Equivalence

Two formulas F and G are (semantically) equivalent if $\mathcal{A}(F) = \mathcal{A}(G)$ for every assignment $\mathcal{A}$ that is suitable for both F and G.

We write $F \equiv G$ to denote that F and G are equivalent.

Exercise

Which of the following equivalences hold?

$$
\begin{aligned}
(A \wedge (A \vee B)) &\equiv A \\
\neg(A \vee B) &\equiv (\neg A \wedge \neg B) \\
(A \wedge (B \vee C)) &\equiv ((A \wedge B) \vee C) \\
(A \wedge (B \vee C)) &\equiv ((A \wedge B) \vee (A \wedge C))
\end{aligned}
$$

Main logical questions

- Model checking
  Let F be a formula and let $\mathcal{A}$ be a suitable assignment.
  Does $\mathcal{A}(F) = 1$ hold?
- Satisfiability
  Let F be a formula. Is F satisfiable ?
- Validity
  Let F be a formula. Ist F valid ?
- Consequence
  Let F and G be formulas. Does $F \models G$ hold?
- Equivalence
  Let F und G be formulas. Does $F \equiv G$ hold?

Observation

The following connections hold:

$$(F \rightarrow G) \text{ is valid} \quad \text{if and only if} \quad F \models G$$

$$(F \leftrightarrow G) \text{ is valid} \quad \text{if and only if} \quad F \equiv G$$

Reductions between problems (I)

- Validity to Unsatisfiabilty (and back):

$$\begin{array}{rcl} F \text{ valid} & \text{iff} & \neg F \text{ unsatisfiable} \\ F \text{ unsatisfiable} & \text{iff} & \neg F \text{ valid} \end{array}$$

- Validity to Consequence:

$$F \text{ valid} \quad \text{iff} \quad \mathbf{true} \models F$$

- Consequence to Validity:

$$F \models G \quad \text{iff} \quad F \rightarrow G \text{ valid}$$

Reductions bewteen problems (II)

- Validity to Equivalence:

$$F \text{ valid} \quad \text{iff} \quad F \equiv \mathbf{true}$$

- Equivalence to Validity:

$$F \equiv G \quad \text{iff} \quad F \leftrightarrow G \text{ valid}$$

Properties of semantic equivalence

Semantic equivalence is an equivalence relation between formulas.

Semantic equivalence is closed under operators:

> If $F_1 \equiv F_2$ and $G_1 \equiv G_2$ hold, then
> $(F_1 \wedge G_1) \equiv (F_2 \wedge G_2)$, $(F_1 \vee G_1) \equiv (F_2 \vee G_2)$ and $\neg F_1 \equiv \neg F_2$
> hold too.

Equivalence relation $+$ Closure under Operations

$=$

Congruence relation

Substitution theorem

Closure under operations can also be formulated this way:

Theorem (substitution theorem)
Let F and G be equivalent formulas. Let H be a formula with (at least) an occurrence of F as subformula. Then H and H' are equivalent, where H' is the result of substituting an arbitrary occurrence of F in H by G.

Equivalence (I)

Theorem

The following equivalences hold for every formulas F and G:

$$
\begin{array}{rcll}
(F \wedge F) & \equiv & F & \\
(F \vee F) & \equiv & F & \text{(Idempotence)} \\
(F \wedge G) & \equiv & (G \wedge F) & \\
(F \vee G) & \equiv & (G \vee F) & \text{(Commutativity)} \\
((F \wedge G) \wedge H) & \equiv & (F \wedge (G \wedge H)) & \\
((F \vee G) \vee H) & \equiv & (F \vee (G \vee H)) & \text{(Associativity)} \\
(F \wedge (F \vee G)) & \equiv & F & \\
(F \vee (F \wedge G)) & \equiv & F & \text{(Absorption)}
\end{array}
$$

Equivalences (II)

$$\begin{array}{rcll}
(F \wedge (G \vee H)) & \equiv & ((F \wedge G) \vee (F \wedge H)) & \\
(F \vee (G \wedge H)) & \equiv & ((F \vee G) \wedge (F \vee H)) & \text{(Distributivity)} \\
\neg\neg F & \equiv & F & \text{(Double negation)} \\
\neg(F \wedge G) & \equiv & (\neg F \vee \neg G) & \\
\neg(F \vee G) & \equiv & (\neg F \wedge \neg G) & \text{(deMorgan's Laws)} \\
(F \vee G) & \equiv & F\text{, if } F \text{ is a tautology} & \\
(F \wedge G) & \equiv & G\text{, if } F \text{ is a tautologie} & \text{(Tautology Laws)} \\
(F \vee G) & \equiv & G\text{, if } F \text{ is unsatisfiable} & \\
(F \wedge G) & \equiv & F\text{, if } F \text{ is unsatisfiable} & \text{(Unsatisfiability Laws)}
\end{array}$$

Normal forms (I)

Definition (Normal forms)
A literal is an atomic formula or the negation of an atomic formula. (In the former case the literal is positive and negative in the latter).

A formula F is in conjunctive normal form (**CNF**) if it is a conjunction of disjunctions of literals:

$$F = (\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m_i} L_{i,j})),$$

where $L_{i,j} \in \{A_1, A_2, \cdots\} \cup \{\neg A_1, \neg A_2, \cdots\}$

Normal forms (II)

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

$$F = (\bigvee_{i=1}^{n} (\bigwedge_{j=1}^{m_i} L_{i,j})),$$

where $L_{i,j} \in \{A_1, A_2, \cdots\} \cup \{\neg A_1, \neg A_2, \cdots\}$

Normalization methods for CNF

1. Substitute every occurrence of a subformula of the form

$$\begin{array}{rcl} \neg\neg G & \text{by} & G \\ \neg(G \wedge H) & \text{by} & (\neg G \vee \neg H) \\ \neg(G \vee H) & \text{by} & (\neg G \wedge \neg H) \end{array}$$

   until no such formulas occur.

2. Substitute in every occurrence of a subformula of the form

$$\begin{array}{rcl} (F \vee (G \wedge H)) & \text{durch} & ((F \vee G) \wedge (F \vee H)) \\ ((F \wedge G) \vee H) & \text{durch} & ((F \vee H) \wedge (G \vee H)) \end{array}$$

   until no such formulas occur.

Derivation from the truth table

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

DNF: Each row of the truth table with value 1 yields a conjunction, a 0 in column A yields $\neg A$, and a 1 yields A

$$(\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge C)$$
$$\vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge B \wedge C)$$

CNF: Each row of the truth table with value 0 yields a disjunction, a 0 in column A yields A, and a 1 yields $\neg A$

$$(A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C)$$
$$\wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C)$$

Precedence

Operator precedence:

$$
\begin{array}{ll}
\leftrightarrow & \text{binds weaker than} \\
\rightarrow & \text{which binds weaker than} \\
\vee & \text{which binds weaker than} \\
\wedge & \text{which binds weaker than} \\
\neg & .
\end{array}
$$

So we have

$$A \leftrightarrow B \vee \neg C \rightarrow D \wedge \neg E \equiv (A \leftrightarrow ((B \vee \neg C) \rightarrow (D \wedge \neg E)))$$

But: well chosen parenthesis help to visually parse formulas.