A Propositional Calculus in Natural Deduction Style
DRAFT FOR THE LECTURE ON LOGIC, DEC. 2004

Tudor Jebelean
RISC-Linz
Johannes Kepler University, A-4040 Linz, Austria
www.risc.uni-linz.ac.at

Abstract

We construct a simple propositional calculus based on rewrite rules which correspond to the natural rules used by human provers. This is in fact a formalization of the propositional prover implemented in the Theorema system by Buchberger, and later improved by Jebelean. The purpose of the formalization is to show that the propositional prover of Theorema is correct and complete, and also to investigate possible improvements concerning efficiency and naturalness. The main differences from the traditional propositional theory consist in applying the logical connectives "and" and "or" to sets, and in the treatment of sequents as syntactical alternatives to logical formulae.

1 Syntax and Semantics

Let $\mathcal{V}$ be an enumerable set, whose elements will be called *propositional variables* (e.g. P, Q, R, ...).

Definition (F). The set $\mathcal{F}$ of *propositional formulae* is the smallest set having the properties:

(FV) $\mathcal{V} \subseteq \mathcal{F}$ (expressions consisting of one variable);

(F$\neg$) $\neg A \in \mathcal{F}$ for any $A \in \mathcal{F}$;

(F$\wedge$) $\wedge\mathcal{A} \in \mathcal{F}$ for any finite $\mathcal{A} \subset \mathcal{F}$

(F$\vee$) $\vee\mathcal{A} \in \mathcal{F}$ for any finite $\mathcal{A} \subset \mathcal{F}$

Formulae of type $P \in \mathcal{V}$ are called *atoms*, formulae of type P and $\neg P$ (also denoted $\overline{P}$) are called *literals.* Note that in (S$\wedge$) and (S$\vee$) $\mathcal{A}$ may be empty: we will sometimes denote the constants $\wedge\emptyset$ and $\vee\emptyset$ by $\mathbf{T}$ (*true*) and $\mathbf{F}$ (*false*), respectively.

Definition (I). An *interpretation* I is a subset of $\mathcal{V}$: $I \subset \mathcal{V}$. Given an interpretation I, the truth value of a formula A under the interpretation I $\langle A\rangle_I$ is defined by:

(IV) for any $P \in \mathcal{V}$: $\langle P\rangle_I = \mathbf{T}$ if $P \in I$, and $\mathbf{F}$ otherwise;

(I$\neg$) for any $A \in \mathcal{F}$: $\langle \neg A\rangle_I = \mathbf{T}$ if $\langle A\rangle_I = \mathbf{F}$, and $\mathbf{F}$ otherwise;

(I$\wedge$) for any $\mathcal{A} \subset \mathcal{F}$: $\langle \wedge\mathcal{A}\rangle_I = \mathbf{T}$ if $\langle A\rangle_I = \mathbf{T}$ for any $A \in \mathcal{A}$, and $\mathbf{F}$ otherwise;

(I$\vee$) for any $\mathcal{A} \subset \mathcal{F}$: $\langle \vee\mathcal{A} \rangle_I = \mathbf{T}$ if $\langle A \rangle_I = \mathbf{T}$ for some $A \in \mathcal{A}$, and $\mathbf{F}$ otherwise;

The notions: *true under I, satisfied by I, I-true, I is a model of A, false under I, I-false,* are defined in the usual way, as are the notions of *validity, inconsistency, satisfiability, and [semantic] equivalence* for formulae.

Note that $\wedge\emptyset$ ($\mathbf{T}$) is valid and $\vee\emptyset$ ($\mathbf{F}$) is inconsistent.

We denote formulae of the type $\vee\{G, \neg\wedge\mathcal{A}\}$ by $\langle G, \mathcal{A} \rangle$, and call them *proof-situations.* We call *assumptions* the formulae in $\mathcal{A}$ and *goal* the formula G. Such a proof situation is usually called *sequent* in the traditional theory, but note that we only consider sequents which have *one single* formula as the goal, because this is more similar to the natural human proving style.

2 Proving by Rewriting

Our purpose is to design a rewriting system which constructs proofs for formulae of type $\langle G, \mathcal{A} \rangle$, i.e. for proof-situations. Simpler rewrite rules could be defined in a similar manner (e.g. for computing the con[dis]junctive normal form, for resolution proving, etc.), however we choose a particular system which is appropriate for developing proofs in "natural deduction style", i.e. imitating human provers.

In the sequel A, G denote (arbitrary) formulae, $\mathcal{A}, \mathcal{A}', \mathcal{G}$ denote sets of formulae (including the empty set), and $\overline{\mathcal{A}} = \{\neg A | A \in \mathcal{A}\}$ (note that $\overline{\emptyset} = \emptyset$). In the context of rewrite rules we will use expressions like $\{A\}\breve{\cup}\mathcal{A}$ in order to denote a set which contains A. Here the symbol $\breve{\cup}$ denotes *exclusive union* (union between disjoint sets) and should be understood as an abbreviation for stating that the respective rule applies if $A \notin \mathcal{A}$.

Definition ($\longrightarrow$). The rewriting relation "$\longrightarrow$" is defined by:

(A) Assumption-rules

(A.$\neg$.$\neg$)	$\langle G,\ \{\neg\neg A\}\breve{\cup}\mathcal{A} \rangle$	$\longrightarrow$	$\langle G,\ \{A\}\cup\mathcal{A} \rangle$	(delete double negation)
(A.$\neg$.$\wedge$)	$\langle G,\ \{\neg \wedge \mathcal{A}\}\breve{\cup}\mathcal{A}' \rangle$	$\longrightarrow$	$\langle G,\ \{\vee\overline{\mathcal{A}}\}\cup\mathcal{A}' \rangle$	(de Morgan)
(A.$\neg$.$\vee$)	$\langle G,\ \{\neg \vee \mathcal{A}\}\breve{\cup}\mathcal{A}' \rangle$	$\longrightarrow$	$\langle G,\ \{\wedge\overline{\mathcal{A}}\}\cup\mathcal{A}' \rangle$	(de Morgan)
(A.$\wedge$)	$\langle G,\ \{\wedge\mathcal{A}\}\breve{\cup}\mathcal{A}' \rangle$	$\longrightarrow$	$\langle G,\ \mathcal{A}\cup\mathcal{A}' \rangle$	(split conjunction)
(A.$\vee$)	$\langle G,\ \{\vee\mathcal{A}\}\breve{\cup}\mathcal{A}' \rangle$	$\longrightarrow$	$\wedge\{\langle G,\ \{A\}\cup\mathcal{A}' \rangle \mid A \in \mathcal{A}\}$	(proof by cases)

(G) Goal-rules

(G.$\neg$)	$\langle \neg G,\ \mathcal{A} \rangle$	$\longrightarrow$	$\langle \mathbf{F},\ \{G\}\cup\mathcal{A} \rangle$	(proof by contradiction)
(G.$\wedge$)	$\langle \wedge\mathcal{G},\ \mathcal{A} \rangle$	$\longrightarrow$	$\wedge\{\langle G, \mathcal{A} \rangle \mid G \in \mathcal{G}\rangle\}$	(prove each part of a conjunction)
(G.$\vee$)	$\langle \vee\{G\}\breve{\cup}\mathcal{G},\ \mathcal{A} \rangle$	$\longrightarrow$	$\langle G,\ \overline{\mathcal{G}}\cup\mathcal{A} \rangle$	(prove one alternative by negating the others)

Note the particular behaviour of some rules when $\mathcal{A}$ or $\mathcal{G}$ is empty:

(A.$\neg$.$\wedge$)$_\emptyset$	$\langle G,\ \{\neg\mathbf{T}\}\breve{\cup}\mathcal{A} \rangle$	$\longrightarrow$	$\langle G,\ \{\mathbf{F}\}\cup\mathcal{A} \rangle$	("not true" rewrites to "false")
(A.$\neg$.$\vee$)$_\emptyset$	$\langle G,\ \{\neg\mathbf{F}\}\breve{\cup}\mathcal{A} \rangle$	$\longrightarrow$	$\langle G,\ \{\mathbf{T}\}\cup\mathcal{A} \rangle$	("not false" rewrites to "true")
(A.$\wedge$)$_\emptyset$	$\langle G,\ \{\mathbf{T}\}\breve{\cup}\mathcal{A} \rangle$	$\longrightarrow$	$\langle G,\ \mathcal{A}' \rangle$	(eliminate "true" assumption)
(A.$\vee$)$_\emptyset$	$\langle G,\ \{\mathbf{F}\}\breve{\cup}\mathcal{A} \rangle$	$\longrightarrow$	$\mathbf{T}$	(anything follows from "false" assumption)
(G.$\wedge$)$_\emptyset$	$\langle \mathbf{T},\ \mathcal{A} \rangle$	$\longrightarrow$	$\mathbf{T}$	("true" follows from anything)

3 Correctness

It is straightforward to prove that each rule transforms a formula into an equivalent one. As illustration, we prove below one implication corresponding to (A.$\wedge$). (Note that some of the rules listed above are practically applied in the proof.)

Lemma. For any $G, \mathcal{A}, \mathcal{A}'$,

$$\langle G,\ \{\vee\mathcal{A}\}\cup\mathcal{A}'\rangle \quad \text{is equivalent to} \quad \wedge\{\langle G,\ \{A\}\cup\mathcal{A}'\rangle \mid A \in \mathcal{A}\}.$$

Proof. Let I be an arbitrary but fixed interpretation. We prove:

$$\langle G,\ \{\vee\mathcal{A}\}\cup\mathcal{A}'\rangle \text{ is } I\text{-true} \quad \textbf{iff} \quad \wedge\{\langle G,\ \{A\}\cup\mathcal{A}'\rangle \mid A \in \mathcal{A}\} \text{ is } I\text{-true}.$$

($\Rightarrow$) Assume $\vee\{G,\ \neg\wedge\{\vee\mathcal{A}\}\cup\mathcal{A}'\}$ is I-true.

Case 1 : G is I-true. Then $\vee\{G,\ \neg\wedge\{A\}\cup\mathcal{A}'\}$ is I-true for each A in $\mathcal{A}$.

Case 2 : $\neg\wedge\{\vee\mathcal{A}\}\cup\mathcal{A}'$ is I-true. Then $\wedge\{\vee\mathcal{A}\}\cup\mathcal{A}'$ is I-false, hence either $\vee\mathcal{A}$ or some A' in $\mathcal{A}'$ is I-false.

Case 2.1 : $\vee\mathcal{A}$ is I-false. Then each A in $\mathcal{A}$ is I-false, hence $\wedge\{A\}\cup\mathcal{A}'$ is I-false.

Case 2.2 : Some A' in $\mathcal{A}'$ is I-false. Then again the former holds.

Therefore, $\vee\{G,\ \neg\wedge\{A\}\cup\mathcal{A}'\}\}$ is I-true for all A in $\mathcal{A}$.

Consequently, $\wedge\{\vee\{G,\ \neg\wedge\{A\}\cup\mathcal{A}'\}\} \mid A \in \mathcal{A}\}$ is I-true.

($\Leftarrow$) ...(similarly).

The rewrite rules (A), (G) transform each proof-situation into another proof-situation or a (conjunctive) set of proof-situations. It is natural to represent this process as a *proof-tree*, in which the root is the initial proof-situation, while each node has as son[s] the proof-situation[s] in which it is rewritten. A node has no sons only if no rewrite rule can apply to it (that is, we consider only *complete* proof trees). Since, on one hand, the rewrite rules preserve the truth-value and, on the other hand, each set of "brothers" represents a conjunction, it is clear that a node is valid **iff** all his descendents are valid. Moreover, if the tree is finite, then validating the root is equivalent to validating all leaves.

4 Termination

We show now that each proof tree is finite. For this, by König's Lemma, it is enough to show that each path in the tree is finite, therefore we show that the rule which defines the son-father relationship is terminating. This relationship, which will be denoted by "$\longmapsto$", is defined by the same rules as "$\longrightarrow$", except for (A.$\vee$) and (G.$\wedge$), which have to be replaced by:

$$\begin{array}{llcll} (\text{A}.\vee)\text{'} & \langle G,\ \{\vee\mathcal{A}\}\check{\cup}\mathcal{A}'\rangle & \longmapsto & \langle G,\ \{A\}\cup\mathcal{A}'\rangle & \text{iff } A \in \mathcal{A} \\ (\text{G}.\wedge)\text{'} & \langle\wedge\mathcal{G},\ \mathcal{A}\rangle & \longmapsto & \langle G,\ \mathcal{A}\rangle & \text{iff } G \in \mathcal{G} \end{array}$$

The special cases $(\text{A}.\vee)_\emptyset$ and $(\text{G}.\wedge)_\emptyset$ do not hold for $\longmapsto$, but this does not affect termination.

Let us consider separately the relation $\longmapsto_A$ defined on the set of assumptions by the A-rules and the relation $\longmapsto_G$ defined on the goals by the G-rules. Then $\longmapsto$ is a subset of their lexicografic composition:

$$\langle G, \mathcal{A}\rangle \longmapsto_{G,A} \langle G', \mathcal{A}'\rangle \ \textbf{ iff } \begin{cases} G \longmapsto_G G' \\ \textbf{or} \\ G = G' \textbf{ and } \mathcal{A} \longmapsto_A \mathcal{A}' \end{cases}$$

hence it is enough to prove termination for $\longmapsto_A$ and $\longmapsto_G$: $\longmapsto_A$ is normalization combined with $\vee$ elimination, (G.$\neg$) is terminal, and (G.$\wedge$), (G.$\vee$)' shorten G.

5 Completeness

Each proof-tree has a finite number of leaves, which will be called *terminal proof-situations (TPS)*. Examining the right-hand-side of $\longrightarrow$ we see that a TPS can have only one of the forms $\langle G, \mathcal{A}\rangle$ or $\mathbf{T}$.

Lemma. A formula X is a TPS **iff**:

(1) $X \;=\; \mathbf{T}$ **or**

(2) $X \;=\; \langle \mathbf{F}, \mathcal{L}\rangle$ for some set of literals $\mathcal{L}$ **or**

(3) $X \;=\; \langle P, \mathcal{L}\rangle$ for some atom P and some set of literals $\mathcal{L}$

($\Leftarrow$). By examining the rewrite rules, one sees that formulae of the above types cannot be rewritten.
($\Rightarrow$). Let X be a TPS which is not $\mathbf{T}$. Then $X = \langle G, \mathcal{A}\rangle$ for some formula G and some set of formulae $\mathcal{A}$. If G is neither an atom, nor $\mathbf{F}$, then one of the (G)-rules will apply to it. Also, if $\mathcal{A}$ is not empty, then any member of it must be of the form P or $\neg P$, otherwise one of the (A)-rules will apply.

Lemma. $\langle \vee\emptyset, \mathcal{L}\rangle$ is valid **iff** $\{P, \neg P\} \subset \mathcal{L}$ for some atom P.
Proof. $\langle \mathbf{F}, \mathcal{L}\rangle$ denotes the formula $\vee\{\mathbf{F}, \neg\wedge\mathcal{L}\}$, which is equivalent (by $\vee$-flattening) to $\neg\wedge\mathcal{L}$.
If $\{P, \neg P\} \subset \mathcal{L}$ then $\wedge\mathcal{L}$ is obviously inconsistent.
Conversely, if $\wedge\mathcal{L}$ is inconsistent, then $L\neg = \emptyset$. Consider $\mathcal{L}_+$ the set of positive atoms in $\mathcal{L}$. If no P from $\mathcal{L}_+$ has its opposite in $\mathcal{L}$, then $\mathcal{L}_+$ is an interpretation which satisfies $\wedge\mathcal{L}$.

Lemma. $\langle Q, \mathcal{L}\rangle$ is valid **iff** $Q \in \mathcal{L}$ **or** $\{P, \neg P\} \subset \mathcal{L}$ for some atom P.
Proof. $\langle Q, \mathcal{L}\rangle$ denotes the formula $\vee\{Q, \neg\wedge\mathcal{L}\}$
The right-to-left implication is obvious.
For the converse, let us assume $\vee\{Q, \neg\wedge\mathcal{L}\}$ is valid, and Q is not in $\mathcal{L}$. Then $\mathcal{L}\neg = \emptyset$ and one can construct, as in the previous proof, an interpretation which satisfies $\mathcal{L}$ but does not contain Q.

Since, on one hand, the rewrite rules preserve the truth-value and, on the other hand, each set of "brothers" in the proof-tree represents a conjunction, it is clear that a node is valid **iff** all his descendents are valid. Moreover, if the tree is finite, then validating the root is equivalent to validating all TPSs, which is simple to perform by the criteria given by the last lemmatas. Therefore, the rewrite system can be already considered a proof procedure, which is terminating, *complete* and *correct*, no matter in which order the rules are applied.

Example. A proof tree is presented in Fig. 1.

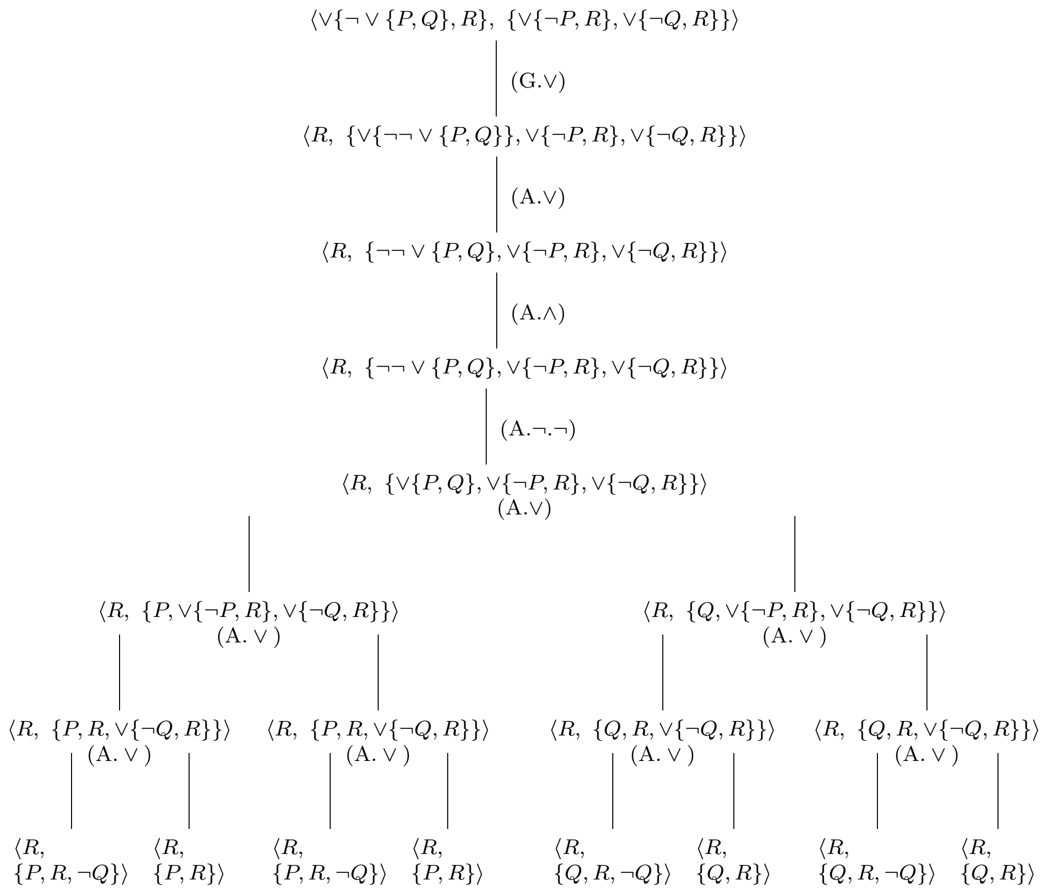
$\langle \vee\{\neg\vee\{P,Q\},R\},\ \{\vee\{\neg P,R\},\vee\{\neg Q,R\}\}\rangle$
(G.∨)
$\langle R,\ \{\vee\{\neg\neg\vee\{P,Q\}\},\vee\{\neg P,R\},\vee\{\neg Q,R\}\}\rangle$
(A.∨)
$\langle R,\ \{\neg\neg\vee\{P,Q\},\vee\{\neg P,R\},\vee\{\neg Q,R\}\}\rangle$
(A.∧)
$\langle R,\ \{\neg\neg\vee\{P,Q\},\vee\{\neg P,R\},\vee\{\neg Q,R\}\}\rangle$
(A.¬.¬)
$\langle R,\ \{\vee\{P,Q\},\vee\{\neg P,R\},\vee\{\neg Q,R\}\}\rangle$
(A.∨)
$\langle R,\ \{P,\vee\{\neg P,R\},\vee\{\neg Q,R\}\}\rangle$
(A. ∨)
$\langle R,\ \{Q,\vee\{\neg P,R\},\vee\{\neg Q,R\}\}\rangle$
(A. ∨)
$\langle R,\ \{P,R,\vee\{\neg Q,R\}\}\rangle$
(A. ∨)
$\langle R,\ \{P,R,\vee\{\neg Q,R\}\}\rangle$
(A. ∨)
$\langle R,\ \{Q,R,\vee\{\neg Q,R\}\}\rangle$
(A. ∨)
$\langle R,\ \{Q,R,\vee\{\neg Q,R\}\}\rangle$
(A. ∨)
$\langle R,\ \{P,R,\neg Q\}\rangle$
$\langle R,\ \{P,R\}\rangle$
$\langle R,\ \{P,R,\neg Q\}\rangle$
$\langle R,\ \{P,R\}\rangle$
$\langle R,\ \{Q,R,\neg Q\}\rangle$
$\langle R,\ \{Q,R\}\rangle$
$\langle R,\ \{Q,R,\neg Q\}\rangle$
$\langle R,\ \{Q,R\}\rangle$

Figure 1: A proof-tree.

6 Possible Extensions

It is possible to add more formulae and rules to the system, in order to obtain *simpler* and *more natural* proofs. The new formulae and rules will be “short-hands” for the initial formulae and for the rules (A), (G) and for the TPS validity checks, therefore they do not change the termination, completeness and correctness of the system.

First one can add some *terminal* rules which perform the TPS validity check:
Definition (T). Terminal-rules are defined by:

(T.pos) $\langle G, \{G\}\breve{\cup}\mathcal{A}\rangle \longrightarrow \mathbf{T}$,

(T.neg) $\langle G, \{A, \neg A\}\breve{\cup}\mathcal{A}\rangle \longrightarrow \mathbf{T}$

for any formulae A, G and any set of formulae $\mathcal{A}$.
These rules will rewrite each terminal proof situation into $\mathbf{T}$, hence we have the following:

Theorem.
If a proof-situation is valid, then any proof-tree of it has only **T**-leaves.
If the proof-tree of a proof-situation has only **T**-leaves, then the proof-situation is valid.

Furthermore one can add the resolution rule (in order to make the calculus more efficient), and one can enrich the syntax by introducing more logical connectives and their corresponding rules.