf{V}(\mathcal{K}) = \mathbf{HSP}(\mathcal{K})$.

Second-order logic

Many important concepts in computer science like well-foundedness cannot be formalized within the framework of first-order logic. Therefore in second-order logic the set of variables is extended by *relation variables*, which can be used when building terms, and quantification is allowed on these relation variables.

Validity of a formula φ in a relational structure $\mathcal{U}$ is defined relative to a valuation v of individual variables to elements of the universe as well as a valuation u of n-ary relation variables to n-ary relations on the universe. Similar to the first-order case we have

- $\mathcal{U}, \mathsf{u}, \mathsf{v} \models \exists R\,(\varphi)$ if $\mathcal{U}, \mathsf{u}', \mathsf{v} \models \varphi$ for some u' agreeing with u on all relation variables except R.

Again, a formula φ (possibly containing free individual or relation variables) is true in $\mathcal{U}$ if $\mathcal{U}, \mathsf{u}, \mathsf{v} \models \varphi$ for all valuations u and v. It is *valid* if it is true in every structure $\mathcal{U}$.

In contrast to first-order logic, second-order logic is incomplete: there is no proof system enumerating all valid second-order sentences. Therefore, attention is often restricted to certain fragments and/or certain classes of structures. For example *monadic second-order logic* allows only monadic (i.e. unary) relation variables (which are interpreted as subsets of a relational structure).

Modal logic

It is often helpful to formalize an area without the full generality of the quantifiers, but with more expressive power than plain propositional logic. In this case one can add further connectives to propositional logic and formalize their intended meanings. For example, when reasoning about a property Q of a computer program, statements like "after the next step Q" and "from now on Q" can be viewed as unary connectives applied to the proposition Q. Such unary connectives, often called *modalities*, are usually not truthfunctional: the truth value of a compound formula does not depend only on the truth values of its constituent formulae.

There are many modal logics; by way of example we consider a basic one called $\mathbf{K}$. This logic is built from propositions $\{p_0, p_1, \ldots\}$ and Boolean connectives $\neg$, $\vee$, just like ordinary propositional logic, but it also has an additional unary connective $\Diamond$, called *possibility*. For any propositional formula φ, the formula $\Diamond\varphi$ is read "possibly φ". A *necessity* operator $\Box$ can be defined as its dual: $\Box\varphi \triangleq \neg\Diamond\neg\varphi$.

The semantics of $\mathbf{K}$, as for other modal logics, is given in terms of a so-called *Kripke structure*: a relational structure $\mathcal{W} = (W, R)$ where W is a nonempty set of "possible worlds"[12] and R is a binary relation on W, usually called the *accessibility relation*. A valuation v is a function from $\{p_0, p_1, \ldots\}$ to $\mathcal{P}(W)$. A Kripke structure $\mathcal{W}$ together with a valuation is called a *Kripke model based on* $\mathcal{W}$. Satisfaction of a sentence φ in a Kripke model $\mathcal{M} = (W, R, \mathsf{v})$ at a world $w \in W$ is defined inductively:

$\mathcal{M}, w \models p_i$ if $w \in \mathsf{v}(p_i)$.

$\mathcal{M}, w \models (\varphi \vee \psi)$ if $\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$.

$\mathcal{M}, w \models \neg\varphi$ if $\mathcal{M}, w \not\models \varphi$.

$\mathcal{M}, w \models \Diamond\varphi$ if $\mathcal{M}, w' \models \varphi$ for some $w' \in W$ with wRw'.

A sentence φ is valid in a Kripke model $\mathcal{M}$, if $\mathcal{M}, w \models \varphi$ for all $w \in W$.

To get a complete formalisation of the modal logic $\mathbf{K}$, start with any complete formalisation of propositional logic, and add the following axiom and rules:

- *monotonicity*: $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$
- *necessitation*: $\varphi \vdash \Box\varphi$
- *modus ponens*: φ, $(\varphi \to \psi) \vdash \psi$ (if not already included).

[12]This traditional terminology is motivated by philosophy; in particular by the Leibnizian idea that "necessarily true" means "true in all possible worlds".

Many further modal logics can be obtained by adding further axioms to those of $\mathbf{K}$. For example, a well-known logic called $\mathbf{S4}$ is obtained by adding to $\mathbf{K}$ the axioms

$$\Box\varphi \to \varphi$$
$$\Box\varphi \to \Box\Box\varphi.$$

In order for this formalisation of $\mathbf{S4}$ to be complete we need to add to its Kripke semantics the stipulation that the accessibility relation should be reflexive and transitive – i.e. a quasi-order. This correspondence between logical conditions on the modal operators and first-order conditions on the accessibility relation is typical of modal logic. The same idea also applies to operators of higher arity than just unary, and to the simultaneous use of many different modal operators (multi-modal logic).

1.7 Conclusion

The calculus of relations is pervasive in mathematics and logic, in the same sense of pervasiveness as the calculus of sets. In this chapter we have highlighted some aspects of it; these serve as background to the remaining chapters, in which we address the role and applications of relational methods in computer science.

The basic connection is that we may think of a program as an input-output relation: from a given initial state, it terminates (if at all) in another state, where a "state" is thought of as a snapshot of the current values of all the program variables. If we consider programs to be nondeterministic, then it is natural to think of a program simply as a binary relation over the set of all possible states. The *relational model* of program semantics, then, in its simplest form, is a relational structure $(\mathcal{S}, \{R_i\}_{i\in I})$, where $\mathcal{S}$ is the "state space" (the set S of states, usually structured in some way, e.g. as an ordered set), and $\{R_i\}_{i\in I}$ is a suitably indexed set of binary relations over the state space, representing programs. This can be thought of as a Kripke structure: the "possible worlds" now being states, and each program acting as an "accessibility relation" in the sense that from a given initial state the program can access certain other (terminal) states. A logical characterisation of program semantics would then aim to find a "program logic", with a modality for each of the programs, which is complete with respect to this semantic structure. Alternatively, each of these binary relations over S can give rise to a unary operator over $\mathcal{P}(S)$ (Peirce product, for example), and since sets of states may be thought of extensionally as properties (or predicates) of states, this would lead to the realm of so-called "predicate transformer semantics". It is this close interaction between a relational structure, a logic, and an algebra, that allows us to make use of the calculus of relations in a variety of ways.