Automated Theorem Proving

Lecture 2: Preliminaries Continued and Propositional Logic

Prof. Dr. Jasmin Blanchette

based on slides by Dr. Uwe Waldmann

Winter Term 2025/26

1.4 Multisets

Let M be a set. A multiset S over M is a mapping $S : M \to \mathbb{N}$. We interpret $S(m)$ as the number of occurrences of elements m of the base set M within the multiset S.

Example. $S = \{a, a, a, b, b\}$ is a multiset over $\{a, b, c\}$, where $S(a) = 3$, $S(b) = 2$, $S(c) = 0$.

We say that m is an element of S if $S(m) > 0$.

Multisets

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

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

Multisets

A multiset S is called finite if the set

$$\{m \in M \mid S(m) > 0\}$$

is finite.

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

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

Multiset Orderings

Theorem 1.4.1:

(a) If $\succ$ is transitive, then $\succ_{\mathsf{mul}}$ is transitive.

(b) If $\succ$ is irreflexive and transitive, then $\succ_{\mathsf{mul}}$ is irreflexive.

(c) If $\succ$ is a well-founded ordering, then $\succ_{\mathsf{mul}}$ is a well-founded ordering.

(d) If $\succ$ is a strict total ordering, then $\succ_{\mathsf{mul}}$ is a strict total ordering.

Multiset Orderings

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):

Multiset Orderings

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

$$S_1 \succ^{\mathrm{HO}}_{\mathrm{mul}} S_2 \text{ if and only if}$$
$$S_1 \neq S_2 \text{ and}$$
$$\forall m \in M\colon \big(S_2(m) > S_1(m)$$
$$\Rightarrow \; \exists m' \in M\colon m' \succ m \text{ and } S_1(m') > S_2(m')\big)$$

Multiset Orderings

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

$S_1 \succ^1_{\mathrm{mul}} S_2$ if and only if

there exists $x \in S_1$ and a multiset Y over M such that

$$S_2 = (S_1 - \{x\}) \cup Y,$$

$$\forall y \in Y\colon x \succ y$$

Multiset Orderings

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

Theorem 1.4.2:
If $\succ$ is a strict partial ordering, then
(a) $\succ_{\mathsf{mul}} = \succ_{\mathsf{mul}}^{\mathsf{HO}}$,
(b) $\succ_{\mathsf{mul}} = (\succ_{\mathsf{mul}}^{1})^{+}$.

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

Part 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).

2.1 Syntax

When we define a logic, we must define

what formulas of the logic look like (syntax),
and what they mean (semantics).

We start with the syntax.

Propositional formulas are built from

- propositional variables,
- logical connectives (e.g., $\wedge$, $\vee$).

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:

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

Propositional Formulas

Sometimes further connectives are used, for instance

$(F \leftarrow G)$	(reverse implication)
$(F \oplus G)$	(exclusive or)
$(\text{if } F \text{ then } G_1 \text{ else } G_0)$	(if-then-else)

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 $P \wedge Q \wedge R$ (analogously for $\vee$).

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

Formula Manipulation

Automated theorem proving is very much formula manipulation.
We perform syntactic operations on formulas to show semantic properties of formulas.

Formula Manipulation

To precisely describe 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

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

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

Formula Manipulation

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 $p \parallel q$.

Formula Manipulation

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

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

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

Formula Manipulation

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

$$
\begin{aligned}
F[G]_\varepsilon &:= G \\
(\neg F)[G]_{1p} &:= \neg(F[G]_p) \\
(F_1 \circ F_2)[G]_{1p} &:= (F_1[G]_p \circ F_2) \\
(F_2 \circ F_2)[G]_{2p} &:= (F_1 \circ F_2[G]_p)
\end{aligned}
$$

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

Formula Manipulation

Example 2.1.1:
The set of positions for the formula $F = (P \rightarrow Q) \rightarrow (P \wedge \neg R)$
is $\mathrm{pos}(F) = \{\varepsilon, 1, 11, 12, 2, 21, 22, 221\}$.

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

2.2 Semantics

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

There are multi-valued logics that have more than two truth values.

Valuations

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

A Π-valuation is a function

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

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

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

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

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

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

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

Note that formulas and truth values are disjoint classes of objects. Statements like $P = 1$ or $F \wedge G = 0$ that equate formulas and truth values are nonsensical.

A formula is never *equal to* a truth value, but it *has* a truth value in some valuation $\mathcal{A}$.

2.3 Models, Validity, and Satisfiability

Let F be a Π-formula.

We say that F is true in $\mathcal{A}$ ($\mathcal{A}$ is a model of F;
F is valid in $\mathcal{A}$; F holds in $\mathcal{A}$), 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 $F \models\!\!\mid G$,
if for all Π-valuations $\mathcal{A}$ we have

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

or equivalently

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

Entailment and Equivalence

F and G are called equisatisfiable
if either both F and G are satisfiable, or both F and G are unsatisfiable.

Entailment and Equivalence

The notions defined above for formulas, such as satisfiability, validity, or entailment, are extended to sets of formulas N by treating sets of formulas analogously to conjunctions of formulas, e.g.:

$\mathcal{A} \models N$ if $\mathcal{A} \models G$ for all $G \in N$.

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

Entailment and Equivalence

Proposition 2.3.1:

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

Proposition 2.3.2:

$F \mathrel{|}\!\!\!\models\!\!\mathrel{|} G$ if and only if $\models (F \leftrightarrow G)$.

Validity vs. Unsatisfiability

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

Proposition 2.3.3:
F is valid if and only if $\neg F$ is unsatisfiable.

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

Validity vs. Unsatisfiability

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

Proposition 2.3.4:
$G \models F$ if and only if $G \wedge \neg F$ is unsatisfiable.
$N \models F$ if and only if $N \cup \{\neg F\}$ is unsatisfiable.

Proposition 2.3.5:
$G \models \bot$ if and only if G is unsatisfiable.
$N \models \bot$ 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 in $\mathcal{A}$.

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.

Replacement Theorem

Proposition 2.3.6:
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 $\mathcal{A}(H[F]_p) = \mathcal{A}(H[G]_p)$.

Theorem 2.3.7:
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 $H[G]_p$.

Some Important Equivalences

Proposition 2.3.8:

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

$$
\begin{aligned}
(F \wedge F) &\;\;\models\!\!\!=\!\!|\;\; F && \\
(F \vee F) &\;\;\models\!\!\!=\!\!|\;\; F && \text{(Idempotence)} \\
(F \wedge G) &\;\;\models\!\!\!=\!\!|\;\; (G \wedge F) && \\
(F \vee G) &\;\;\models\!\!\!=\!\!|\;\; (G \vee F) && \text{(Commutativity)} \\
(F \wedge (G \wedge H)) &\;\;\models\!\!\!=\!\!|\;\; ((F \wedge G) \wedge H) && \\
(F \vee (G \vee H)) &\;\;\models\!\!\!=\!\!|\;\; ((F \vee G) \vee H) && \text{(Associativity)} \\
(F \wedge (G \vee H)) &\;\;\models\!\!\!=\!\!|\;\; ((F \wedge G) \vee (F \wedge H)) && \\
(F \vee (G \wedge H)) &\;\;\models\!\!\!=\!\!|\;\; ((F \vee G) \wedge (F \vee H)) && \text{(Distributivity)}
\end{aligned}
$$

Some Important Equivalences

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

$$
\begin{array}{rcll}
(F \wedge (F \vee G)) & \models\mid & F & \\
(F \vee (F \wedge G)) & \models\mid & F & \text{(Absorption)} \\
(\neg\neg F) & \models\mid & F & \text{(Double Negation)} \\
\neg(F \wedge G) & \models\mid & (\neg F \vee \neg G) & \\
\neg(F \vee G) & \models\mid & (\neg F \wedge \neg G) & \text{(De Morgan's Laws)} \\
(F \wedge G) & \models\mid & F \text{ if } G \text{ is a tautology} & \\
(F \vee G) & \models\mid & \top \text{ if } G \text{ is a tautology} & \\
(F \wedge G) & \models\mid & \bot \text{ if } G \text{ is unsatisfiable} & \\
(F \vee G) & \models\mid & F \text{ if } G \text{ is unsatisfiable} & \text{(Tautology Laws)}
\end{array}
$$

Some Important Equivalences

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

$$(F \leftrightarrow G) \quad \models\mid \quad ((F \rightarrow G) \wedge (G \rightarrow F))$$

$$(F \leftrightarrow G) \quad \models\mid \quad ((F \wedge G) \vee (\neg F \wedge \neg G)) \qquad \text{(Equivalence)}$$

$$(F \rightarrow G) \quad \models\mid \quad (\neg F \vee G) \qquad \text{(Implication)}$$

An Important Entailment

Proposition 2.3.9:

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

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