2.2 Semantics

In *classical logic* (dating back to Aristoteles) 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:

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

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

2.3 Models, Validity, and Satisfiability

F is *valid* in $\mathcal{A}$ ($\mathcal{A}$ is a *model* of F; F *holds* under $\mathcal{A}$):

$$\mathcal{A} \models F \;:\Leftrightarrow\; \mathcal{A}(F) = 1$$

F is *valid* (or is a *tautology*):

$$\models F \;:\Leftrightarrow\; \mathcal{A} \models F \text{ for all } \Pi\text{-valuations } \mathcal{A}$$

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 $\mathcal{A} \models F \;\Rightarrow\; \mathcal{A} \models G$.

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

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

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

($\Leftarrow$) 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, $\mathcal{A}(F \rightarrow G) = \max(1 - \mathcal{A}^*(F), \mathcal{A}^*(G)) = \max(1 - 1, 0) = 0$, so $(F \rightarrow G)$ does not hold in $\mathcal{A}$. □

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. ($\Rightarrow$) 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.

($\Leftarrow$) Analogously. $\square$

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

In a similar way, entailment $N \models F$ can be reduced to unsatisfiability:

Proposition 2.6 *$N \models F$ if and only if $N \cup \{\neg F\}$ 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 $\Rightarrow$ 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 2.7 *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. The proof proceeds by induction over the formula structure of H.

Each of the formulas $\bot$, $\top$, and P for $P \in \Pi$ contains only one subformula, namely itself. Hence, if $H = H[F]_\varepsilon$ equals $\bot$, $\top$, or P, then $H[F]_\varepsilon = F$, $H[G]_\varepsilon = G$, and we are done by assumption.

If $H = H_1 \wedge H_2$, then either $p = \varepsilon$ (this case is treated as above), or F is a subformula of H_1 or H_2 at position $1p'$ or $2p'$, respectively. Without loss of generality, assume that F is a subformula of H_1, so $H = H_1[F]_{p'} \wedge H_2$. By the induction hypothesis, $H_1[F]_{p'}$ and $H_1[G]_{p'}$ are equivalent. Hence, for any valuation $\mathcal{A}$, $\mathcal{A}(H[F]_{1p'}) = \mathcal{A}(H_1[F]_{p'} \wedge H_2) = \min(\mathcal{A}(H_1[F]_{p'}), \mathcal{A}(H_2)) = \min(\mathcal{A}(H_1[G]_{p'}), \mathcal{A}(H_2)) = \mathcal{A}(H_1[G]_{p'} \wedge G_2) = \mathcal{A}(H[G]_{1p'})$.

The other boolean connectives are handled analogously. $\square$

Some Important Equivalences

Proposition 2.8 *The following equivalences are valid for all formulas F, G, H:*

$$
\begin{aligned}
(F \wedge F) &\leftrightarrow F \\
(F \vee F) &\leftrightarrow F && \textit{(Idempotency)} \\
(F \wedge G) &\leftrightarrow (G \wedge F) \\
(F \vee G) &\leftrightarrow (G \vee F) && \textit{(Commutativity)} \\
(F \wedge (G \wedge H)) &\leftrightarrow ((F \wedge G) \wedge H) \\
(F \vee (G \vee H)) &\leftrightarrow ((F \vee G) \vee H) && \textit{(Associativity)} \\
(F \wedge (G \vee H)) &\leftrightarrow ((F \wedge G) \vee (F \wedge H)) \\
(F \vee (G \wedge H)) &\leftrightarrow ((F \vee G) \wedge (F \vee H)) && \textit{(Distributivity)}
\end{aligned}
$$

$$
\begin{aligned}
(F \wedge (F \vee G)) &\leftrightarrow F \\
(F \vee (F \wedge G)) &\leftrightarrow F && \textit{(Absorption)} \\
(\neg\neg F) &\leftrightarrow F && \textit{(Double Negation)} \\
\neg(F \wedge G) &\leftrightarrow (\neg F \vee \neg G) \\
\neg(F \vee G) &\leftrightarrow (\neg F \wedge \neg G) && \textit{(De Morgan's Laws)} \\
(F \wedge G) &\leftrightarrow F, \textit{ if } G \textit{ is a tautology} \\
(F \vee G) &\leftrightarrow \top, \textit{ if } G \textit{ is a tautology} \\
(F \wedge G) &\leftrightarrow \bot, \textit{ if } G \textit{ is unsatisfiable} \\
(F \vee G) &\leftrightarrow F, \textit{ if } G \textit{ is unsatisfiable} && \textit{(Tautology Laws)}
\end{aligned}
$$

$$
\begin{aligned}
(F \leftrightarrow G) &\leftrightarrow ((F \rightarrow G) \wedge (G \rightarrow F)) && \textit{(Equivalence)} \\
(F \rightarrow G) &\leftrightarrow (\neg F \vee G) && \textit{(Implication)}
\end{aligned}
$$

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 2.9 *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 associativity and 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. Using this property, 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.

Satisfiability-preserving Transformations

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.

Idea: A formula $H[F]_p$ is satisfiable if and only if $H[P] \wedge (P \leftrightarrow F)$ is satisfiable (where P is a new propositional variable that works as an abbreviation for F).

We can use this rule recursively for all subformulas in the original formula (this introduces a linear number of new propositional variables).

Conversion of the resulting formula to CNF increases the size only by an additional factor (each formula $P \leftrightarrow F$ gives rise to at most one application of the distributivity law).

Optimized Transformations

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

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

Define the formula $\operatorname{def}(H, p, P, F)$ by

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

Proposition 2.10 *Let P be a propositional variable not occurring in $H[F]_p$. Then $H[F]_p$ is satisfiable if and only if $H[P]_p \wedge \operatorname{def}(H, p, P, F)$ is satisfiable.*

Proof. Exercise. □

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

The functions ν and $\bar{\nu}$ give us an overapproximation for the number of clauses generated by a formula that occurs positively/negatively.

G	$\nu(G)$	$\bar{\nu}(G)$
$F_1 \land F_2$	$\nu(F_1) + \nu(F_2)$	$\bar{\nu}(F_1)\bar{\nu}(F_2)$
$F_1 \lor F_2$	$\nu(F_1)\nu(F_2)$	$\bar{\nu}(F_1) + \bar{\nu}(F_2)$
$F_1 \rightarrow F_2$	$\bar{\nu}(F_1)\nu(F_2)$	$\nu(F_1) + \bar{\nu}(F_2)$
$F_1 \leftrightarrow F_2$	$\nu(F_1)\bar{\nu}(F_2) + \bar{\nu}(F_1)\nu(F_2)$	$\nu(F_1)\nu(F_2) + \bar{\nu}(F_1)\bar{\nu}(F_2)$
$\neg F_1$	$\bar{\nu}(F_1)$	$\nu(F_1)$
$P, \top, \bot$	1	1

Optimized CNF

A better CNF transformation:

Step 1: Exhaustively apply modulo commutativity of $\leftrightarrow$ and associativity/commutativity of $\land$, $\lor$:

$$
\begin{aligned}
H[(F \land \top)]_p &\Rightarrow_{\mathrm{OCNF}} H[F]_p \\
H[(F \lor \bot)]_p &\Rightarrow_{\mathrm{OCNF}} H[F]_p \\
H[(F \leftrightarrow \bot)]_p &\Rightarrow_{\mathrm{OCNF}} H[\neg F]_p \\
H[(F \leftrightarrow \top)]_p &\Rightarrow_{\mathrm{OCNF}} H[F]_p \\
H[(F \lor \top)]_p &\Rightarrow_{\mathrm{OCNF}} H[\top]_p \\
H[(F \land \bot)]_p &\Rightarrow_{\mathrm{OCNF}} H[\bot]_p
\end{aligned}
$$

$$
\begin{aligned}
H[(F \land F)]_p &\Rightarrow_{\mathrm{OCNF}} H[F]_p \\
H[(F \lor F)]_p &\Rightarrow_{\mathrm{OCNF}} H[F]_p \\
H[(F \land (F \lor G))]_p &\Rightarrow_{\mathrm{OCNF}} H[F]_p \\
H[(F \lor (F \land G))]_p &\Rightarrow_{\mathrm{OCNF}} H[F]_p \\
H[(F \land \neg F)]_p &\Rightarrow_{\mathrm{OCNF}} H[\bot]_p \\
H[(F \lor \neg F)]_p &\Rightarrow_{\mathrm{OCNF}} H[\top]_p \\
H[\neg \top]_p &\Rightarrow_{\mathrm{OCNF}} H[\bot]_p \\
H[\neg \bot]_p &\Rightarrow_{\mathrm{OCNF}} H[\top]_p
\end{aligned}
$$

$$
\begin{aligned}
H[(F \rightarrow \bot)]_p &\Rightarrow_{\mathrm{OCNF}} H[\neg F]_p \\
H[(F \rightarrow \top)]_p &\Rightarrow_{\mathrm{OCNF}} H[\top]_p \\
H[(\bot \rightarrow F)]_p &\Rightarrow_{\mathrm{OCNF}} H[\top]_p \\
H[(\top \rightarrow F)]_p &\Rightarrow_{\mathrm{OCNF}} H[F]_p
\end{aligned}
$$

Step 2: Introduce top-down fresh variables for beneficial subformulas:

$$H[F]_p \Rightarrow_{\mathrm{OCNF}} H[P]_p \wedge \mathrm{def}(H, p, P, F)$$

where P is new to $H[F]_p$ and $\nu(H[F]_p) > \nu(H[P]_p \wedge \mathrm{def}(H, p, P, F))$.

Remark: Although computing ν is not practical in general, the test $\nu(H[F]_p) > \nu(H[P]_p \wedge \mathrm{def}(H, p, P, F))$ can be computed in constant time.

Step 3: Eliminate equivalences dependent on their polarity:

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

if $\mathrm{pol}(F, p) = 1$ or $\mathrm{pol}(F, p) = 0$.

$$H[F \leftrightarrow G]_p \Rightarrow_{\mathrm{OCNF}} H[(F \wedge G) \vee (\neg F \wedge \neg G)]_p$$

if $\mathrm{pol}(F, p) = -1$.

Step 4: Apply steps 2, 3, 4, 5 of $\Rightarrow_{\mathrm{CNF}}$

Remark: The $\Rightarrow_{\mathrm{OCNF}}$ algorithm is already close to a state of the art algorithm, but some additional redundancy tests and simplification mechanisms are missing.