Definition 2.1.2 (Atom, Literal)**.** A propositional formula P is called an *atom*. It is also called a *(positive) literal* and its negation $\neg P$ is called a *(negative) literal*. If L is a literal, then $\neg L = P$ if $L = \neg P$ and $\neg L = \neg P$ if $L = P$. Literals are denoted by letters L, K. The literals P and $\neg P$ are called *complementary*.

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

Definition 2.1.3 (Position)**.** A *position* is a word over $\mathbb{N}$. The set of positions of a formula ϕ is inductively defined by

$$\begin{array}{rcl} \mathrm{pos}(\phi) & := & \{\epsilon\} \text{ if } \phi \in \{\top, \bot\} \text{ or } \phi \in \Sigma \\ \mathrm{pos}(\neg\phi) & := & \{\epsilon\} \cup \{1p \mid p \in \mathrm{pos}(\phi)\} \\ \mathrm{pos}(\phi \circ \psi) & := & \{\epsilon\} \cup \{1p \mid p \in pos(\phi)\} \cup \{2p \mid p \in \mathrm{pos}(\psi)\} \end{array}$$

where $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

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 $\leq$, i.e., $p < q$ if $p \leq q$ but not $q \leq p$. By $\parallel$ we denote incomparable positions, i.e., $p \parallel q$ if neither $p \leq q$, nor $q \leq p$. Then 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 ϕ is given by the cardinality of $\mathrm{pos}(\phi)$: $|\phi| := |\mathrm{pos}(\phi)|$. The *subformula* of ϕ at position $p \in \mathrm{pos}(\phi)$ is recursively defined by $\phi|_\epsilon := \phi$, $\neg\phi|_{1p} := \phi|_p$, and $(\phi_1 \circ \phi_2)|_{ip} := \phi_i|_p$ where $i \in \{1, 2\}$, $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$. Finally, the *replacement* of a subformula at position $p \in \mathrm{pos}(\phi)$ by a formula ψ is recursively defined by $\phi[\psi]_\epsilon := \psi$ and $(\phi_1 \circ \phi_2)[\psi]_{1p} := (\phi_1[\psi]_p \circ \phi_2)$, $(\phi_1 \circ \phi_2)[\psi]_{2p} := (\phi_1 \circ \phi_2[\psi]_p)$, where $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Example 2.1.4. The set of positions for the formula $\phi = (P \wedge Q) \rightarrow (P \vee Q)$ is $\mathrm{pos}(\phi) = \{\epsilon, 1, 11, 12, 2, 21, 22\}$. The subformula at position 22 is Q, $\phi|_{22} = Q$ and replacing this formula by $P \leftrightarrow Q$ results in $\phi[P \leftrightarrow Q]_{22} = (P \wedge Q) \rightarrow (P \vee (P \leftrightarrow Q))$.

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

Definition 2.1.5 (Polarity)**.** The *polarity* of a subformula of ϕ at position $p \in \mathrm{pos}(\phi)$ is inductively defined by

$$\begin{array}{rcl} \mathrm{pol}(\phi, \epsilon) & := & 1 \\ \mathrm{pol}(\neg\phi, 1p) & := & -\,\mathrm{pol}(\phi, p) \\ \mathrm{pol}(\phi_1 \circ \phi_2, ip) & := & \mathrm{pol}(\phi_i, p) \text{ if } \circ \in \{\wedge, \vee\} \\ \mathrm{pol}(\phi_1 \rightarrow \phi_2, 1p) & := & -\,\mathrm{pol}(\phi_1, p) \\ \mathrm{pol}(\phi_1 \rightarrow \phi_2, 2p) & := & \mathrm{pol}(\phi_2, p) \\ \mathrm{pol}(\phi_1 \leftrightarrow \phi_2, ip) & := & 0 \end{array}$$

Example 2.1.6. We reuse the formula $\phi = (A \wedge B) \rightarrow (A \vee B)$ of Example 2.1.4. Then $\mathrm{pol}(\phi, 1) = \mathrm{pol}(\phi, 11) = -1$ and $\mathrm{pol}(\phi, 2) = \mathrm{pol}(\phi, 22) = 1$. For the formula $\phi' = (A \wedge B) \leftrightarrow (A \vee B)$ we get $\mathrm{pol}(\phi', \epsilon) = 1$ and $\mathrm{pol}(\phi', p) = 0$ for all other $p \in \mathrm{pos}(\phi')$, $p \neq \epsilon$.

2.2 Semantics

In *classical logic* there are two truth values "true" and "false" which we shall denote, respectively, by 1 and 0. There are *many-valued logics* [21] having more than two truth values and in fact, as we will see later on, for the definition of some propositional logic calculi, we will need an implicit third truth value called "undefined".

Definition 2.2.1 ((Partial) Valuation)**.** A Σ-*valuation* is a map

$$\mathcal{A} : \Sigma \rightarrow \{0, 1\}.$$

where $\{0, 1\}$ is the set of *truth values*. A *partial* Σ-*valuation* is a map $\mathcal{A}' : \Sigma' \rightarrow \{0, 1\}$ where $\Sigma' \subseteq \Sigma$.

Definition 2.2.2 (Semantics)**.** A Σ-valuation $\mathcal{A}$ is inductively extended from propositional variables to propositional formulas $\phi, \psi \in \mathrm{PROP}(\Sigma)$ by

$$\begin{array}{rcl}
\mathcal{A}(\bot) & := & 0 \\
\mathcal{A}(\top) & := & 1 \\
\mathcal{A}(\neg\phi) & := & 1 - \mathcal{A}(\phi) \\
\mathcal{A}(\phi \wedge \psi) & := & \min(\{\mathcal{A}(\phi), \mathcal{A}(\psi)\}) \\
\mathcal{A}(\phi \vee \psi) & := & \max(\{\mathcal{A}(\phi), \mathcal{A}(\psi)\}) \\
\mathcal{A}(\phi \rightarrow \psi) & := & \max(\{(1 - \mathcal{A}(\phi)), \mathcal{A}(\psi)\}) \\
\mathcal{A}(\phi \leftrightarrow \psi) & := & \text{if } \mathcal{A}(\phi) = \mathcal{A}(\psi) \text{ then 1 else 0}
\end{array}$$

If $\mathcal{A}(\phi) = 1$ for some Σ-valuation $\mathcal{A}$ of a formula ϕ then ϕ is *satisfiable* and we write $\mathcal{A} \models \phi$. If $\mathcal{A}(\phi) = 1$ for all Σ-valuations $\mathcal{A}$ of a formula ϕ then ϕ is *valid* and we write $\models \phi$. If there is no Σ-valuations $\mathcal{A}$ for a formula ϕ where $\mathcal{A}(\phi) = 1$ we say ϕ is *unsatisfiable*. A formula ϕ *entails* ψ, written $\phi \models \psi$, if for all Σ-valuations $\mathcal{A}$ whenever $\mathcal{A} \models \phi$ then $\mathcal{A} \models \psi$.

Accordingly, a formula ϕ is satisfiable, valid, unsatisfiable, respectively, with respect to a partial valuation $\mathcal{A}'$ with domain Σ', if for any valuation $\mathcal{A}$ with $\mathcal{A}(P) = \mathcal{A}'(P)$ for all $P \in \Sigma'$ the formula ϕ is satisfiable, valid, unsatisfiable, respectively, with respect to a $\mathcal{A}$.

I call the fact that some formula ϕ is satisfiable, unsatisfiable, or valid, the *status* of ϕ. Note that if ϕ is valid it is also satisfiable, but not the other way round.

Valuations can be nicely represented by sets or sequences of literals that do not contain complementary literals nor duplicates. If $\mathcal{A}$ is a (partial) valuation of domain Σ then it can be represented by the set $\{P \mid P \in \Sigma \text{ and } \mathcal{A}(P) = 1\} \cup \{\neg P \mid P \in \Sigma \text{ and } \mathcal{A}(P) = 0\}$. For example, for the valuation $\mathcal{A} = \{P, \neg Q\}$

the truth value of $P \vee Q$ is $\mathcal{A}(P \vee Q) = 1$, for $P \vee R$ it is $\mathcal{A}(P \vee R) = 1$, for $\neg P \wedge R$ it is $\mathcal{A}(\neg P \wedge R) = 0$, and the status of $\neg P \vee R$ cannot be established by $\mathcal{A}$. In particular, $\mathcal{A}$ is a partial valuation for $\Sigma = \{P, Q, R\}$.

Example 2.2.3. The formula $\phi \vee \neg\phi$ is valid, independently of ϕ. According to Definition 2.2.2 we need to prove that for all Σ-valuations $\mathcal{A}$ of ϕ we have $\mathcal{A}(\phi \vee \neg\phi) = 1$. So let $\mathcal{A}$ be an arbitrary valuation. There are two cases to consider. If $\mathcal{A}(\phi) = 1$ then $\mathcal{A}(\phi \vee \neg\phi) = 1$ because the valuation function takes the maximum if distributed over $\vee$. If $\mathcal{A}(\phi) = 0$ then $\mathcal{A}(\neg\phi) = 1$ and again by the before argument $\mathcal{A}(\phi \vee \neg\phi) = 1$. This finishes the proof that $\models \phi \vee \neg\phi$.

Proposition 2.2.4. $\phi \models \psi$ iff $\models \phi \to \psi$

Proof. ($\Rightarrow$) Suppose that ϕ entails ψ and let $\mathcal{A}$ be an arbitrary Σ-valuation. We need to show $\mathcal{A} \models \phi \to \psi$. If $\mathcal{A}(\phi) = 1$, then $\mathcal{A}(\psi) = 1$, because ϕ entails ψ, and therefore $\mathcal{A} \models \phi \to \psi$. For otherwise, if $\mathcal{A}(\phi) = 0$, then $\mathcal{A}(\phi \to \psi) = \max(\{(1 - \mathcal{A}(\phi)), \mathcal{A}(\psi)\}) = \max(\{(1, \mathcal{A}(\psi)\}) = 1$, independently of the value of $\mathcal{A}(\psi)$. In both cases $\mathcal{A} \models \phi \to \psi$.
($\Leftarrow$) By contraposition. Suppose that ϕ does not entail ψ. Then there exists a Σ-valuation $\mathcal{A}$ such that $\mathcal{A} \models \phi$, $\mathcal{A}(\phi) = 1$ but $\mathcal{A} \not\models \psi$, $\mathcal{A}(\psi) = 0$. By definition, $\mathcal{A}(\phi \to \psi) = \max(\{(1 - \mathcal{A}(\phi)), \mathcal{A}(\psi)\}) = \max(\{(1 - 1), 0\}) = 0$, hence $\phi \to \psi$ does not hold in $\mathcal{A}$. □

Proposition 2.2.5. The equivalences of Figure 2.1 are valid for all formulas ϕ, ψ, χ.

From Figure 2.1 we conclude that the propositional language introduced in Definition 2.1.1 is redundant in the sense that certain connectives can be expressed by others. For example, the equivalence Eliminate $\to$ expresses implication by means of disjunction and negation. So for any propositional formula ϕ there exists an equivalent formula ϕ' such that ϕ' does not contain the implication connective. In order to prove this proposition we need the below replacement lemma.

Note that the formulas $\phi \wedge \psi$ and $\psi \wedge \phi$ are equivalent. Nevertheless, recalling the problem state definition for Sudokus in Section 1.1 the two states $(N; f(2,3) = 1 \wedge f(2,4) = 4; \top)$ and $(N; f(2,4) = 4 \wedge f(2,3) = 1; \top)$ are significantly different. For example, it can be that the first state can lead to a solution by the rules of the algorithm where the latter cannot, because the latter implicitly means that the square $(2,4)$ has already been checked for all values smaller than 4. This reveals the important point that arguing by logical equivalence in the context of a rule set manipulating formulas can lead to wrong results.

Lemma 2.2.6 (Formula Replacement)**.** Let ϕ be a propositional formula containing a subformula ψ at position p, i.e., $\phi|_p = \psi$. Furthermore, assume $\models \psi \leftrightarrow \chi$. Then $\models \phi \leftrightarrow \phi[\chi]_p$.

(I)	$(\phi \wedge \phi) \leftrightarrow \phi$	Idempotency $\wedge$
	$(\phi \vee \phi) \leftrightarrow \phi$	Idempotency $\vee$
(II)	$(\phi \wedge \psi) \leftrightarrow (\psi \wedge \phi)$	Commutativity $\wedge$
	$(\phi \vee \psi) \leftrightarrow (\psi \vee \phi)$	Commutativity $\vee$
(III)	$(\phi \wedge (\psi \wedge \chi)) \leftrightarrow ((\phi \wedge \psi) \wedge \chi)$	Associativity $\wedge$
	$(\phi \vee (\psi \vee \chi)) \leftrightarrow ((\phi \vee \psi) \vee \chi)$	Associativity $\vee$
(IV)	$(\phi \wedge (\psi \vee \chi)) \leftrightarrow (\phi \wedge \psi) \vee (\phi \wedge \chi)$	Distributivity $\wedge\vee$
	$(\phi \vee (\psi \wedge \chi)) \leftrightarrow (\phi \vee \psi) \wedge (\phi \vee \chi)$	Distributivity $\vee\wedge$
(V)	$(\phi \wedge (\phi \vee \psi)) \leftrightarrow \phi$	Absorption $\wedge\vee$
	$(\phi \vee (\phi \wedge \psi)) \leftrightarrow \phi$	Absorption $\vee\wedge$
(VI)	$\neg(\phi \vee \psi) \leftrightarrow (\neg\phi \wedge \neg\psi)$	De Morgan $\neg\vee$
	$\neg(\phi \wedge \psi) \leftrightarrow (\neg\phi \vee \neg\psi)$	De Morgan $\neg\wedge$
(VII)	$(\phi \wedge \neg\phi) \leftrightarrow \bot$	Introduction $\bot$
	$(\phi \vee \neg\phi) \leftrightarrow \top$	Introduction $\top$
	$\neg\top \leftrightarrow \bot$	Propagate $\neg\top$
	$\neg\bot \leftrightarrow \top$	Propagate $\neg\bot$
	$(\phi \wedge \top) \leftrightarrow \phi$	Absorption $\top\wedge$
	$(\phi \vee \bot) \leftrightarrow \phi$	Absorption $\bot\vee$
	$(\phi \rightarrow \bot) \leftrightarrow \neg\phi$	Eliminate $\rightarrow \bot$
	$(\bot \rightarrow \phi) \leftrightarrow \top$	Eliminate $\bot \rightarrow$
	$(\phi \rightarrow \top) \leftrightarrow \top$	Eliminate $\rightarrow \top$
	$(\top \rightarrow \phi) \leftrightarrow \phi$	Eliminate $\top \rightarrow$
	$(\phi \leftrightarrow \bot) \leftrightarrow \neg\phi$	Eliminate $\bot \leftrightarrow$
	$(\phi \leftrightarrow \top) \leftrightarrow \phi$	Eliminate $\top \leftrightarrow$
	$(\phi \vee \top) \leftrightarrow \top$	Propagate $\top$
	$(\phi \wedge \bot) \leftrightarrow \bot$	Propagate $\bot$
(VIII)	$(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$	Eliminate $\rightarrow$
(IX)	$(\phi \leftrightarrow \psi) \leftrightarrow (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$	Eliminate1 $\leftrightarrow$
	$(\phi \leftrightarrow \psi) \leftrightarrow (\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$	Eliminate2 $\leftrightarrow$

Figure 2.1: Valid Propositional Equivalences

Proof. By induction on $|p|$ and structural induction on ϕ. For the base step let $p = \epsilon$ and $\mathcal{A}$ be an arbitrary valuation.

$$\begin{aligned} \mathcal{A}(\phi) &= \mathcal{A}(\psi) && \text{(by definition of replacement)} \\ &= \mathcal{A}(\chi) && \text{(because } \mathcal{A} \models \psi \leftrightarrow \chi) \\ &= \mathcal{A}(\phi[\chi]_\epsilon) && \text{(by definition of replacement)} \end{aligned}$$

For the induction step the lemma holds for all positions p and has to be shown for all positions ip. By structural induction on ϕ I show the cases where $\phi = \neg\phi_1$ and $\phi = \phi_1 \rightarrow \phi_2$ in detail. All other cases are analogous.

If $\phi = \neg\phi_1$ then showing the lemma amounts to proving $\models \neg\phi_1 \leftrightarrow \neg\phi_1[\chi]_{1p}$.

Let $\mathcal{A}$ be an arbitrary valuation.

$$\begin{aligned}\mathcal{A}(\neg\phi_1) &= 1 - \mathcal{A}(\phi_1) && \text{(expanding semantics)}\\ &= 1 - \mathcal{A}(\phi_1[\chi]_p) && \text{(by induction hypothesis)}\\ &= \mathcal{A}(\neg\phi[\chi]_{1p}) && \text{(applying semantics)}\end{aligned}$$

If $\phi = \phi_1 \to \phi_2$ then showing the lemma amounts to proving the two cases $\models (\phi_1 \to \phi_2) \leftrightarrow (\phi_1 \to \phi_2)[\chi]_{1p}$ and $\models (\phi_1 \to \phi_2) \leftrightarrow (\phi_1 \to \phi_2)[\chi]_{2p}$. Both cases are similar so I show only the first case. Let $\mathcal{A}$ be an arbitrary valuation.

$$\begin{aligned}\mathcal{A}(\phi_1 \to \phi_2) &= \max(\{(1 - \mathcal{A}(\phi_1)), \mathcal{A}(\phi_2)\}) && \text{(expanding semantics)}\\ &= \max(\{(1 - \mathcal{A}(\phi_1[\chi]_p)), \mathcal{A}(\phi_2)\}) && \text{(by induction hypothesis)}\\ &= \mathcal{A}((\phi_1 \to \phi_2)[\chi]_{1p}) && \text{(applying semantics)}\end{aligned}$$

□

C The equivalences of Figure 2.1 show that the propositional language introduced in Definition 2.1.1 is redundant in the sense that certain connectives can be expressed by others. For example, the equivalence Eliminate $\to$ expresses implication by means of disjunction and negation. So for any propositional formula ϕ there exists an equivalent formula ϕ' such that ϕ' does not contain the implication connective. In order to prove this proposition the above replacement lemma is key.

2.3 Abstract Properties of Calculi

A proof procedure can be *sound*, *complete*, *strongly complete*, *refutationally complete* or *terminating*. Terminating means that it terminates on any input formula. Now depending on whether the calculus investigates validity (unsatisfiability) or satisfiability the before notions have a different meaning.

	Validity	Satisfiability
Sound	Whenever the calculus outputs a proof the formula is valid.	Whenever the calculus outputs a model the formula has a model.
Complete	If the formula is valid the calculus outputs a proof.	If the formula is satisfiable, the calculus outputs a model.
Strongly Complete	For any proof of the formula, there is a sequence of rule applications that generates this proof.	For any model of the formula, there is a sequence of rule applications that generates this model.

There are some assumptions underlying these informal definitions. First, the calculus actually produces a proof in case of investigating validity, and in case of investigating satisfiability it produces a model. This in fact requires the notion of a proof and a model. Then soundness means in both cases that the calculus has no bugs. The results it produces are correct. Completeness means that if there is a proof (model) for a formula, the calculus will eventually find it. Strong completeness requires in addition that any proof (model) can be found by the calculus. A variant of complete calculus is a *refutationally complete* calculus: a calculus is refutationally complete, if for any unsatisfiable formula it outputs a proof of contradiction. Many automated theorem procedures like resolution (see Section 2.7), or tableau (see Section 2.5) are actually only refutationally complete.

Note that soundness and completeness are not closely related to termination. A sound and complete (strongly) complete calculus needs not to be terminating. For example, while investigating validity of an invalid formula, a sound and complete calculus for validity may not terminate. A sound and terminating procedure needs not to be complete. It can simply terminate, "giving up", without producing a proof (model).

2.4 Truth Tables

The first calculus I consider are truth tables. For example, consider proving validity of the formula $\phi = (A \wedge B) \rightarrow A$. According to Definition 2.2.2 this is the case if actually for all valuations $\mathcal{A}$ over $\Sigma = \{A, B\}$ we have $\mathcal{A}(\phi) = 1$. The extension of $\mathcal{A}$ to formulas is defined inductively over the connectives, so if the result of $\mathcal{A}$ on the arguments of a connective is known, it can be straightforwardly computed for the overall formula. That's the idea behind truth tables. We simply make all valuations $\mathcal{A}$ on Σ explicit and then extend it connective by connective bottom-up to the overall formula. Stated otherwise, in order to establish the truth value for a formula ϕ we establish it subformula by subformula of ϕ according to $\leq$. If $p, q \in \mathrm{pos}(\phi)$ and $p \leq q$ then we first compute the truth value for $\phi|_q$. The truth table for $(P \wedge Q) \rightarrow P$ is then depicted in Figure 2.2

P	Q	$P \wedge Q$	$(P \wedge Q) \rightarrow P$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	1

Figure 2.2: Truth Table for $(P \wedge Q) \rightarrow P$

Definition 2.4.1 (Truth Table)**.** Let ϕ be a propositional formula over variables $P_1, \ldots, P_n$, $p_i \in \mathrm{pos}(\phi)$, $1 \leq i \leq k$ and $p_k = \epsilon$. Then a *truth table* for ϕ is a table with $n + k$ columns and $2^n + 1$ rows of the form

P_1	...	P_n	$\phi\|_{p_1}$	...	$\phi\|_{p_k}$
0	...	0	$\mathcal{A}_1(\phi\|_{p_1})$	...	$\mathcal{A}_1(\phi\|_{p_k})$
			$\vdots$		
1	...	1	$\mathcal{A}_{2^n}(\phi\|_{p_1})$	...	$\mathcal{A}_{2^n}(\phi\|_{p_k})$

such that the $\mathcal{A}_i$ are exactly the 2^n different valuations for $P_1, \ldots, P_n$ and either $p_i \parallel p_{i+j}$ or $p_i \geq p_{i+j}$, for all $i, j \geq 0$, $i + j \leq k$ and whenever $\phi|_{p_i}$ has a proper subformula ψ that is not an atom, there is exactly one $j < i$ with $\phi|_{p_j} = \psi$.

Now given a truth table for some formula ϕ, ϕ is satisfiable, if there is at least one 1 in the ϕ column. It is valid, if there is no 0 in the ϕ column. It is unsatisfiable, if there is no 1 in the ϕ column. So truth tables are a simple and "easy" way to establish the status of a formula. They need not to be completely computed in order to establish the status of a formula. For example, as soon as the column of ϕ in a truth table contains a 1 and a 0, then ϕ is satisfiable but neither valid nor unsatisfiable.

The formula $(P \vee Q) \leftrightarrow (P \vee R)$ is satisfiable but not valid. Figure 2.3 contains a truth table for the formula.

P	Q	R	$P \vee Q$	$P \vee R$	$(P \vee Q) \leftrightarrow (P \vee R)$
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	1	1	1
1	1	0	1	1	1
0	0	1	0	1	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	1	1	1	1

Figure 2.3: Truth Table for $(P \vee Q) \leftrightarrow (P \vee R)$

Of course, there are cases where a truth table for some formula ϕ can have less columns than the number of variables occurring in ϕ plus the number of subformulas in ϕ. For example, for the formula $\phi = (P \vee Q) \wedge (R \rightarrow (P \vee Q))$ only one column with formula $(P \vee Q)$ is needed for both subformulas $\phi|_1$ and $\phi|_{22}$. In general, there is only for each *different* subformula a column is needed. Detecting subformula equivalence is beneficial. For the above example, this was simply syntactic, i.e., the two subformulas $\phi|_1$ and $\phi|_{22}$. But what about a slight variation of the formula $\phi' = (P \vee Q) \wedge (R \rightarrow (Q \vee P))$? Strictly speaking, now the two subformulas $\phi'|_1$ and $\phi'|_{22}$ are different, but since disjunction is commutative, they are equivalent. One or two columns in the truth table for the two subformulas? Again, saving a column is beneficial but in general, detecting equivalence of two subformulas may become as difficult as checking whether the overall formula is valid. A compromise, often performed in practice, are normal forms that guarantee that certain occurrences of equivalent subformulas can be found in polynomial time. For our example, we can simply assume some

ordering on the propositional variables and assume that for a disjunction of two propositional variables, the smaller variable always comes first. So if $P < Q$ then the normal form of $P \vee Q$ and $Q \vee P$ is in fact $P \vee Q$.

In practice, nobody uses truth tables as a reasoning procedure. Worst case, computing a truth table for checking the status of a formula ϕ requires $O(2^n)$ steps, where n is the number of different propositional variables in ϕ. But this is actually not the reason why the procedure is impractical, because the worst case behavior of all other procedures for propositional logic known today is also of exponential complexity. So why are truth tables not a good procedure? The answer is: because they do not adapt to the inherent structure of a formula. The reasoning mechanism of a truth table for two formulas ϕ and ψ sharing the same propositional variables is exactly the same: we enumerate all valuations. However, if ϕ is, e.g., of the form $\phi = P \wedge \phi'$ and we are interested in the satisfiability of ϕ, then ϕ can only become true for a valuation $\mathcal{A}$ with $\mathcal{A}(P) = 1$. Hence, 2^{n-1} rows of ϕ's truth table are superfluous. All procedures I will introduce in the sequel, automatically detect this (and further) specific structures of a formula and use it to speed up the reasoning process.

2.5 Semantic Tableaux

Like resolution, semantic tableaux were developed in the sixties, independently by Lis [14] and Smullyan [19] on the basis of work by Gentzen in the 30s [11] and of Beth [3] in the 50s. For an at that time state of the art overview consider Fitting's book [10].

In contrast to the calculi introduced in subsequent sections, semantic tableau does not rely on a normal form of input formulas but actually applies to any propositional formula. The formulas are divided into α- and β-formulas, where intuitively an α formula represents a (hidden) conjunction and a β formula a (hidden) disjunction.

Definition 2.5.1 (α-, β-Formulas)**.** A formula ϕ is called an α-formula if ϕ is a formula $\neg\neg\phi_1$, $\phi_1 \wedge \phi_2$, $\phi_1 \leftrightarrow \phi_2$, $\neg(\phi_1 \vee \phi_2)$, or $\neg(\phi_1 \rightarrow \phi_2)$. A formula ϕ is called an β-formula if ϕ is a formula $\phi_1 \vee \phi_2$, $\phi_1 \rightarrow \phi_2$, $\neg(\phi_1 \wedge \phi_2)$, or $\neg(\phi_1 \leftrightarrow \phi_2)$.

A common property of α-, β-formulas is that they can be decomposed into direct descendants representing (modulo negation) subformulas of the respective formulas. Then an α-formula is valid iff all its descendants are valid and a β-formula is valid if one of its descendants is valid. Therefore, the literature uses both the notions semantic tableaux and analytic tableaux.

Definition 2.5.2 (Direct Descendant)**.** Given an α- or β-formula ϕ, Figure 2.4 shows its direct descendants.

Duplicating ϕ for the α-descendants of $\neg\neg\phi$ is a trick for conformity. Any propositional formula is either an α-formula or a β-formula or a literal.

α	Left Descendant	Right Descendant
$\neg\neg\phi$	ϕ	ϕ
$\phi_1 \wedge \phi_2$	ϕ_1	ϕ_2
$\phi_1 \leftrightarrow \phi_2$	$\phi_1 \rightarrow \phi_2$	$\phi_2 \rightarrow \phi_1$
$\neg(\phi_1 \vee \phi_2)$	$\neg\phi_1$	$\neg\phi_2$
$\neg(\phi_1 \rightarrow \phi_2)$	ϕ_1	$\neg\phi_2$

β	Left Descendant	Right Descendant
$\phi_1 \vee \phi_2$	ϕ_1	ϕ_2
$\phi_1 \rightarrow \phi_2$	$\neg\phi_1$	ϕ_2
$\neg(\phi_1 \wedge \phi_2)$	$\neg\phi_1$	$\neg\phi_2$
$\neg(\phi_1 \leftrightarrow \phi_2)$	$\neg(\phi_1 \rightarrow \phi_2)$	$\neg(\phi_2 \rightarrow \phi_1)$

Figure 2.4: α- and β-Formulas

Proposition 2.5.3. For any valuation $\mathcal{A}$: (i) if ϕ is an α-formula then $\mathcal{A}(\phi) = 1$ iff $\mathcal{A}(\phi_1) = 1$ and $\mathcal{A}(\phi_2) = 1$ for its descendants ϕ_1, ϕ_2. (ii) if ϕ is a β-formula then $\mathcal{A}(\phi) = 1$ iff $\mathcal{A}(\phi_1) = 1$ or $\mathcal{A}(\phi_2) = 1$ for its descendants ϕ_1, ϕ_2.

Proof. Exercise **??**. □

The tableaux calculus operates on states that are sets of sequences of formulas. Semantically, the set represents a disjunction of sequences that are interpreted as conjunctions of the respective formulas. A sequence of formulas $(\phi_1, \ldots, \phi_n)$ is called *closed* if there are two formulas ϕ_i and ϕ_j in the sequence where $\phi_i = \neg\phi_j$ or $\neg\phi_i = \phi_j$. A state is *closed* if all its formula sequences are closed. A state actually represents a tree and this tree is called a tableau in the literature. So if a state is closed, the respective tree, the tableau is closed too. The tableaux calculus is a calculus showing unsatisfiability. Such calculi are called *refutational* calculi. Later on soundness and completeness of the calculus imply that a formula ϕ is valid iff the rules of tableaux produce a closed state starting with $N = \{(\neg\phi)\}$.

A formula ϕ occurring in some sequence is called *open* if in case ϕ is an α-formula not both direct descendants are already part of the sequence and if it is a β-formula non of its descendants is part of the sequence.

α-Expansion $N \uplus \{(\phi_1, \ldots, \psi, \ldots, \phi_n)\} \Rightarrow_{\mathrm{T}} N \uplus \{(\phi_1, \ldots, \psi, \ldots, \phi_n, \psi_1, \psi_2)\}$
provided ψ is an open α-formula, ψ_1, ψ_2 its direct descendants and the sequence is not closed.

β-Expansion $N \uplus \{(\phi_1, \ldots, \psi, \ldots, \phi_n)\} \Rightarrow_{\mathrm{T}} N \uplus \{(\phi_1, \ldots, \psi, \ldots, \phi_n, \psi_1)\} \uplus \{(\phi_1, \ldots, \psi, \ldots, \phi_n, \psi_2)\}$
provided ψ is an open β-formula, ψ_1, ψ_2 its direct descendants and the sequence is not closed.

Consider the question of validity of the formula $(P \wedge \neg(Q \vee \neg R)) \rightarrow (Q \wedge R)$. Applying the tableau rules generates the following derivation:

$$
\begin{array}{l}
\{(\neg[(P \wedge \neg(Q \vee \neg R)) \rightarrow (Q \wedge R)])\} \\
\quad \alpha\text{-Expansion} \Rightarrow_{\mathrm{T}}^{*} \{(\neg[(P \wedge \neg(Q \vee \neg R)) \rightarrow (Q \wedge R)], \\
\qquad P \wedge \neg(Q \vee \neg R), \neg(Q \wedge R), P, \neg(Q \vee \neg R), \neg Q, \neg\neg R, R)\} \\
\quad \beta\text{-Expansion} \Rightarrow_{\mathrm{T}} \{(\neg[(P \wedge \neg(Q \vee \neg R)) \rightarrow (Q \wedge R)], \\
\qquad P \wedge \neg(Q \vee \neg R), \neg(Q \wedge R), P, \neg(Q \vee \neg R), \neg Q, \neg\neg R, R, \neg Q), \\
\qquad (\neg[(P \wedge \neg(Q \vee \neg R)) \rightarrow (Q \wedge R)], \\
\qquad P \wedge \neg(Q \vee \neg R), \neg(Q \wedge R), P, \neg(Q \vee \neg R), \neg Q, \neg\neg R, R, \neg R)\}
\end{array}
$$

The state after β-expansion is final, i.e., no more rule can be applied. The first sequence is not closed, whereas the second sequence is because it contains R and $\neg R$. A tree representation, where common formulas of sequences are shared, can be found in Figure 2.5.

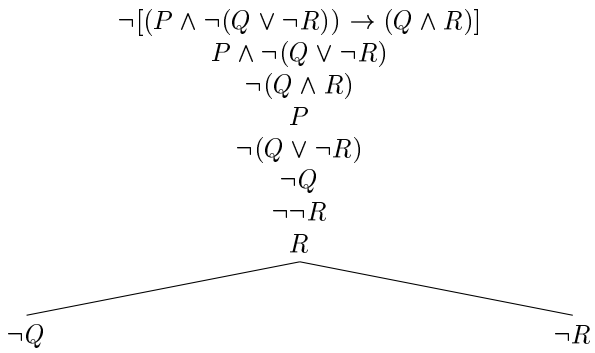

Figure 2.5: A Tableau for $(P \wedge \neg(Q \vee \neg R)) \rightarrow (Q \wedge R)$

Theorem 2.5.4 (Semantic Tableaux is Sound)**.** If for a formula ϕ the tableaux calculus computes $\{(\neg\phi)\} \Rightarrow_{\mathrm{T}}^{*} N$ and N is a closed, then ϕ is valid.

Proof. It is sufficient to show the following: (i) if N is closed then the disjunction of the conjunction of all sequence formulas is unsatisfiable (ii) all three sematic tableaux rules preserve satisfiability.

Part (i) is obvious: if N is closed all its sequences are closed. A sequence is closed if it contains a formula and its negation. The conjunction of two such formulas is unsatisfiable.

Part (ii) is shown by induction on the length of a derivation and then by a case analysis for the two rules. α-Expansion: for any valuation $\mathcal{A}$ if $\mathcal{A}(\psi) = 1$ then $\mathcal{A}(\psi_1) = \mathcal{A}(\psi_2) = 1$. β-Expansion: for any valuation $\mathcal{A}$ if $\mathcal{A}(\psi) = 1$ then $\mathcal{A}(\psi_1) = 1$ or $\mathcal{A}(\psi_2) = 1$ (see Proposition 2.5.3). □

Theorem 2.5.5 (Semantic Tableaux Terminates)**.** Starting from a start state $\{(\phi)\}$ for some formula ϕ, $\Rightarrow_{\mathrm{T}}^{+}$ is well-founded.

Proof. Take the two-folded multi-set extension of the lexicographic extension of $>$ on the naturals on triples (n, k, l). The measure μ is first defined on formulas by $\mu(\phi) := (n, k, l)$ where n is the number of equivalence symbols in ϕ, k is the sum of all disjunction, conjunction, implication symbols in ϕ and l is $|\phi|$. On sequences $(\phi_1, \ldots, \phi_n)$ the measure is defined to deliver a multiset by $\mu((\phi_1, \ldots, \phi_n)) := \{t_1, \ldots, t_n\}$ where $t_i = \mu(\phi_i)$ if ϕ is open in the sequence and $t_i = (0, 0, 0)$ otherwise. Finally, μ is extended to states by computing the multiset $\mu(N) := \{\mu(s) \mid s \in N\}$.

Note, that α-, as well as β-expansion strictly extend sequences. Once a formula is closed in a sequence by applying an expansion rule, it remains closed forever in the sequence.

An α-expansion on a formula $\psi_1 \wedge \psi_2$ on the sequence $(\phi_1, \ldots, \psi_1 \wedge \psi_2, \ldots, \phi_n)$ results in $(\phi_1, \ldots, \psi_1 \wedge \psi_2, \ldots, \phi_n, \psi_1, \psi_2)$. It needs to be shown $\mu((\phi_1, \ldots, \psi_1 \wedge \psi_2, \ldots, \phi_n)) >_{\text{mul}} \mu((\phi_1, \ldots, \psi_1 \wedge \psi_2, \ldots, \phi_n, \psi_1, \psi_2))$. In the second sequence $\mu(\psi_1 \wedge \psi_2) = (0, 0, 0)$ because the formula is closed. For the triple (n, k, l) assigned by μ to $\psi_1 \wedge \psi_2$ in the first sequence, it holds $(n, k, l) >_{\text{lex}} \mu(\psi_1)$, $(n, k, l) >_{\text{lex}} \mu(\psi_2)$ and $(n, k, l) >_{\text{lex}} (0, 0, 0)$, the former because the ψ_i are subformulas and the latter because $l \neq 0$. This proves the case.

A β-expansion on a formula $\psi_1 \vee \psi_2$ on the sequence $(\phi_1, \ldots, \psi_1 \vee \psi_2, \ldots, \phi_n)$ results in $(\phi_1, \ldots, \psi_1 \vee \psi_2, \ldots, \phi_n, \psi_1)$, $(\phi_1, \ldots, \psi_1 \vee \psi_2, \ldots, \phi_n, \psi_2)$. It needs to be shown $\mu((\phi_1, \ldots, \psi_1 \vee \psi_2, \ldots, \phi_n)) >_{\text{mul}} \mu((\phi_1, \ldots, \psi_1 \wedge \psi_2, \ldots, \phi_n, \psi_1))$ and $\mu((\phi_1, \ldots, \psi_1 \vee \psi_2, \ldots, \phi_n)) >_{\text{mul}} \mu((\phi_1, \ldots, \psi_1 \wedge \psi_2, \ldots, \phi_n, \psi_2))$. In the derived sequences $\mu(\psi_1 \vee \psi_2) = (0, 0, 0)$ because the formula is closed. For the triple (n, k, l) assigned by μ to $\psi_1 \vee \psi_2$ in the starting sequence, it holds $(n, k, l) >_{\text{lex}} \mu(\psi_1)$, $(n, k, l) >_{\text{lex}} \mu(\psi_2)$ and $(n, k, l) >_{\text{lex}} (0, 0, 0)$, the former because the ψ_i are subformulas and the latter because $l \neq 0$. This proves the case. □

Theorem 2.5.6 (Semantic Tableaux is Complete)**.** If ϕ is valid, semantic tableaux computes a closed state out of $\{(\neg\phi)\}$.

Proof. If ϕ is valid then $\neg\phi$ is unsatisfiable. Now assume after termination the resulting state and hence at least one sequence is not closed. For this sequence consider a valuation $\mathcal{A}$ consisting of the literals in the sequence. By assumption there are no opposite literals, so $\mathcal{A}$ is well-defined. I prove by contradiction that $\mathcal{A}$ is a model for the sequence. Assume not. Then there is a minimal formula in the sequence, with respect to the ordering on triples considered in the proof of Theorem 2.5.5, that is not satisfied by $\mathcal{A}$. By definition of $\mathcal{A}$ the formula cannot be a literal. So it is an α-formula or a β-formula. In all cases at least one descendant formula is contained in the sequence, is smaller than the original formula, false in $\mathcal{A}$ (Proposition 2.5.3) and hence contradicts the assumption. Therefore, $\mathcal{A}$ satisfies the sequence contradicting that $\neg\phi$ is unsatisfiable. □

Corollary 2.5.7 (Semantic Tableaux generates Models)**.** Let ϕ be a formula, $\{(\phi)\} \Rightarrow_{\text{T}}^{*} N$ and $s \in N$ be a sequence that is not closed and neither α-expansion nor β-expansion are applicable to s. Then the literals in s form a valuation that is a model for ϕ.

Proof. A consequence of the proof of Theorem 2.5.6 □

Consider the example tableau shown in Figure 2.5. The open first branch corresponds to the valuation $\mathcal{A} = \{P, R, \neg Q\}$ which is a model of the formula $\neg[(P \wedge \neg(Q \vee \neg R)) \rightarrow (Q \wedge R)]$.

2.6 Normal Forms

In order to check the status of a formula ϕ via truth tables, the truth table contains a column for the subformulas of ϕ and all valuations for its variables. Any shape of ϕ is fine in order to generate the respective truth table. For the superposition calculus (Section 2.8) and the CDCL (Conflict Driven Clause Learning) calculus (Section 2.10) I introduce in the next two sections, the shape of ϕ is restricted. Both calculi accept only conjunctions of disjunctions of literals, a particular *normal form*. It is called *Clause Normal Form* or simply *CNF*. The purpose of this section is to show that an arbitrary formula ϕ can be effectively transformed into an equivalent formula in CNF.

Definition 2.6.1 (CNF, DNF)**.** A formula is in *conjunctive normal form (CNF)* or *clause normal form* if it is a conjunction of disjunctions of literals, or in other words, a conjunction of clauses.

A formula is in *disjunctive normal form (DNF)*, if it is a disjunction of conjunctions of literals.

Although CNF and DNF are defined in almost any text book on automated reasoning, the definitions in the literature differ with respect to the "border" cases: (i) are complementary literals permitted in a clause? (ii) are duplicated literals permitted in a clause? (iii) are empty disjunctions/conjunctions permitted? For the above Definition 2.6.1 the answer is "yes" to all three questions. A clause containing complementary literals is valid, as in $P \vee Q \vee \neg P$. Duplicate literals may occur, as in $P \vee Q \vee P$. The empty disjunction is $\bot$ and the empty conjunction $\top$, i.e., the empty disjunction is always false while the empty conjunction is always true.

Checking the validity of CNF formulas or the unsatisfiability of DNF formulas is easy: (i) a formula in CNF is valid, if and only if each of its disjunctions contains a pair of complementary literals P and $\neg P$, (ii) conversely, a formula in DNF is unsatisfiable, if and only if each of its conjunctions contains a pair of complementary literals P and $\neg P$ (see Exercise 2.12).

On the other hand, checking the unsatisfiability of CNF formulas or the validity of DNF formulas is coNP-complete. For any propositional formula ϕ there is an equivalent formula in CNF and DNF and I will prove this below by actually providing an effective procedure for the transformation. However, also because of the above comment on validity and satisfiability checking for CNF and DNF formulas, respectively, the transformation is costly.