We use set notation ($\in$, $\subseteq$, $\cup$, $\cap$, etc.) with analogous meaning also for multisets, e. g.,

$$\begin{aligned}
m \in S &\;:\Leftrightarrow\; S(m) > 0 \\
(S_1 \cup S_2)(m) &\;:=\; S_1(m) + S_2(m) \\
(S_1 \cap S_2)(m) &\;:=\; \min\{S_1(m), S_2(m)\} \\
(S_1 - S_2)(m) &\;:=\; \begin{cases} S_1(m) - S_2(m) & \text{if } S_1(m) \geq S_2(m) \\ 0 & \text{otherwise} \end{cases} \\
S_1 \subseteq S_2 &\;:\Leftrightarrow\; S_1(m) \leq S_2(m) \text{ for all } m \in M
\end{aligned}$$

A multiset S is called *finite*, if

$$|\{\, m \in M \mid S(m) > 0 \,\}| < \infty.$$

From now on we only consider finite multisets.

Multiset Orderings

Let $(M, \succ)$ be an abstract reduction system. The *multiset extension* of $\succ$ to multisets over M is defined by

$S_1 \succ_{\text{mul}} S_2$ if and only if
there exist multisets X and Y over M such that
$\emptyset \neq X \subseteq S_1$,
$S_2 = (S_1 - X) \cup Y$,
$\forall y \in Y\ \exists x \in X\colon x \succ y$

Lemma 1.9 (König's Lemma) *Every finitely branching tree with infinitely many nodes contains an infinite path.*

Theorem 1.10
(a) If $\succ$ is transitive, then $\succ_{\text{mul}}$ is transitive.
(b) If $\succ$ is irreflexive and transitive, then $\succ_{\text{mul}}$ is irreflexive.
(c) If $\succ$ is a well-founded ordering, then $\succ_{\text{mul}}$ is a well-founded ordering.
(d) If $\succ$ is a strict total ordering, then $\succ_{\text{mul}}$ is a strict total ordering.

Proof. see Baader and Nipkow, page 22–24. □

The multiset extension as defined above is due to Dershowitz and Manna (1979).

There are several other ways to characterize the multiset extension of a binary relation. The following one is due to Huet and Oppen (1980):

Let $(M, \succ)$ be an abstract reduction system. The *(Huet/Oppen) multiset extension* of $\succ$ to multisets over M is defined by

$$
\begin{array}{l}
S_1 \succ_{\text{mul}}^{\text{HO}} S_2 \text{ if and only if} \\
\quad S_1 \neq S_2 \text{ and} \\
\quad\quad \forall m \in M\colon \big(S_2(m) > S_1(m) \\
\quad\quad\quad\quad \Rightarrow\ \exists m' \in M\colon m' \succ m \text{ and } S_1(m') > S_2(m')\big)
\end{array}
$$

A third way to characterize the multiset extension of a binary relation $\succ$ is to define it as the transitive closure of the relation $\succ_{\text{mul}}^{1}$ given by

$$
\begin{array}{l}
S_1 \succ_{\text{mul}}^{1} S_2 \text{ if and only if} \\
\quad \text{there exists } x \in S_1 \text{ and a multiset } Y \text{ over } M \text{ such that} \\
\quad\quad S_2 = (S_1 - \{x\}) \cup Y, \\
\quad\quad \forall y \in Y\colon x \succ y
\end{array}
$$

For strict partial orderings all three characterizations of $\succ_{\text{mul}}$ are equivalent:

Theorem 1.11 *If $\succ$ is a strict partial ordering, then*
(a) $\succ_{\text{mul}} = \succ_{\text{mul}}^{\text{HO}}$,
(b) $\succ_{\text{mul}} = (\succ_{\text{mul}}^{1})^{+}$.

Proof. (a) see Baader and Nipkow, page 24–26. (b) Exercise. □

Note, however, that for an arbitrary binary relation $\succ$ all three relations $\succ_{\text{mul}}$, $\succ_{\text{mul}}^{\text{HO}}$, and $(\succ_{\text{mul}}^{1})^{+}$ may be different.

1.5 Complexity Theory Prerequisites

A *decision problem* is a subset $L \subseteq \Sigma^*$ for some fixed finite alphabet Σ.

The function $\text{chr}(L, x)$ denotes the *characteristic function* for some decision problem L and is defined by $\text{chr}(L, u) = 1$ if $u \in L$ and $\text{chr}(L, u) = 0$ otherwise.

P and NP

A decision problem is called *solvable in polynomial time* if its characteristic function can be computed in polynomial time. The class P denotes all polynomial-time decision problems.

We say that a decision problem L is in *NP* if there is a predicate $Q(x, y)$ and a polynomial $p(n)$ such that for all $u \in \Sigma^*$ we have

(i) $u \in L$ if and only if there is a $v \in \Sigma^*$ with $|v| \leq p(|u|)$ and $Q(u, v)$ holds, and

(ii) the predicate Q is in P.

Reducibility, NP-Hardness, NP-Completeness

A decision problem L is *polynomial-time reducible* to a decision problem L' if there is a function g computable in polynomial time such that for all $u \in \Sigma^*$ we have $u \in L$ iff $g(u) \in L'$.

For example, if L is polynomial-time reducible to L' and $L' \in$ P then $L \in$ P.

A decision problem is *NP-hard* if every problem in NP is polynomial-time reducible to it.

A decision problem is *NP-complete* if it is NP-hard and in NP.

2 Propositional Logic

Propositional logic

- logic of truth values
- decidable (but NP-complete)
- can be used to describe functions over a finite domain
- industry standard for many analysis/verification tasks (e. g., model checking),
- growing importance for discrete optimization problems

2.1 Syntax

- propositional variables
- logical connectives
  $\Rightarrow$ Boolean combinations

Propositional Variables

Let Π be a set of *propositional variables*.

We use letters P, Q, R, S, to denote propositional variables.

Propositional Formulas

F_Π is the set of propositional formulas over Π defined inductively as follows:

$$
\begin{array}{rcll}
F, G & ::= & \bot & \text{(falsum)} \\
& \mid & \top & \text{(verum)} \\
& \mid & P, \quad P \in \Pi & \text{(atomic formula)} \\
& \mid & (\neg F) & \text{(negation)} \\
& \mid & (F \wedge G) & \text{(conjunction)} \\
& \mid & (F \vee G) & \text{(disjunction)} \\
& \mid & (F \rightarrow G) & \text{(implication)} \\
& \mid & (F \leftrightarrow G) & \text{(equivalence)}
\end{array}
$$

Notational Conventions

As a notational convention we assume that $\neg$ binds strongest, and we remove outermost parentheses, so $\neg P \vee Q$ is actually a shorthand for $((\neg P) \vee Q)$.

Instead of $((P \wedge Q) \wedge R)$ we simply write $P \wedge Q \wedge R$ (and analogously for $\vee$).

For all other logical connectives we will use parentheses when needed.

Formula Manipulation

Automated reasoning is very much formula manipulation. In order to precisely represent the manipulation of a formula, we introduce positions.

A *position* is a word over $\mathbb{N}$. The set of positions of a formula F is inductively defined by

$$\begin{aligned} \mathrm{pos}(F) &:= \{\varepsilon\} \text{ if } F \in \{\top, \bot\} \text{ or } F \in \Pi \\ \mathrm{pos}(\neg F) &:= \{\varepsilon\} \cup \{\, 1p \mid p \in \mathrm{pos}(F) \,\} \\ \mathrm{pos}(F \circ G) &:= \{\varepsilon\} \cup \{\, 1p \mid p \in \mathrm{pos}(F) \,\} \cup \{\, 2p \mid p \in \mathrm{pos}(G) \,\} \\ &\quad \text{where } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}. \end{aligned}$$

The prefix order $\leq$ on positions is defined by $p \leq q$ if there is some p' such that $pp' = q$.

Note that the prefix order is partial, e.g., the positions 12 and 21 are not comparable, they are "parallel", see below.

By $<$ we denote the strict part of $\leq$, that is, $p < q$ if $p \leq q$ but not $q \leq p$.

By $\parallel$ we denote incomparable positions, that is, $p \parallel q$ if neither $p \leq q$ nor $q \leq p$.

We say that p is *above* q if $p \leq q$, p is *strictly above* q if $p < q$, and p and q are *parallel* if $p \parallel q$.

The *size* of a formula F is given by the cardinality of $\mathrm{pos}(F)$: $|F| := |\mathrm{pos}(F)|$.

The *subformula* of F at position $p \in \mathrm{pos}(F)$ is recursively defined by

$$\begin{aligned} F|_\varepsilon &:= F \\ (\neg F)|_{1p} &:= F|_p \\ (F_1 \circ F_2)|_{ip} &:= F_i|_p \quad \begin{array}{l} \text{where } i \in \{1, 2\} \\ \text{and } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}. \end{array} \end{aligned}$$

Finally, the *replacement* of a subformula at position $p \in \mathrm{pos}(F)$ by a formula G is recursively defined by

$$\begin{aligned} F[G]_\varepsilon &:= G \\ (\neg F)[G]_{1p} &:= \neg(F[G]_p) \\ (F_1 \circ F_2)[G]_{1p} &:= (F_1[G]_p \circ F_2) \\ (F_1 \circ F_2)[G]_{2p} &:= (F_1 \circ F_2[G]_p) \\ &\qquad \text{where } \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}. \end{aligned}$$

Example 2.1 *The set of positions for the formula* $F = (P \rightarrow Q) \rightarrow (P \wedge \neg Q)$ *is* $\mathrm{pos}(F) = \{\varepsilon, 1, 11, 12, 2, 21, 22, 221\}$.

The subformula at position 22 *is* $F|_{22} = \neg Q$ *and replacing this formula by* $P \leftrightarrow Q$ *results in* $F[P \leftrightarrow Q]_{22} = (P \rightarrow Q) \rightarrow (P \wedge (P \leftrightarrow Q))$.

Polarities

A further prerequisite for efficient formula manipulation is the polarity of a subformula G of F. The polarity determines the number of "negations" starting from F down to G. It is 1 for an even number, -1 for an odd number and 0 if there is at least one equivalence connective along the path.

The *polarity* of a subformula $G = F|_p$ at position p is $\mathrm{pol}(F, p)$, where pol is recursively defined by

$$\begin{aligned} \mathrm{pol}(F, \varepsilon) &:= 1 \\ \mathrm{pol}(\neg F, 1p) &:= -\mathrm{pol}(F, p) \\ \mathrm{pol}(F_1 \circ F_2, ip) &:= \mathrm{pol}(F_i, p) \text{ if } \circ \in \{\wedge, \vee\} \\ \mathrm{pol}(F_1 \rightarrow F_2, 1p) &:= -\mathrm{pol}(F_1, p) \\ \mathrm{pol}(F_1 \rightarrow F_2, 2p) &:= \mathrm{pol}(F_2, p) \\ \mathrm{pol}(F_1 \leftrightarrow F_2, ip) &:= 0 \end{aligned}$$

Example 2.2 *Let* $F = (P \rightarrow Q) \rightarrow (P \wedge \neg Q)$. *Then* $\mathrm{pol}(F, 1) = \mathrm{pol}(F, 12) = \mathrm{pol}(F, 221) = -1$ *and* $\mathrm{pol}(F, \varepsilon) = \mathrm{pol}(F, 11) = \mathrm{pol}(F, 2) = \mathrm{pol}(F, 21) = \mathrm{pol}(F, 22) = 1$.

For the formula $F' = (P \wedge Q) \leftrightarrow (P \vee Q)$ *we get* $\mathrm{pol}(F', \varepsilon) = 1$ *and* $\mathrm{pol}(F', p) = 0$ *for all* $p \in \mathrm{pos}(F')$ *different from* ε.

2.2 Semantics

In *classical logic* (dating back to Aristotle) there are "only" two truth values "true" and "false" which we shall denote, respectively, by 1 and 0.

There are *multi-valued logics* having more than two truth values.

Valuations

A propositional variable has no intrinsic meaning. The meaning of a propositional variable has to be defined by a valuation.

A Π-*valuation* is a map

$$\mathcal{A} : \Pi \to \{0, 1\}.$$

where $\{0, 1\}$ is the set of *truth values*.

Truth Value of a Formula in $\mathcal{A}$

Given a Π-valuation $\mathcal{A}$, its extension to formulas $\mathcal{A}^* : \mathrm{F}_\Pi \to \{0, 1\}$ is defined inductively as follows:

$$\begin{aligned}
\mathcal{A}^*(\bot) &= 0 \\
\mathcal{A}^*(\top) &= 1 \\
\mathcal{A}^*(P) &= \mathcal{A}(P) \\
\mathcal{A}^*(\neg F) &= 1 - \mathcal{A}^*(F) \\
\mathcal{A}^*(F \wedge G) &= \min(\mathcal{A}^*(F), \mathcal{A}^*(G)) \\
\mathcal{A}^*(F \vee G) &= \max(\mathcal{A}^*(F), \mathcal{A}^*(G)) \\
\mathcal{A}^*(F \to G) &= \max(1 - \mathcal{A}^*(F), \mathcal{A}^*(G)) \\
\mathcal{A}^*(F \leftrightarrow G) &= \text{if } \mathcal{A}^*(F) = \mathcal{A}^*(G) \text{ then 1 else 0}
\end{aligned}$$

For simplicity, the extension $\mathcal{A}^*$ of $\mathcal{A}$ is usually also denoted by $\mathcal{A}$.

2.3 Models, Validity, and Satisfiability

Let F be a Π-formula.

We say that F is *true* under $\mathcal{A}$ ($\mathcal{A}$ is a *model* of F; F *is valid* in $\mathcal{A}$; F *holds* under $\mathcal{A}$), written $\mathcal{A} \models F$, if $\mathcal{A}(F) = 1$.

We say that F is *valid* or that F is a *tautology*, written $\models F$, if $\mathcal{A} \models F$ for all Π-valuations $\mathcal{A}$.

F is called *satisfiable* if there exists an $\mathcal{A}$ such that $\mathcal{A} \models F$. Otherwise F is called *unsatisfiable* (or *contradictory*).

Entailment and Equivalence

F *entails* (*implies*) G (or G *is a consequence of* F), written $F \models G$, if for all Π-valuations $\mathcal{A}$ we have

$$\text{if } \ \mathcal{A} \models F \ \text{ then } \ \mathcal{A} \models G,$$

or equivalently

$$\mathcal{A}(F) \leq \mathcal{A}(G).$$

F and G are called *equivalent*, written $F \models\!\!\models G$, if for all Π-valuations $\mathcal{A}$ we have

$$\mathcal{A} \models F \ \text{ if and only if } \ \mathcal{A} \models G,$$

or equivalently

$$\mathcal{A}(F) = \mathcal{A}(G).$$

Proposition 2.3 *$F \models G$ if and only if $\models (F \to G)$.*

Proof. ($\Rightarrow$) Suppose that F entails G. Let $\mathcal{A}$ be an arbitrary Π-valuation. We have to show that $\mathcal{A} \models F \to G$. If $\mathcal{A}(F) = 1$, then $\mathcal{A}(G) = 1$ (since $F \models G$), and hence $\mathcal{A}(F \to G) = \max(1 - 1, 1) = 1$. Otherwise $\mathcal{A}(F) = 0$, then $\mathcal{A}(F \to G) = \max(1 - 0, \mathcal{A}(G)) = 1$ independently of $\mathcal{A}(G)$. In both cases, $\mathcal{A} \models F \to G$.

($\Leftarrow$) Suppose that F does not entail G. Then there exists a Π-valuation $\mathcal{A}$ such that $\mathcal{A} \models F$, but not $\mathcal{A} \models G$. Consequently, $\mathcal{A}(F \to G) = \max(1 - \mathcal{A}(F), \mathcal{A}(G)) = \max(1 - 1, 0) = 0$, so $(F \to G)$ does not hold under $\mathcal{A}$. $\square$