Proposition 2.4 *$F \models\!\mid G$ if and only if $\models (F \leftrightarrow G)$.*

Proof. Analogously to Prop. 2.3. □

Entailment is extended to sets of formulas N in the "natural way":

$N \models F$ if for all Π-valuations $\mathcal{A}$:
if $\mathcal{A} \models G$ for all $G \in N$, then $\mathcal{A} \models F$.

Note: Formulas are always finite objects; but sets of formulas may be infinite. Therefore, it is in general not possible to replace a set of formulas by the conjunction of its elements.

Validity vs. Unsatisfiability

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

Proposition 2.5 *F is valid if and only if $\neg F$ is unsatisfiable.*

Proof. (⇒) If F is valid, then $\mathcal{A}(F) = 1$ for every valuation $\mathcal{A}$. Hence $\mathcal{A}(\neg F) = 1 - \mathcal{A}(F) = 0$ for every valuation $\mathcal{A}$, so $\neg F$ is unsatisfiable.

(⇐) Analogously. □

Hence in order to design a theorem prover (validity checker) it is sufficient to design a checker for unsatisfiability.

In a similar way, entailment can be reduced to unsatisfiability and vice versa:

Proposition 2.6 *$N \models F$ if and only if $N \cup \{\neg F\}$ is unsatisfiable.*

Proposition 2.7 *$N \models \bot$ if and only if N is unsatisfiable.*

Checking Unsatisfiability

Every formula F contains only finitely many propositional variables. Obviously, $\mathcal{A}(F)$ depends only on the values of those finitely many variables in F under $\mathcal{A}$.

If F contains n distinct propositional variables, then it is sufficient to check 2^n valuations to see whether F is satisfiable or not ⇒ truth table.

So the satisfiability problem is clearly decidable (but, by Cook's Theorem, NP-complete).

Nevertheless, in practice, there are (much) better methods than truth tables to check the satisfiability of a formula. (later more)

Substitution Theorem

Proposition 2.8 *Let $\mathcal{A}$ be a valuation, let F and G be formulas, and let $H = H[F]_p$ be a formula in which F occurs as a subformula at position p.*

If $\mathcal{A}(F) = \mathcal{A}(G)$, then $\mathcal{A}(H[F]_p) = \mathcal{A}(H[G]_p)$.

Proof. The proof proceeds by induction over the length of p.

If $p = \varepsilon$, then $H[F]_\varepsilon = F$ and $H[G]_\varepsilon = G$, so $\mathcal{A}(H[F]_p) = \mathcal{A}(F) = \mathcal{A}(G) = H[G]_p$ by assumption.

If $p = 1q$ or $p = 2q$, then $H = \neg H_1$ or $H = H_1 \circ H_2$ for $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$. Assume that $p = 1q$ and that $H = H_1 \wedge H_2$, hence $H[F]_p = H[F]_{1q} = H_1[F]_q \wedge H_2$. By the induction hypothesis, $\mathcal{A}(H_1[F]_q) = \mathcal{A}(H_1[G]_q)$. Hence $\mathcal{A}(H[F]_{1q}) = \mathcal{A}(H_1[F]_q \wedge H_2) = \min(\mathcal{A}(H_1[F]_q), \mathcal{A}(H_2)) = \min(\mathcal{A}(H_1[G]_q), \mathcal{A}(H_2)) = \mathcal{A}(H_1[G]_q \wedge H_2) = \mathcal{A}(H[G]_{1q})$.

The case $p = 2q$ and the other boolean connectives are handled analogously. □

Theorem 2.9 *Let F and G be equivalent formulas, let $H = H[F]_p$ be a formula in which F occurs as a subformula at position p.*

Then $H[F]_p$ is equivalent to $H[G]_p$.

Proof. We have to show that $\mathcal{A}(H[F]_p) = \mathcal{A}(H[G]_p)$ for every Π-valuation $\mathcal{A}$.

Choose $\mathcal{A}$ arbitrarily. Since F and G are equivalent, we know that $\mathcal{A}(F) = \mathcal{A}(G)$. Hence, by the previous proposition, $\mathcal{A}(H[F]_p) = \mathcal{A}(H[G]_p)$. □

Some Important Equivalences

Proposition 2.10 *The following equivalences hold for all formulas F, G, H:*

$$
\begin{array}{rcll}
(F \wedge F) & \models\!\!\!\!\dashv & F & \\
(F \vee F) & \models\!\!\!\!\dashv & F & \textit{(Idempotency)} \\[4pt]
(F \wedge G) & \models\!\!\!\!\dashv & (G \wedge F) & \\
(F \vee G) & \models\!\!\!\!\dashv & (G \vee F) & \textit{(Commutativity)} \\[4pt]
(F \wedge (G \wedge H)) & \models\!\!\!\!\dashv & ((F \wedge G) \wedge H) & \\
(F \vee (G \vee H)) & \models\!\!\!\!\dashv & ((F \vee G) \vee H) & \textit{(Associativity)} \\[4pt]
(F \wedge (G \vee H)) & \models\!\!\!\!\dashv & ((F \wedge G) \vee (F \wedge H)) & \\
(F \vee (G \wedge H)) & \models\!\!\!\!\dashv & ((F \vee G) \wedge (F \vee H)) & \textit{(Distributivity)} \\[4pt]
(F \wedge (F \vee G)) & \models\!\!\!\!\dashv & F & \\
(F \vee (F \wedge G)) & \models\!\!\!\!\dashv & F & \textit{(Absorption)} \\[4pt]
(\neg\neg F) & \models\!\!\!\!\dashv & F & \textit{(Double Negation)} \\[4pt]
\neg(F \wedge G) & \models\!\!\!\!\dashv & (\neg F \vee \neg G) & \\
\neg(F \vee G) & \models\!\!\!\!\dashv & (\neg F \wedge \neg G) & \textit{(De Morgan's Laws)} \\[4pt]
(F \wedge G) & \models\!\!\!\!\dashv & F, \textit{ if } G \textit{ is a tautology} & \\
(F \vee G) & \models\!\!\!\!\dashv & \top, \textit{ if } G \textit{ is a tautology} & \\
(F \wedge G) & \models\!\!\!\!\dashv & \bot, \textit{ if } G \textit{ is unsatisfiable} & \\
(F \vee G) & \models\!\!\!\!\dashv & F, \textit{ if } G \textit{ is unsatisfiable} & \textit{(Tautology Laws)} \\[4pt]
(F \leftrightarrow G) & \models\!\!\!\!\dashv & ((F \rightarrow G) \wedge (G \rightarrow F)) & \\
(F \leftrightarrow G) & \models\!\!\!\!\dashv & ((F \wedge G) \vee (\neg F \wedge \neg G)) & \textit{(Equivalence)} \\[4pt]
(F \rightarrow G) & \models\!\!\!\!\dashv & (\neg F \vee G) & \textit{(Implication)}
\end{array}
$$

An Important Entailment

Proposition 2.11 *The following entailment holds for all formulas F, G, H:*

$$(F \vee H) \wedge (G \vee \neg H) \models F \vee G \qquad \textit{(Generalized Resolution)}$$

2.4 Normal Forms

We define *conjunctions* of formulas as follows:

$\bigwedge_{i=1}^{0} F_i = \top$.

$\bigwedge_{i=1}^{1} F_i = F_1$.

$\bigwedge_{i=1}^{n+1} F_i = \bigwedge_{i=1}^{n} F_i \wedge F_{n+1}$.

and analogously *disjunctions:*

$\bigvee_{i=1}^{0} F_i = \bot$.

$\bigvee_{i=1}^{1} F_i = F_1$.

$\bigvee_{i=1}^{n+1} F_i = \bigvee_{i=1}^{n} F_i \vee F_{n+1}$.

Literals and Clauses

A *literal* is either a propositional variable P or a negated propositional variable $\neg P$.

A *clause* is a (possibly empty) disjunction of literals.

CNF and DNF

A formula is in *conjunctive normal form (CNF, 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.

Warning: definitions in the literature differ:

are complementary literals permitted?
are duplicated literals permitted?
are empty disjunctions/conjunctions permitted?

Checking the validity of CNF formulas or the unsatisfiability of DNF formulas is easy:

A formula in CNF is valid, if and only if each of its disjunctions contains a pair of complementary literals P and $\neg P$.

Conversely, a formula in DNF is unsatisfiable, if and only if each of its conjunctions contains a pair of complementary literals P and $\neg P$.

On the other hand, checking the unsatisfiability of CNF formulas or the validity of DNF formulas is known to be coNP-complete.

Conversion to CNF/DNF

Proposition 2.12 *For every formula there is an equivalent formula in CNF (and also an equivalent formula in DNF).*

Proof. We describe a (naive) algorithm to convert a formula to CNF.

Apply the following rules as long as possible (modulo commutativity of $\wedge$ and $\vee$):

Step 1: Eliminate equivalences:

$$H[F \leftrightarrow G]_p \;\Rightarrow_{\mathrm{CNF}}\; H[(F \rightarrow G) \wedge (G \rightarrow F)]_p$$

Step 2: Eliminate implications:

$$H[F \rightarrow G]_p \;\Rightarrow_{\mathrm{CNF}}\; H[\neg F \vee G]_p$$

Step 3: Push negations downward:

$$\begin{aligned} H[\neg(F \vee G)]_p &\;\Rightarrow_{\mathrm{CNF}}\; H[\neg F \wedge \neg G]_p \\ H[\neg(F \wedge G)]_p &\;\Rightarrow_{\mathrm{CNF}}\; H[\neg F \vee \neg G]_p \end{aligned}$$

Step 4: Eliminate multiple negations:

$$H[\neg\neg F]_p \;\Rightarrow_{\mathrm{CNF}}\; H[F]_p$$

Step 5: Push disjunctions downward:

$$H[(F \wedge F') \vee G]_p \;\Rightarrow_{\mathrm{CNF}}\; H[(F \vee G) \wedge (F' \vee G)]_p$$

Step 6: Eliminate $\top$ and $\bot$:

$$\begin{aligned} H[F \wedge \top]_p &\;\Rightarrow_{\mathrm{CNF}}\; H[F]_p \\ H[F \wedge \bot]_p &\;\Rightarrow_{\mathrm{CNF}}\; H[\bot]_p \\ H[F \vee \top]_p &\;\Rightarrow_{\mathrm{CNF}}\; H[\top]_p \\ H[F \vee \bot]_p &\;\Rightarrow_{\mathrm{CNF}}\; H[F]_p \\ H[\neg\bot]_p &\;\Rightarrow_{\mathrm{CNF}}\; H[\top]_p \\ H[\neg\top]_p &\;\Rightarrow_{\mathrm{CNF}}\; H[\bot]_p \end{aligned}$$

Proving termination is easy for steps 2, 4, and 6; steps 1, 3, and 5 are a bit more complicated.

For step 1, we can prove termination in the following way: We define a function μ_1 from formulas to positive integers such that $\mu_1(\bot) = \mu_1(\top) = \mu_1(P) = 1$, $\mu_1(\neg F) = \mu_1(F)$, $\mu_1(F \wedge G) = \mu_1(F \vee G) = \mu_1(F \rightarrow G) = \mu_1(F) + \mu_1(G)$, and $\mu_1(F \leftrightarrow G) = 2\mu_1(F) + 2\mu_1(G) + 1$. Observe that μ_1 is constructed in such a way that $\mu_1(F) > \mu_1(G)$ implies $\mu_1(H[F]) > \mu_1(H[G])$ for all formulas F, G, and H. Furthermore, μ_1 has the property that swapping the arguments of some $\wedge$ or $\vee$ in a formula F does not change the value of $\mu_1(F)$. (This is important since the transformation rules can be applied modulo commutativity of $\wedge$ and $\vee$.). Using these properties, we can show that whenever a formula H' is the result of applying the rule of step 1 to a formula H, then $\mu_1(H) > \mu_1(H')$. Since μ_1 takes only positive integer values, step 1 must terminate.

Termination of steps 3 and 5 is proved similarly. For step 3, we use function μ_2 from formulas to positive integers such that $\mu_2(\bot) = \mu_2(\top) = \mu_2(P) = 1$, $\mu_2(\neg F) = 2\mu_2(F)$, $\mu_2(F \wedge G) = \mu_2(F \vee G) = \mu_2(F \rightarrow G) = \mu_2(F \leftrightarrow G) = \mu_2(F) + \mu_2(G) + 1$. Whenever a formula H' is the result of applying a rule of step 3 to a formula H, then $\mu_2(H) > \mu_2(H')$. Since μ_2 takes only positive integer values, step 3 must terminate.

For step 5, we use a function μ_3 from formulas to positive integers such that $\mu_3(\bot) = \mu_3(\top) = \mu_3(P) = 1$, $\mu_3(\neg F) = \mu_3(F) + 1$, $\mu_3(F \wedge G) = \mu_3(F \rightarrow G) = \mu_3(F \leftrightarrow G) = \mu_3(F) + \mu_3(G) + 1$, and $\mu_3(F \vee G) = 2\mu_3(F)\mu_3(G)$. Again, if a formula H' is the result of applying a rule of step 5 to a formula H, then $\mu_3(H) > \mu_3(H')$. Since μ_3 takes only positive integer values, step 5 terminates, too.

The resulting formula is equivalent to the original one and in CNF.

The conversion of a formula to DNF works in the same way, except that conjunctions have to be pushed downward in step 5. □

Negation Normal Form (NNF)

The formula after application of Step 4 is said to be in *Negation Normal Form*, i.e., it contains neither $\rightarrow$ nor $\leftrightarrow$ and negation symbols only occur in front of propositional variables (atoms).

Complexity

Conversion to CNF (or DNF) may produce a formula whose size is *exponential* in the size of the original one.

2.5 Improving the CNF Transformation

The goal

"find a formula G in CNF such that $F \models\mid G$"

is unpractical.

But if we relax the requirement to

"find a formula G in CNF such that $F \models \bot \Leftrightarrow G \models \bot$"

we can get an efficient transformation.

Tseitin Transformation

Proposition 2.13 *A formula $H[F]_p$ is satisfiable if and only if $H[Q]_p \wedge (Q \leftrightarrow F)$ is satisfiable, where Q is a new propositional variable that works as an abbreviation for F.*

Satisfiability-preserving CNF transformation (Tseitin 1970):

Use the rule above recursively for all subformulas in the original formula (this introduces a linear number of new propositional variables Q and definitions $Q \leftrightarrow F$).

Convert of the resulting conjunction to CNF (this increases the size only by an additional factor, since each formula $Q \leftrightarrow F$ yields at most four clauses in the CNF).

Polarity-based CNF Transformation

A further improvement is possible by taking the *polarity* of the subformula F into account.

Proposition 2.14 *Let $\mathcal{A}$ be a valuation, let F and G be formulas, and let $H = H[F]_p$ be a formula in which F occurs as a subformula at position p.*

If $\mathrm{pol}(H, p) = 1$ *and* $\mathcal{A}(F) \leq \mathcal{A}(G)$, *then* $\mathcal{A}(H[F]_p) \leq \mathcal{A}(H[G]_p)$.

If $\mathrm{pol}(H, p) = -1$ *and* $\mathcal{A}(F) \geq \mathcal{A}(G)$, *then* $\mathcal{A}(H[F]_p) \leq \mathcal{A}(H[G]_p)$.

Proof. Exercise. □

Let Q be a propositional variable not occurring in $H[F]_p$.

Define the formula $\mathrm{def}(H, p, Q, F)$ by

- $(Q \rightarrow F)$, if $\mathrm{pol}(H, p) = 1$,
- $(F \rightarrow Q)$, if $\mathrm{pol}(H, p) = -1$,
- $(Q \leftrightarrow F)$, if $\mathrm{pol}(H, p) = 0$.

Proposition 2.15 *Let Q be a propositional variable not occurring in $H[F]_p$. Then $H[F]_p$ is satisfiable if and only if $H[Q]_p \wedge \mathrm{def}(H, p, Q, F)$ is satisfiable.*

Proof. ($\Rightarrow$) Since $H[F]_p$ is satisfiable, there exists a Π-valuation $\mathcal{A}$ such that $\mathcal{A} \models H[F]_p$. Let $\Pi' = \Pi \cup \{Q\}$ and define the Π'-valuation $\mathcal{A}'$ by $\mathcal{A}'(P) = \mathcal{A}(P)$ for $P \in \Pi$ and $\mathcal{A}'(Q) = \mathcal{A}(F)$. Obviously $\mathcal{A}'(\mathrm{def}(H, p, Q, F)) = 1$; moreover $\mathcal{A}'(H[Q]_p) = \mathcal{A}'(H[F]_p) = \mathcal{A}(H[F]_p) = 1$ by Prop. 2.8, so $H[Q]_p \wedge \mathrm{def}(H, p, Q, F)$ is satisfiable.

($\Leftarrow$) Let $\mathcal{A}$ be a valuation such that $\mathcal{A} \models H[Q]_p \wedge \mathrm{def}(H, p, Q, F)$. So $\mathcal{A}(H[Q]_p) = 1$ and $\mathcal{A}(\mathrm{def}(H, p, Q, F)) = 1$. We will show that $\mathcal{A} \models H[F]_p$.

If $\mathrm{pol}(H, p) = 0$, then $\mathrm{def}(H, p, Q, F) = (Q \leftrightarrow F)$, so $\mathcal{A}(Q) = \mathcal{A}(F)$, hence $\mathcal{A}(H[F]_p) = \mathcal{A}(H[Q]_p) = 1$ by Prop. 2.8.

If $\mathrm{pol}(H, p) = 1$, then $\mathrm{def}(H, p, Q, F) = (Q \rightarrow F)$, so $\mathcal{A}(Q) \leq \mathcal{A}(F)$. By Prop. 2.14, $\mathcal{A}(H[F]_p) \geq \mathcal{A}(H[Q]_p) = 1$, so $\mathcal{A}(H[F]_p) = 1$.

If $\mathrm{pol}(H, p) = -1$, then $\mathrm{def}(H, p, Q, F) = (F \rightarrow Q)$, so $\mathcal{A}(F) \leq \mathcal{A}(Q)$. By Prop. 2.14, $\mathcal{A}(H[F]_p) \geq \mathcal{A}(H[Q]_p) = 1$, so $\mathcal{A}(H[F]_p) = 1$. □

Optimized CNF

Not every introduction of a definition for a subformula leads to a smaller CNF.

The number of eventually generated clauses is a good indicator for useful CNF transformations.