PL is an important logic with applications in software and hardware design and analysis, knowledge representation, combinatorial optimization, and complexity theory, to name a few. Although relatively simple, the Boolean structure that is central to PL is often a main source of complexity in applications of the algorithmic reasoning that is the focus of Part II. Exercise 8.1 explores this point in more depth.

Besides being an important logic in its own right, PL serves to introduce the main concepts that are important throughout the book, in particular syntax, semantics, and satisfiability and validity. Chapter 2 presents first-order logic by building on the concepts of this chapter.

## Bibliographic Remarks

For a complete and concise presentation of propositional logic, see Smullyan's text *First-Order Logic* [87]. The semantic argument method is similar to Smullyan's tableau method.

The DPLL algorithm is based on work by Davis and Putnam, presented in [26], and by Davis, Logemann, and Loveland, presented in [25].

## Exercises

**1.1 (PL validity & satisfiability).** For each of the following PL formulae, identify whether it is valid or not. If it is valid, prove it with a truth table or semantic argument; otherwise, identify a falsifying interpretation. Recall our conventions for operator precedence and associativity from Section 1.1.

(*a*) $P \wedge Q \rightarrow P \rightarrow Q$
(*b*) $(P \rightarrow Q) \vee P \wedge \neg Q$
(*c*) $(P \rightarrow Q \rightarrow R) \rightarrow P \rightarrow R$
(*d*) $(P \rightarrow Q \vee R) \rightarrow P \rightarrow R$
(*e*) $\neg(P \wedge Q) \rightarrow R \rightarrow \neg R \rightarrow Q$
(*f*) $P \wedge Q \vee \neg P \vee (\neg Q \rightarrow \neg P)$
(*g*) $(P \rightarrow Q \rightarrow R) \rightarrow \neg R \rightarrow \neg Q \rightarrow \neg P$
(*h*) $(\neg R \rightarrow \neg Q \rightarrow \neg P) \rightarrow P \rightarrow Q \rightarrow R$

**1.2 (Template equivalences).** Use the truth table or semantic argument method to prove the following template equivalences.

(*a*) $\top \Leftrightarrow \neg\bot$
(*b*) $\bot \Leftrightarrow \neg\top$
(*c*) $\neg\neg F \Leftrightarrow F$
(*d*) $F \wedge \top \Leftrightarrow F$
(*e*) $F \wedge \bot \Leftrightarrow \bot$
(*f*) $F \wedge F \Leftrightarrow F$

(*g*) $F \vee \top \;\Leftrightarrow\; \top$
(*h*) $F \vee \bot \;\Leftrightarrow\; F$
(*i*) $F \vee F \;\Leftrightarrow\; F$
(*j*) $F \rightarrow \top \;\Leftrightarrow\; \top$
(*k*) $F \rightarrow \bot \;\Leftrightarrow\; \neg F$
(*l*) $\top \rightarrow F \;\Leftrightarrow\; F$
(*m*) $\bot \rightarrow F \;\Leftrightarrow\; \top$
(*n*) $\top \;\leftrightarrow\; F \;\Leftrightarrow\; F$
(*o*) $\bot \;\leftrightarrow\; F \;\Leftrightarrow\; \neg F$
(*p*) $\neg(F_1 \wedge F_2) \;\Leftrightarrow\; \neg F_1 \vee \neg F_2$
(*q*) $\neg(F_1 \vee F_2) \;\Leftrightarrow\; \neg F_1 \wedge \neg F_2$
(*r*) $F_1 \rightarrow F_2 \;\Leftrightarrow\; \neg F_1 \vee F_2$
(*s*) $F_1 \rightarrow F_2 \;\Leftrightarrow\; \neg F_2 \rightarrow \neg F_1$
(*t*) $\neg(F_1 \rightarrow F_2) \;\Leftrightarrow\; F_1 \wedge \neg F_2$
(*u*) $(F_1 \vee F_2) \;\wedge\; F_3 \;\Leftrightarrow\; (F_1 \wedge F_3) \;\vee\; (F_2 \wedge F_3)$
(*v*) $(F_1 \wedge F_2) \;\vee\; F_3 \;\Leftrightarrow\; (F_1 \vee F_3) \;\wedge\; (F_2 \vee F_3)$
(*w*) $(F_1 \rightarrow F_3) \;\wedge\; (F_2 \rightarrow F_3) \;\Leftrightarrow\; F_1 \vee F_2 \;\rightarrow\; F_3$
(*x*) $(F_1 \rightarrow F_2) \;\wedge\; (F_1 \rightarrow F_3) \;\Leftrightarrow\; F_1 \;\rightarrow\; F_2 \wedge F_3$
(*y*) $F_1 \;\rightarrow\; F_2 \;\rightarrow\; F_3 \;\Leftrightarrow\; F_1 \wedge F_2 \;\rightarrow\; F_3$
(*z*) $(F_1 \leftrightarrow F_2) \;\wedge\; (F_2 \leftrightarrow F_3) \;\Rightarrow\; (F_1 \leftrightarrow F_3)$

**1.3 (Redundant logical connectives).** Given $\top$, $\wedge$, and $\neg$, prove that $\bot$, $\vee$, $\rightarrow$, and $\leftrightarrow$ are redundant logical connectives. That is, show that each of $\bot$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, and $F_1 \leftrightarrow F_2$ is equivalent to a formula that uses only $F_1$, $F_2$, $\top$, $\vee$, and $\neg$.

**1.4 (The nand connective).** Let the logical connective $\overline{\wedge}$ (pronounced "nand") be defined according to the following truth table:

| $F_1$ | $F_2$ | $F_1 \overline{\wedge} F_2$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Show that all standard logical connectives can be defined in terms of $\overline{\wedge}$.

**1.5 (Normal forms).** Convert the following PL formulae to NNF, DNF, and CNF via the transformations of Section 1.6.

(*a*) $\neg(P \;\rightarrow\; Q)$
(*b*) $\neg(\neg(P \wedge Q) \;\rightarrow\; \neg R)$
(*c*) $(Q \wedge R \;\rightarrow\; (P \vee \neg Q)) \;\wedge\; (P \vee R)$
(*d*) $\neg(Q \rightarrow R) \;\wedge\; P \;\wedge\; (Q \;\vee\; \neg(P \wedge R))$

**1.6 (Graph coloring).** A solution to a **graph coloring** problem is an assignment of colors to vertices such that no two adjacent vertices have the same color. Formally, a finite graph $G = \langle V, E\rangle$ consists of vertices $V = \{v_1, \ldots, v_n\}$ and edges $E = \{\langle v_{i_1}, w_{i_1}\rangle, \ldots, \langle v_{i_k}, w_{i_k}\rangle\}$. The finite set of colors is given by $C = \{c_1, \ldots, c_m\}$. A problem instance is given by a graph and a set of colors: the problem is to assign each vertex $v \in V$ a $\mathsf{color}(v) \in C$ such that for every edge $\langle v, w\rangle \in E$, $\mathsf{color}(v) \neq \mathsf{color}(w)$. Clearly, not all instances have solutions.

Show how to encode an instance of a graph coloring problem into a PL formula $F$. $F$ should be satisfiable iff a graph coloring exists.

($a$) Describe a set of constraints in PL asserting that every vertex is colored. Since the sets of vertices, edges, and colors are all finite, use notation such as "$\mathsf{color}(v) = c$" to indicate that vertex $v$ has color $c$. Realize that such an assertion is encodeable as a single propositional variable $P_v^c$.
($b$) Describe a set of constraints in PL asserting that every vertex has at most one color.
($c$) Describe a set of constraints in PL asserting that no two connected vertices have the same color.
($d$) Identify a significant optimization in this encoding. *Hint*: Can any constraints be dropped? Why?
($e$) If the constraints are not already in CNF, specify them in CNF now. For $N$ vertices, $K$ edges, and $M$ colors, how many variables does the optimized encoding require? How many clauses?

**1.7 (CNF).** Example 1.25 constructs a CNF formula that is equisatisfiable to a given small formula in DNF.

($a$) If distribution of disjunction over conjunction (described in Section 1.6) were used, how many clauses would the resulting formula have?
($b$) Consider the formulae

$$F_n : \bigvee_{i=1}^{n} (Q_i \wedge R_i)$$

for positive integers $n$. As a function of $n$, how many clauses are in
($i$) the formula $F'$ constructed based on distribution of disjunction over conjunction?
($ii$) the formula

$$F' : \mathsf{Rep}(F_n) \wedge \bigwedge_{G \in S_{F_n}} \mathsf{En}(G) \ ?$$

($iii$) For which $n$ is the distribution approach better?

**1.8 (DPLL).** Describe the execution of DPLL on the following formulae.

($a$) $(P \vee \neg Q \vee \neg R) \wedge (Q \vee \neg P \vee R) \wedge (R \vee \neg Q)$
($b$) $(P \vee Q \vee R) \wedge (\neg P \vee \neg Q \vee \neg R) \wedge (\neg P \vee Q \vee R) \wedge (\neg Q \vee R) \wedge (Q \vee \neg R)$