Bair & Katoen:
Principles of Model Checking.
MIT Press 2008
Pages 909 – 920

Appendix A

Appendix: Preliminaries

A.1 Frequently Used Symbols and Notations

The symbol ■ only serves readability. It indicates the end of a definition, of a remark, or of other numbered text fragments. The abbreviation "iff" stands for "if and only if". We also often use logical symbols, such as $\wedge$ for "and", $\vee$ for "or" and the following quantifiers:

$\exists$	"there exists ..."
$\forall$	"for all"
$\overset{\infty}{\exists}$	"there exist infinitely many"
$\overset{\infty}{\forall}$	"for almost all, i.e., for all except for finitely many".

The Greek Alphabet At various places, Latin and Greek letters serve as symbols for certain mathematical objects. Although not all Greek letters will be used in this monograph, Figure A.1 shows the complete Greek alphabet. The symbols Γ, Δ, Θ, Λ, Ξ, Σ, Υ, Φ, Ψ, and Ω are capital letters. All other symbols in Figure A.1 are lowercase letters.

Natural and Real Numbers The symbol $\mathbb{N}$ denotes the set $\{0, 1, 2, \ldots\}$ of natural numbers. The set of real numbers is denoted $\mathbb{R}$. Subsets of $\mathbb{N}$ and $\mathbb{R}$ are often denoted by subscripts. For instance, $\mathbb{R}_{\geqslant 0}$ denotes the set of non-negative reals, while $\mathbb{R}_{>5}$ denotes the interval $]5, \infty[$.

α	alpha	ι	iota	ρ, ϱ	rho
β	beta	κ	kappa	σ, ς, Σ	sigma
γ, Γ	gamma	λ, Λ	lambda	τ	tau
δ, Δ	delta	μ	mu	υ, Υ	upsilon
ϵ, ε	epsilon	ν	nu	ϕ, φ, Φ	phi
ζ	zeta	ξ, Ξ	xi	χ	chi
η	eta	o	omicron	ψ, Ψ	psi
θ, ϑ, Θ	theta	π, ϖ, Π	pi	ω, Ω	omega

Figure A.1: The Greek alphabet.

Asymptotic operators $\mathcal{O}$, Ω, and Θ To specify the asymptotic growth of (cost) functions, the Landau symbols $\mathcal{O}$, Ω, and Θ are used (see, e.g., [100]):

$$\begin{aligned} &\mathcal{O} : \text{asymptotic upper bound}\\ &\Omega : \text{asymptotic lower bound}\\ &\Theta : \text{asymptotic upper and lower bound}\end{aligned}$$

The precise meaning is as follows. If $f : \mathbb{N} \to \mathbb{N}$ is a function, then $\mathcal{O}(f)$ denotes the set of all functions $g : \mathbb{N} \to \mathbb{N}$ such that there exists a constant $C > 0$ and natural number N with $g(n) \leqslant C \cdot f(n)$ for all $n \in \mathbb{N}$ where $n \geqslant N$. The function class $\Omega(f)$ consists of all functions $g : \mathbb{N} \to \mathbb{N}$ such that there exists a constant $C > 0$ and a natural number N with $g(n) \geqslant C \cdot f(n)$ for all $n \geqslant N$. The class $\Theta(f)$ is given by $\Theta(f) = \mathcal{O}(f) \cap \Omega(f)$. It is common to use the equality symbol instead of the element symbol "$\in$" and to write, e.g., $g = \mathcal{O}(f)$ or $g(n) = \mathcal{O}(f(n))$ rather than $g \in \mathcal{O}(f)$. Thus, "equations" with the asymptotic operators have to be read from the left to the right.

We write $\mathcal{O}(\text{poly}(n))$ to denote the class of all functions $g : \mathbb{N} \to \mathbb{N}$ that are polynomially bounded, i.e., $g(n) = \mathcal{O}(n^k)$ for some natural number k. Similarly, $\mathcal{O}(\exp(n))$ denotes the class of all functions $g : \mathbb{N} \to \mathbb{N}$ that are exponentially bounded, i.e., $g(n) = \mathcal{O}(a^{n^k})$ where $a > 1$ is a real constant and k a natural number.

Notation for Sets Let X be a set. The symbol 2^X stands for the *powerset* of X, i.e., the set consisting of all subsets of X. We write $|X|$ to denote the cardinality of X, i.e., the number of elements of X. (If X is infinite, then $|X| = \omega$.) Besides the standard symbols $\cup$ for union, $\cap$ for intersection, and $\setminus$ for "set-minus", i.e., $X \setminus Y = \{x \in X \mid x \notin Y\}$, the symbol $\uplus$ will be used which denotes *disjoint union*. Formally, for sets X, Y, $X \uplus Y$ is defined by $\{(x, 1) \mid x \in X\} \cup \{(y, 2) \mid y \in Y\}$. Additionally, we mention special cases for union and intersection. For $x \in X$, let Y_x be subset of a set Y. Then,

$$\bigcup_{x \in \varnothing} Y_x \; = \; \varnothing, \qquad \bigcap_{x \in \varnothing} Y_x \; = \; Y.$$

Thus, $\bigcap_{x \in \varnothing} \ldots$ depends on the chosen universal set Y.

Relations A relation denotes any set of tuples of fixed length. More precisely, let $X_1, \ldots, X_k$ be sets where $k \in \mathbb{N}$ with $k \geqslant 1$. Subsets of the Cartesian products $X_1 \times \ldots \times X_k$ are also called *relations* or *predicates*. The number k denotes the *arity*. If $X_1 = X_2 = \ldots = X_k = X$, then the subsets of

$$X^k = X_1 \times \ldots \times X_k$$

are called k-ary relations on X. Very often we have to deal with binary (2-ary) relations $\mathcal{R}$ over a set X. For these relations the infix notation $x\mathcal{R}y$ is often used instead of $(x, y) \in \mathcal{R}$. If X is a set and $\mathcal{R}$ a binary relation on X, then $\mathcal{R}$ is called

- *transitive* if for all x, y, $z \in X$, $x\mathcal{R}y$, and $y\mathcal{R}z$ implies $x\mathcal{R}z$;
- *reflexive* if for all $x \in X$, $x\mathcal{R}x$;
- *symmetric* if for all x, $y \in X$, $x\mathcal{R}y$ implies $y\mathcal{R}x$;
- *antisymmetric* if for all x, $y \in X$, $x\mathcal{R}y$, and $y\mathcal{R}x$ implies $x = y$.

Equivalences An equivalence relation (or briefly equivalence) on X is a transitive, reflexive, and symmetric binary relation on X. Symmetric symbols (like $\sim$, $\equiv$, or similar symbols) are often used to denote an equivalence relation. If $x \in X$, then $[x]_{\mathcal{R}} = \{y \in X \mid x\mathcal{R}y\}$ is called the *equivalence class* of x with respect to $\mathcal{R}$. If $\mathcal{R}$ follows from the context, $[x]$ is often written for short instead of $[x]_{\mathcal{R}}$. The *quotient space* of X with respect to $\mathcal{R}$ is the set of all equivalence classes with respect to $\mathcal{R}$ and is denoted $X/\mathcal{R}$. Then, $X/\mathcal{R} = \{[x]_{\mathcal{R}} \mid x \in X\}$. If $\mathcal{R}$ is an equivalence relation on X, then for all x, $y \in X$:

$$x\mathcal{R}y \text{ iff } [x]_{\mathcal{R}} = [y]_{\mathcal{R}} \text{ iff } [x]_{\mathcal{R}} \cap [y]_{\mathcal{R}} \neq \varnothing.$$

So the quotient space consists of pairwise disjoint nonempty subsets of X whose union just results in X. For each element $x \in X$, there is always exactly one element A of the quotient space with $x \in A$ (i.e. $A = [x]_{\mathcal{R}}$). The *index* of an equivalence relation $\mathcal{R}$ denotes the number of equivalence classes, i.e., the cardinality of $X/\mathcal{R}$. It is usual to use spellings like "$\mathcal{R}$ *is of finite index*" if the index of $\mathcal{R}$ is finite (a natural number).

Let $\mathcal{R}$ and $\mathcal{R}'$ be two equivalence relations on X. Relation $\mathcal{R}$ is a *refinement* of $\mathcal{R}'$ if $\mathcal{R} \subseteq \mathcal{R}'$, i.e., $\mathcal{R}$ "distinguishes" more elements than $\mathcal{R}'$. In this case, it is also said that $\mathcal{R}$ is *finer* than $\mathcal{R}'$ and that $\mathcal{R}'$ is *coarser* than $\mathcal{R}$. It is referred to as a *proper refinement* if $\mathcal{R}$ is a refinement of $\mathcal{R}'$ with $\mathcal{R} \neq \mathcal{R}'$. If $\mathcal{R}$ is a refinement of $\mathcal{R}'$, then every equivalence class with respect to $\mathcal{R}$ is contained in *exactly* one equivalence class with respect to $\mathcal{R}'$, because we have $[x]_{\mathcal{R}} \subseteq [x]_{\mathcal{R}'}$. This observation can be enforced as follows. Each equivalence class with respect to $\mathcal{R}'$ can be written as a disjoint union of equivalence classes with respect to $\mathcal{R}$. Thus, if $\mathcal{R}$ is a refinement of $\mathcal{R}'$, then $|X/\mathcal{R}'| \leqslant |X/\mathcal{R}|$. Therefore, the index of $\mathcal{R}'$ is at most the index of $\mathcal{R}$.

Transitive and Reflexive Closure Let $\mathcal{R}$ be a binary relation over X. The *transitive and reflexive closure* of $\mathcal{R}$ is the smallest transitive, reflexive binary relation on X containing $\mathcal{R}$. Usually, this is denoted by $\mathcal{R}^*$. Thus,

$$\mathcal{R}^* = \bigcup_{n \geqslant 0} \mathcal{R}^n$$

where $\mathcal{R}^0 = \{(x,x) \mid x \in X\}$ and $\mathcal{R}^{n+1} = \{(x,y) \in X \times X \mid \exists z \in X.(x,z) \in \mathcal{R} \wedge (z,y) \in \mathcal{R}^n\}$. If $\mathcal{R}$ is symmetric, then $\mathcal{R}^*$ is an equivalence relation.

Partitions A *partition* (or partitioning) of a set X is a set $\mathcal{B} \subseteq 2^X$ consisting of pairwise disjoint, nonempty subsets of X such that $\bigcup_{B \in \mathcal{B}} B = X$. Elements of $\mathcal{B}$ are called *blocks*, as well. In particular, each element $x \in X$ is included in exactly one block $B \in \mathcal{B}$. There is a close connection between equivalence relations on X and partitions of X. If $\mathcal{R}$ is an equivalence relation on X, then the quotient space $X/\mathcal{R}$ is a partition of X. Vice versa, if $\mathcal{B}$ is a partition of X, then

$$\{(x,y) \mid x \text{ and } y \text{ are in the same block } B \in \mathcal{B}\}$$

is an equivalence relation with quotient space $\mathcal{B}$.

Preorder A preorder $\mathcal{R}$ on X denotes a reflexive, transitive relation on X. Any preorder induces an equivalence, the so-called *kernel* of $\mathcal{R}$, which is given by $\mathcal{R} \cap \mathcal{R}^{-1}$, i.e., the relation $\{(x,y) \mid x\mathcal{R}y \text{ and } y\mathcal{R}x\}$.

A.2 Formal Languages

This section summarizes the main concepts of regular languages. Further details and the proofs of the results mentioned here can be found in any text book on formal language theory, see e.g., [214, 272, 363, 383].

Words over an Alphabet An alphabet is an arbitrary nonempty and finite set Σ. The elements of Σ are typically called symbols or letters. A word over Σ denotes a finite or infinite sequence of symbols in Σ, i.e., words have the form $w = A_1 A_2 \ldots A_n$ where $n \in \mathbb{N}$ or $\sigma = A_1 A_2 A_3 \ldots$ and where the A_i's are symbols in Σ. [1] The special case $n = 0$ is

[1] Formally, infinite words can be defined as functions $\sigma : \mathbb{N} \to \Sigma$ and the notation $\sigma = A_1 A_2 A_3 \ldots$ means that $\sigma(i) = A_i$ for all $i \in \mathbb{N}$. Similarly, finite words are obtained by functions $w : \{1, \ldots, n\} \to \Sigma$.

allowed, in which case the so-called empty word, denoted ε, is obtained. The *length* of a word is the number of symbols in the given word. Thus, for $w = A_1 A_2 \ldots A_n$, the length is n, while each infinite word has the length ω. (The Greek letter ω (omega) is typically used to denote "infinity".) Σ^* denotes the set consisting of all finite words over Σ, while Σ^ω denotes the set of all infinite words over Σ. Note that $\varepsilon \in \Sigma^*$. Thus, the set of all nonempty finite words, denoted Σ^+, is $\Sigma^* \setminus \{\varepsilon\}$. A set of finite words over the alphabet Σ is called a *language*, and is ranged over by $\mathcal{L}$ (and primed and subscripted versions thereof).

A *prefix* of a finite word $w = A_1 A_2 \ldots A_n$, is a word v of the form $A_1 A_2 \ldots A_i$ for some i where $0 \leqslant i \leqslant n$. A *suffix* of w is a word v of the form $A_i A_{i+1} \ldots A_n$ where $1 \leqslant i \leqslant n{+}1$. (In particular, ε is a prefix and suffix of any finite word.) The words of the form $A_i A_{i+1} \ldots A_j$ with $1 \leqslant i \leqslant j \leqslant n$ are called subwords of w. The definition of prefixes, suffixes, and subwords of infinite words are similar. For infinite word $\sigma = A_0 A_1 A_2 \ldots$, the suffix $A_j A_{j+1} A_{j+2} \ldots$ is denoted $\sigma[j..]$. For technical reasons, we start with the index 0. Thus, $\sigma = \sigma[0..]$.

Operations on Words Important operations on words are concatenation and finite repetition. *Concatenation* takes two words and "glues them together" to construct a new word. It is denoted by juxtaposition. For instance, the concatenation of the words BA and AAB yields the word $BA.AAB = BAAAB$. The concatenation of a word with itself is denoted by squaring, e.g., $(AB)^2$ equals $ABAB$; this is generalized in a straightforward manner for arbitrary n. In the special cases $n = 0$ and $n = 1$, we have $w^0 = \varepsilon$ (the empty word) and $w^1 = w$.

Finite repetition, also called *Kleene star* and denoted by *, of a finite word w yields the language consisting of all finite words that arise by zero or more (but finitely many) repetitions of w. Formally, for $w \in \Sigma^*$ we have $w^* = \{w^i \mid i \in \mathbb{N}\}$. For instance, $(AB)^* = \{\varepsilon, AB, ABAB, ABABAB, \ldots\}$. Note that the empty word ε is included in w^* for each finite word w. This is precisely the difference with the slight variant of the Kleene star, denoted $^+$, defined by $w^+ = \{w^i \mid i \in \mathbb{N}, i \geqslant 1\}$, or, equivalently, $w^+ = w^* \setminus \{\varepsilon\}$. For instance, $(AB)^+$ denotes the set $\{AB, ABAB, ABABAB, \ldots\}$.

Operations on Languages Concatenation is lifted to languages in a natural way as a pointwise extension of concatenation on words. The same applies to repetition. For languages $\mathcal{L}$, $\mathcal{L}_1, \mathcal{L}_2 \subseteq \Sigma^*$ we have

$$\mathcal{L}_1.\mathcal{L}_2 = \{\, w_1.w_2 \mid w_1 \in \mathcal{L}_1, w_2 \in \mathcal{L}_2 \,\}$$

and

$$\mathcal{L}^* = \bigcup_{i=0}^{\infty} \mathcal{L}^i, \quad \text{and} \quad \mathcal{L}^+ = \bigcup_{i=1}^{\infty} \mathcal{L}^i,$$

where $\mathcal{L}^i$ denotes the concatenation of i times $\mathcal{L}$. Note that the star and plus notation for languages over finite words is consistent with the standard notations Σ^* and Σ^+ for the set of all finite words over Σ with or without the empty word, respectively. For instance, for $\mathcal{L}_1 = \{ A, AB \}$ and $\mathcal{L}_2 = \{ \varepsilon, BBB \}$ we have that

$$\begin{aligned} \mathcal{L}_1.\mathcal{L}_2 &= \{ A, AB, ABBB, ABBBB \}, \text{and} \\ \mathcal{L}_1^2 &= \{ AA, AAB, ABAB, ABA \}. \end{aligned}$$

There are several equivalent formalisms to describe regular languages. In this monograph we only use regular expressions and finite automata. We start with the former and introduce the automata approach later.

Regular Expressions Regular expressions (denoted E or F) are built from the symbols $\underline{\varnothing}$ (to denote the empty language), $\underline{\varepsilon}$ (to denote the language $\{ \varepsilon \}$ consisting of the empty word), the symbols $\underline{A}$ for $A \in \Sigma$ (for the singleton sets $\{ A \}$), and the language operators "+" (union), "*" (Kleene star, finite repetition), and "." (concatenation). Formally, regular expressions can be defined inductively:

1. $\underline{\varnothing}$ and $\underline{\varepsilon}$ are regular expressions over Σ.
2. If $A \in \Sigma$, then $\underline{A}$ is a regular expression over Σ.
3. If E, E_1 and E_2 are regular expressions over Σ, then so are $\mathsf{E}_1 + \mathsf{E}_2$, $\mathsf{E}_1.\mathsf{E}_2$, and E^*.
4. Nothing else is a regular expression over Σ.

E^+ is an abbreviation for the regular expression $\mathsf{E}.\mathsf{E}^*$. The semantics of a regular expression E is a language $\mathcal{L}(\mathsf{E}) \subseteq \Sigma^*$ that is defined as follows:

$$\mathcal{L}(\underline{\varnothing}) = \varnothing, \quad \mathcal{L}(\underline{\varepsilon}) = \{ \varepsilon \}, \quad \mathcal{L}(\underline{A}) = \{ A \}$$

and

$$\mathcal{L}(\mathsf{E}_1 + \mathsf{E}_2) = \mathcal{L}(\mathsf{E}_1) \cup \mathcal{L}(\mathsf{E}_2), \quad \mathcal{L}(\mathsf{E}_1.\mathsf{E}_2) = \mathcal{L}(\mathsf{E}_1).\mathcal{L}(\mathsf{E}_2), \quad \mathcal{L}(\mathsf{E}^*) = \mathcal{L}(\mathsf{E})^*.$$

From this definition, we derive $\mathcal{L}(\mathsf{E}^+) = \mathcal{L}(\mathsf{E})^+$.

A language $\mathcal{L} \subseteq \Sigma^*$ is called *regular* if there is some regular expression E over Σ such that $\mathcal{L}(\mathsf{E}) = \mathcal{L}$. For instance, $\mathsf{E} = (\underline{A}+\underline{B})^*.\underline{B}.\underline{B}.(\underline{A}+\underline{B})$ is a regular expression over $\Sigma = \{ A, B \}$ representing the language

$$\mathcal{L}(\mathsf{E}) = \{ wBBA \mid w \in \Sigma^* \} \cup \{ wB^3 \mid w \in \Sigma^* \},$$

consisting of all finite words that end with three B's or with the word BBA. The regular expression $\mathsf{E}' = (\underline{A} + \underline{B})^*.\underline{B}.\underline{B}.(\underline{A} + \underline{B})^*$ represents the regular language consisting of all finite words over $\Sigma = \{ A, B \}$ that contain the subword BB.

It is standard to use a simplified syntax for regular expressions that does not distinguish between the atomic expression $\underline{x}$ and the symbol x for $x \in \{\varnothing, \varepsilon\} \cup \Sigma$ and skips the operator symbol ". " for concatenation. For example,

$$(A + B)^* BB(A + B)$$

stands for the regular expression $(\underline{A} + \underline{B})^*.\underline{B}.\underline{B}.(\underline{A} + \underline{B})$.

A.3 Propositional Logic

This section summarizes the basic principles of propositional logic. For more elaborate treatments we refer to the textbook [342].

Given is a finite set AP of *atomic propositions*, sometimes also called *propositional symbols.* In the following, Latin letters like a, b, and c (with or without subscripts) are used to denote elements of AP. The set of *propositional logic formulae* over AP, formulae for short, is inductively defined by the following four rules:

1. true is a formula.
2. Any atomic proposition $a \in AP$ is a formula.
3. If Φ_1, Φ_2 and Φ are formulae, then so are $(\neg\Phi)$ and $(\Phi_1 \wedge \Phi_2)$.
4. Nothing else is a formula.

Any formula stands for a proposition that might hold or not, depending on which of the atomic propositions are assumed to hold. Intuitively, the formula a stands for the propositions stating that a holds. The intuitive meaning of the symbol $\wedge$ is conjunction ("and"), i.e., $\Phi_1 \wedge \Phi_2$ holds if and only if both propositions Φ_1 and Φ_2 hold. The symbol $\neg$ denotes negation, i.e., $\neg\Phi$ holds if and only if Φ does not hold. Thus, e.g., $a \wedge \neg b$ holds if and only if a holds and b does not hold. The constant true stands for a proposition which holds in any context, independent of the interpretation of the atomic propositions a.

It is standard to use simplified notations for formulae with the help of derived operators and by declaring a precedence order on the basic and derived operators, which often allows skipping brackets. The standard precedence order assigns a higher priority to the unary negation operator $\neg$ than the binary conjunction operator $\wedge$. Thus, $\neg a \wedge b$ is short for

$((\neg a) \wedge b)$. Moreover, conjunction $\wedge$ binds stronger than the derived binary operators, such as

$\Phi_1 \vee \Phi_2$	$\overset{\text{def}}{=}$	$\neg(\neg\Phi_1 \wedge \neg\Phi_2)$	disjunction ("or")
$\Phi_1 \rightarrow \Phi_2$	$\overset{\text{def}}{=}$	$\neg\Phi_1 \vee \Phi_2$	implication
$\Phi_1 \leftrightarrow \Phi_2$	$\overset{\text{def}}{=}$	$(\neg\Phi_1 \wedge \neg\Phi_2) \vee (\Phi_1 \wedge \Phi_2)$	equivalence
$\Phi_1 \oplus \Phi_2$	$\overset{\text{def}}{=}$	$(\neg\Phi_1 \wedge \Phi_2) \vee (\Phi_1 \wedge \neg\Phi_2)$	parity (xor)

For example, $\neg a \vee \neg b \wedge c$ is a short-form notation for $(\neg a) \vee ((\neg b) \wedge c)$ which – by the definition of $\vee$ – stands for the formula $\Phi = (\neg(\neg a) \wedge \neg((\neg b)) \wedge c))$.

Abstract Syntax Throughout this monograph, we provide the definition of the syntax of logics in a more relaxed way. Skipping the syntactic rules for brackets (which can be derived from the precedence order of the operators that will be declared in words), the above inductive definition of propositional formulae over AP can be rewritten as

$$\Phi \ ::= \ \text{true} \ \big| \ a \ \big| \ \Phi_1 \wedge \Phi_2 \ \big| \ \neg\Phi$$

where $a \in AP$. The above can be understood as a casual notation for the Backus-Naur form of a context-free grammar over the alphabet $\Sigma = \{\text{true}\} \cup AP \cup \{\neg, \wedge\}$. In this short-form notation, the symbol Φ serves simultaneously for (1) a nonterminal symbol (variable) of the grammar and (2) its derived words over Σ^* (i.e., propositional formulae). The latter explains the indices in the term $\Phi_1 \wedge \Phi_2$, which is correct on the formula level, although the correct notation would be $\Phi \wedge \Phi$ (without indices) in the grammar.

Length of a Formula The length of a formula Φ is defined by the number of operators in Φ and is denoted by $|\Phi|$. For instance, the formula $\Phi = (\neg b) \wedge c$ has the length 2. Since in most cases we are only interested in the asymptotic length of formulae in formula sequences (Φ_n), we may also assign one cost unit to the derived operators $\vee$ and $\rightarrow$. In fact, the asymptotic formula length does not depend on whether $\vee$ and $\rightarrow$ are treated as a basic operator (with one cost unit per occurrence in a formula) or a derived one (using conjunction and negation).

Semantics of Propositional Logic To formalize the intuitive meaning of propositional formulae, we first need a precise definition of the "context" that declares which atomic propositions hold and which do not hold. This is done by means of an *evaluation* which assigns a truth value 0 ("false") or 1 ("true") to each atomic proposition. Formally, an *evaluation* for AP is a function $\mu : AP \rightarrow \{0, 1\}$. $\mathit{Eval}(AP)$ denotes the set of all evaluations for AP. The semantics of propositional logic is specified by a *satisfaction relation* $\models$ indicating the evaluations μ for which a formula Φ is true. Formally, $\models$ is a set of pairs (μ, Φ) where μ is an evaluation and Φ is a formula. It is written

$$\begin{array}{lll} \mu \models \text{true} & & \\ \mu \models a & \text{iff} & \mu(a) = 1 \\ \mu \models \neg\Phi & \text{iff} & \mu \not\models \Phi \\ \mu \models \Phi \wedge \Psi & \text{iff} & \mu \models \Phi \text{ and } \mu \models \Psi. \end{array}$$

Figure A.2: The satisfaction relation $\models$ of propositional logic.

$$\mu \models \Phi \ \text{ instead of } \ (\mu, \Phi) \in \models.$$

Accordingly, $\mu \not\models \Phi$ stands for $(\mu, \Phi) \notin \models$. Intuitively, $\mu \models \Phi$ stands for the fact that Φ is true under evaluation μ. The satisfaction relation $\models$ is inductively defined by the conditions indicated in Figure A.2. If $\mu \models \Phi$, then μ is called a *satisfied condition* for Φ. In literature, the notation $\mu(\Phi) = 1$ if $\mu \models \Phi$, and $\mu(\Phi) = 0$, if $\mu \not\models \Phi$, is used, too. The value $\mu(\Phi) \in \{0, 1\}$ is called the *truth-value* of Φ under μ.

Formulae with derived operators like disjunction $\vee$ or implication $\rightarrow$ have the expected semantics. Thus,

$$\begin{array}{lll} \mu \models \Phi \vee \Psi & \text{iff} & \mu \models \Phi \text{ or } \mu \models \Psi \\ \mu \models \Phi \rightarrow \Psi & \text{iff} & \mu \not\models \Phi \text{ or } \mu \models \Psi \\ & \text{iff} & \mu \models \Phi \text{ implies } \mu \models \Psi. \end{array}$$

Set Notation for Evaluations An alternative representation of evaluations for AP is based upon representation of sets. Each evaluation μ can be represented by the set $A_\mu = \{a \in AP \mid \mu(a) = 1\}$. And conversely, an evaluation $\mu = \mu_A$ with $A = A_\mu$ can be assigned to each subset A of AP. Evaluation μ_A is the characteristic function of A_μ, that is, $\mu_A(a) = 1$ if $a \in A$ and $\mu_A(a) = 0$ if $a \notin A$. This observation suggests extending the satisfaction relation $\models$ to subsets of AP by

$$A \models \Phi \quad \text{iff} \quad \mu_A \models \Phi.$$

As an example, we look at $\Phi = (a \wedge \neg b) \vee c$. Given an evaluation μ with $\mu(a) = 0$ and $\mu(b) = \mu(c) = 1$, then $\mu \not\models a \wedge \neg b$ and $\mu \models c$, and thus, $\mu \models \Phi$. The accompanying set A_μ is $A_\mu = \{b, c\}$. Hence, $\{b, c\} \models \Phi$. The empty set induces an evaluation $\mu_\varnothing$ with $\mu_\varnothing(a) = \mu_\varnothing(b) = \mu_\varnothing(c) = 0$. Due to $\mu_\varnothing \not\models \Phi$ (where $\Phi = (a \wedge \neg b) \vee c$ as above), we get $\varnothing \not\models \Phi$. However, we have $\varnothing \models \neg a \wedge \neg b$ since $\neg a$ and $\neg b$ hold for the associated evaluation $\mu_\varnothing$.

Semantic Equivalence Two propositional logic formulae Φ, Ψ are called *(semantically) equivalent* if they have the same truth-value for each evaluation. That is, for all evaluations μ:

rule for double negation $\neg\neg\Phi \equiv \Phi$	*idempotency law* $\Phi \vee \Phi \equiv \Phi$ $\Phi \wedge \Phi \equiv \Phi$
absorption law $\Phi \wedge (\Psi \vee \Phi) \equiv \Phi$ $\Phi \vee (\Psi \wedge \Phi) \equiv \Phi$	*commutativity law* $\Phi \wedge \Psi \equiv \Psi \wedge \Phi$ $\Phi \vee \Psi \equiv \Psi \vee \Phi$
associativity law $\Phi \wedge (\Psi \wedge \Xi) \equiv (\Phi \wedge \Psi) \wedge \Xi$ $\Phi \vee (\Psi \vee \Xi) \equiv (\Phi \vee \Psi) \vee \Xi$	*de Morgan's law* $\neg(\Phi \wedge \Psi) \equiv \neg\Phi \vee \neg\Psi$ $\neg(\Phi \vee \Psi) \equiv \neg\Phi \wedge \neg\Psi$
distributivity law $\Phi \vee (\Psi_1 \wedge \Psi_2) \equiv (\Phi \vee \Psi_1) \wedge (\Phi \vee \Psi_2)$ $\Phi \wedge (\Psi_1 \vee \Psi_2) \equiv (\Phi \wedge \Psi_1) \vee (\Phi \wedge \Psi_2)$	

Figure A.3: Some equivalence rules for propositional logic.

$$\mu \models \Phi \text{ iff } \mu \models \Psi.$$

In this case we write $\Phi \equiv \Psi$. For example, the formulae $a \wedge \neg\neg b$ and $a \wedge b$ are semantically equivalent. A few of the most important equivalence rules for propositional logic and the operators $\neg$, $\wedge$ and $\vee$ are shown in Figure A.3. Here, the Greek capital letters Φ, Ψ, Ξ (with or without subscripts) serve as metasymbols for formulae of propositional logic. The associativity and commutativity law for disjunction $\vee$ and conjunction $\wedge$ justify the omission of brackets and notations like

$$\bigwedge_{1 \leqslant i \leqslant n} \Phi_i \quad \text{or} \quad \Phi_1 \wedge \ldots \wedge \Phi_n \quad .$$

Note that the length of a formula of type $\bigwedge_{1 \leqslant i \leqslant n} \Phi_i$ is equal to $n-1$ (and not to 1). Furthermore, notations like $\bigwedge_{i \in I} \Phi_i$ or $\bigwedge\{\Phi_i \mid i \in I\}$ are often used where I is an arbitrary finite index set. If I is nonempty, then Φ stands for one of the formulae $\Phi_{i_1} \wedge \ldots \wedge \Phi_{i_k}$ where $I = \{i_1, \ldots, i_k\}$ and $i_1, \ldots, i_k$ are pairwise different. For $I = \varnothing$, the convention is

$$\bigwedge_{i \in \varnothing} \Phi_i \overset{\text{def}}{=} \text{true}, \quad \text{while} \quad \bigvee_{i \in \varnothing} \Phi_i \overset{\text{def}}{=} \text{false}.$$

Satisfiability and Validity Propositional formula Φ is called *satisfiable* if there is an evaluation μ with $\mu \models \Phi$. Φ is called *valid* (or a *tautology*) if $\mu \models \Phi$ for each evaluation μ. Φ is *unsatisfiable* if Φ is not satisfiable. For example, $a \wedge \neg a$ is unsatisfiable, while

$a \vee \neg(a \wedge b)$ is a tautology. The formulae $a \vee \neg b$ and $a \wedge \neg b$ are satisfiable, but not tautologies. Obviously:

$$\begin{array}{ll} & \Phi \text{ is unsatisfiable} \\ \text{iff} & \mu \not\models \Phi \text{ for all evaluations } \mu \\ \text{iff} & \mu \models \neg\Phi \text{ for all evaluations } \mu \\ \text{iff} & \neg\Phi \text{ is valid.} \end{array}$$

Thus, Φ is unsatisfiable if and only if $\neg\Phi$ is a tautology.

Literals and Positive Normal Form (PNF) A *literal* means a formula of the form a or $\neg a$ where $a \in AP$ is an atomic proposition. Propositional formulae in *positive normal form* (PNF for short, also sometimes called *negation normal form*) use the negation operator only on the level of literals. To ensure that the class of PNF formulae is as expressive as full propositional logic, both the conjunction and the disjunction operator serve as basic operators. Thus, the abstract syntax of PNF formulae is

$$\Phi \quad ::= \quad \text{true} \;\Big|\; \text{false} \;\Big|\; a \;\Big|\; \neg a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \Phi_1 \vee \Phi_2$$

where $a \in AP$. Given a (non-PNF) formula Φ, de Morgan's laws and the rule for double negation allow for "pushing the negation inside" until an equivalent formula in PNF arises. That is, successive application of the transformations

$$\begin{array}{lcl} \neg(\Phi_1 \wedge \Phi_2) & \rightsquigarrow & \neg\Phi_1 \vee \neg\Phi_2 \\ \neg(\Phi_1 \vee \Phi_2) & \rightsquigarrow & \neg\Phi_1 \wedge \neg\Phi_2 \\ \neg\neg\Psi & \rightsquigarrow & \Psi \end{array}$$

to Φ's subformulae yields an equivalent formula in PNF of the same asymptotic length.

Conjunctive and Disjunctive Normal Form Special cases of PNF are the *conjunctive normal form* (CNF) and *disjunctive normal form* (DNF). A CNF formula has the form

$$\bigwedge_{i \in I} \bigvee_{j \in J_i} lit_{i,j}$$

where I and J_i are arbitrary finite index sets and $lit_{i,j}$ for $i \in I$ and $j \in J_i$ are literals. The subformulae $\bigvee_{j \in J_i} lit_{i,j}$ are called *clauses*. For instance, $(a_1 \vee \neg a_3 \vee a_4) \wedge (\neg a_2 \vee \neg a_3 \vee a_4) \wedge a_4$ is a CNF formula with three clauses. Note that, e.g., true and false are also representable by CNF formulae: true is obtained by $I = \varnothing$, while false is equivalent to $a \wedge \neg a$, a CNF with two clauses consisting of one literal each. Given a PNF formula Φ, an equivalent CNF formula is obtained (on the basis of the distributivity laws) by applying the transformation rules:

$$\begin{array}{lcl} \Phi_0 \vee (\Psi_1 \wedge \Psi_2) & \rightsquigarrow & (\Phi_0 \vee \Psi_1) \wedge (\Phi_0 \vee \Psi_2) \\ (\Psi_1 \wedge \Psi_2) \vee \Phi_0 & \rightsquigarrow & (\Psi_1 \vee \Phi_0) \wedge (\Psi_2 \vee \Phi_0) \end{array}$$

to Φ's subformulae as long as possible. Thus, for each propositional formula Φ there exists an equivalent CNF formula Φ'. Similarly, DNF formulae are formulae of the form

$$\bigvee_{i \in I} \bigwedge_{j \in J_i} lit_{i,j}$$

where I and J_i are arbitrary finite index sets and $lit_{i,j}$ for $i \in I$ and $j \in J_i$ are literals. E.g., $(a_1 \wedge \neg a_2) \vee (\neg a_2 \wedge \neg a_3 \wedge a_4) \vee (\neg a_1 \wedge \neg a_3)$ is a DNF formula. A transformation similar to the one for CNF can be applied to prove that any propositional formula Φ has an equivalent DNF formula Φ'.

A.4 Graphs

In most chapters of this monograph, the reader is supposed to be familiar with the basic principles of graph theory and elementary graph algorithms. These are supposed to be known from basic courses. This section is a brief summary of the terms and concepts that are central to this monograph. The details can be found in any elementary textbook on algorithms and data structures; see, e.g. [24, 100], or textbooks that provide an introduction to graph theory, see, e.g. [396].

Digraphs (Graphs, for short) A digraph (or directed graph, in the following called *graph*) is a pair $G = (V, E)$ consisting of a set V of *vertices* and an edge relation $E \subseteq V \times V$. The elements of E are called *edges*. G is called finite if the set V (and hence also E) is finite. The basic models considered in this monograph will rely on graphs where the vertices and edges are augmented with certain information. E.g., instead of defining E as a set of vertex pairs, we may also deal with labeled edges, in which case E is a subset of $V \times \Sigma \times V$ for a certain alphabet Σ. If this additional information is irrelevant, the edge labels may be omitted and rather than considering $E \subseteq V \times \Sigma \times V$ we use $E' = \{ (v, w) \mid \text{there exists an edge } (v, \sigma, w) \in E \}$ to obtain a digraph in the above sense.

The following explanations refer to the case of a digraph $G = (V, E)$ where E is a binary relation over the vertices. For $v \in V$, let $Post_G(v)$, or briefly $Post(v)$, denote the set of direct successors of v; that is, $Post(v) = \{ w \in V \mid (v, w) \in E \}$. Similarly, $Pre_G(v)$, or briefly $Pre(v)$, stands for the set of direct predecessors of v, i.e., $Pre(v) = \{ w \in V \mid (w, v) \in E \}$. A vertex v is called *terminal* if $Post(v) = \varnothing$. Edges of the form (v, v) are called *self-loops*.

Paths in Graphs A *path* in G denotes a (finite or infinite) non-empty sequence of vertices $\hat{\pi} = v_0\, v_1 \ldots v_r$ with $r \geqslant 0$ or $\pi = v_0\, v_1\, v_2 \ldots$ such that $v_{i+1} \in Post(v_i)$, $i = 0, 1, 2, \ldots$. (In