Automated Reasoning*

Christoph Weidenbach

Summer Term 2010

Topics of the Course I

Propositional logic

language: syntax, semantics – orderings
calculi: DPLL-procedure
implementation: 2-watched literal, clause learning

Linear arithmetic

language: syntax, semantics
calculi: Fourier-Motzkin

Propositional logic modulo a theory T

syntax, semantics
calculi: DPLL(T)-procedure, ...
implementation: coupling

*This document contains the text of the lecture slides (almost verbatim) plus some additional information, mostly proofs of theorems that are presented on the blackboard during the course. It is not a full script and does not contain the examples and additional explanations given during the lecture. Moreover it should not be taken as an example how to write a research paper – neither stylistically nor typographically.

Topics of the Course II

First-order predicate logic with equality

syntax, semantics, model theory, ...
calculi: superposition (SUP)
implementation: sharing, indexing

First-order predicate logic with equality modulo a theory T

syntax, semantics, model theory, ...
calculi: SUP(T)
implementation: coupling

1 Propositional Logic

Propositional logic

- logic of truth values
- decidable (but NP-complete)
- can be used to describe functions over a finite domain
- important for hardware applications (e. g., model checking)

1.1 Syntax

- propositional variables
- logical symbols
  $\Rightarrow$ Boolean combinations

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 as follows:

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

Notational Conventions

- We omit brackets according to the following rules:
  - $\neg \quad >_p \quad \vee \quad >_p \quad \wedge \quad >_p \quad \to \quad >_p \quad \leftrightarrow$ (binding precedences)
  - $\vee$ and $\wedge$ are associative
  - $\to$ is right-associative,
    i. e., $F \to G \to H$ means $F \to (G \to H)$.

1.2 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 $\mathcal{A}$, the function $\mathcal{A}^* : \Sigma$-formulas $\rightarrow \{0,1\}$ is defined inductively over the structure of F as follows:

$$\begin{aligned}
\mathcal{A}^*(\bot) &= 0 \\
\mathcal{A}^*(\top) &= 1 \\
\mathcal{A}^*(P) &= \mathcal{A}(P) \\
\mathcal{A}^*(\neg F) &= \mathsf{B}_\neg(\mathcal{A}^*(F)) \\
\mathcal{A}^*(F \rho G) &= \mathsf{B}_\rho(\mathcal{A}^*(F), \mathcal{A}^*(G))
\end{aligned}$$

where B_ρ is the Boolean function associated with ρ defined by the usual truth table.

For simplicity, we write $\mathcal{A}$ instead of $\mathcal{A}^*$.

We also write ρ instead of B_ρ, i. e., we use the same notation for a logical symbol and for its meaning (but remember that formally these are different things.)

1.3 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 $\mathcal{A}$, whenever $\mathcal{A} \models F$ then $\mathcal{A} \models G$.

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

Proposition 1.1 *$F \models G$ if and only if $\models (F \rightarrow G)$.(Proof follows)*

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

($\Leftarrow$) 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, $\mathcal{A}(F \to G) = \mathsf{B}_{\to}(\mathcal{A}(F), \mathcal{A}(G)) = \mathsf{B}_{\to}(1, 0) = 0$, so $(F \to G)$ does not hold in $\mathcal{A}$. □

Proposition 1.2 *$F \models\!\models G$ if and only if $\models (F \leftrightarrow G)$.*

Proof. Follows from Prop. 1.1. □

Extension 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 $\mathcal{A} \models F$.

Validity vs. Unsatisfiability

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

Proposition 1.3 *F is valid if and only if $\neg F$ is unsatisfiable.(Proof follows)*

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

($\Leftarrow$) 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 $N \models F$ can be reduced to unsatisfiability:

Proposition 1.4 *$N \models F$ 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 deciadable (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 1.5 *Let F and G be equivalent formulas, let H be a formula in which F occurs as a subformula.*

Then H is equivalent to H' where H' is obtained from H by replacing the occurrence of the subformula F by G. (Notation: $H = H[F]$, $H' = H[G]$. Proof follows)

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 $H = H[F]$ equals $\bot$, $\top$, or P, then $H = F$, $H' = G$, and H and H' are equivalent by assumption.

If $H = H_1 \wedge H_2$, then either F equals H (this case is treated as above), or F is a subformula of H_1 or H_2. Without loss of generality, assume that F is a subformula of H_1, so $H = H_1[F] \wedge H_2$. By the induction hypothesis, $H_1[F]$ and $H_1[G]$ are equivalent. Hence, for every valuation $\mathcal{A}$, $\mathcal{A}(H') = \mathcal{A}(H_1[G] \wedge H_2) = \mathcal{A}(H_1[G]) \wedge \mathcal{A}(H_2) = \mathcal{A}(H_1[F]) \wedge \mathcal{A}(H_2) = \mathcal{A}(H_1[F] \wedge H_2) = \mathcal{A}(H)$.

The other boolean connectives are handled analogously. □

Some Important Equivalences

Proposition 1.6 *The following equivalences are valid for all formulas F, G, H:*

$$(F \wedge F) \leftrightarrow F$$
$$(F \vee F) \leftrightarrow F \qquad \textit{(Idempotency)}$$
$$(F \wedge G) \leftrightarrow (G \wedge F)$$
$$(F \vee G) \leftrightarrow (G \vee F) \qquad \textit{(Commutativity)}$$
$$(F \wedge (G \wedge H)) \leftrightarrow ((F \wedge G) \wedge H)$$
$$(F \vee (G \vee H)) \leftrightarrow ((F \vee G) \vee H) \qquad \textit{(Associativity)}$$
$$(F \wedge (G \vee H)) \leftrightarrow ((F \wedge G) \vee (F \wedge H))$$
$$(F \vee (G \wedge H)) \leftrightarrow ((F \vee G) \wedge (F \vee H)) \qquad \textit{(Distributivity)}$$

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

$$(F \leftrightarrow G) \leftrightarrow ((F \rightarrow G) \wedge (G \rightarrow F)) \qquad \textit{(Equivalence)}$$
$$(F \rightarrow G) \leftrightarrow (\neg F \vee G) \qquad \textit{(Implication)}$$