A Sequent Based On-the-fly Procedure to Get Hilbert Proofs in Classical Propositional Logic

Mauro Ferrari
DiSTA, Univ. degli Studi dell'Insubria,
Via Mazzini, 5, 21100, Varese, Italy
mauro.ferrari@uninsubria.it

Camillo Fiorentini
DI, Univ. degli Studi di Milano,
Via Celoria 18, 20133 Milano, Italy
fiorentini@di.unimi.it

Guido Fiorino
DISCO, Univ. degli Studi di Milano-Bicocca,
Viale Sarca 336, 20126 Milano, Italy
guido.fiorino@unimib.it

***Abstract*—In this paper, we present a preliminary result on the generation of Hilbert proofs for classical propositional logic. This is part of our ongoing research on the comparison of proof-search in different proof-systems. Exploiting the notion of evaluation function, we define a fully deterministic terminating decision procedure that returns either a derivation in the Hilbert calculus of the given goal or a counter model witnessing its unprovability.**

Keywords–Automated Theorem Proving; Hilbert calculi.

I. INTRODUCTION

It is well-known [1] that the standard formalizations of classical and intuitionistic logic based on Hilbert calculi, sequent calculi and natural deduction are equivalent. In spite of this, proof-search has been mainly developed around the notion of sequent calculi, almost neglecting the cases of natural deduction and Hilbert calculi. This is primarily motivated by the fact that the latters lack the *structural properties* of sequent calculi, like cut-elimination and subformula property, that can be immediately exploited to define a goal-oriented proof-search procedure (see, e.g., [2]-[3] for an accurate discussion). In [4][5] it is shown that in the case of natural deduction it is possible to design proof-search procedures with structural and complexity properties comparable with those based on sequent calculi. This is obtained by imposing a strict discipline on the backward application of the rules and by exploiting some internal features of the goal via evaluation functions. In this paper, we begin an investigation concerning Hilbert calculi with the aim to apply in this context the ideas developed in the above cited papers. We take Classical Propositional Logic (CPL) as the entry point of our investigation. We consider the Hilbert system **Hc** for CPL as defined in [6] and we introduce a proof-search strategy for it. The procedure builds Hilbert proofs during the proof-search phase. Since the proof-search phase is based on a strategy for a sequent calculus, the procedure can be seen as building Hilbert proofs in one-pass, that is, by translating on-the-fly sequent proofs. The procedure relies on a sequent calculus with at most one formula on the right, thus to get termination and completeness we need to introduce the machinery of *evaluations*.

As regards related works, as far as we know, the only two papers addressing this issue are [7][8], and both are scarcely related with our approach. The former deals with backward application of modus ponens, introducing metavariables to represents cut-formulas. In [8], an implementation of a prover for CPL is presented, that relies on a semantic method exploiting Kalmar completeness theorem.

The system **Hc** consists of some axioms and only one rule, *modus ponens* (from $A \to B$ and A infer B). The main problem in defining a proof-search procedure for **Hc** is to control the application of modus ponens. If we backward apply modus ponens to prove a goal formula B, we have to guess a cut-formula A, so that next goals are A and $A \to B$. To avoid considering useless cut-formulas, we adapt to the classical case the notion of evaluation function introduced in [9] for intuitionistic propositional logic. Essentially, an evaluation function is a lightweight computational mechanism that drives the backward application of the rules, only analyzing the current goal of the proof search. The evaluation function for **Hc** is introduced in Section III. The proof-search procedure described in Section IV takes as input a goal G and a set of assumptions Γ and returns either a deduction of G in **Hc** from assumptions Γ or a classical model of Γ falsifying G. As proved in Section IV the proof-search procedure is terminating and it does not require backtracking.

II. THE HILBERT CALCULUS **Hc**

We consider the propositional language $\mathcal{L}$ of CPL based on a denumerable set of propositional variables Pv and the connectives $\wedge$, $\vee$, $\to$ and $\neg$. We write $A \leftrightarrow B$ as a shorthand for $(A \to B) \wedge (B \to A)$. A *literal* is a formula of the form p or $\neg p$ with $p \in \mathrm{Pv}$; the set of literals is denoted by Lit. The size of a formula A, denoted by $|A|$, is the number of logical symbols occurring in A. A *model* $\mathcal{M}$ is a subset of Pv; we write $\mathcal{M} \vDash A$ to denote that the formula A is *valid in* $\mathcal{M}$, according to the usual definition. Let Γ be a set of formulas; $\mathcal{M} \vDash \Gamma$ iff, for every $A \in \Gamma$, $\mathcal{M} \vDash A$. By $\Gamma \vDash A$ we mean that A is a *classical consequence* of Γ, namely: for every model $\mathcal{M}$, $\mathcal{M} \vDash \Gamma$ implies $\mathcal{M} \vDash A$.

We call **Hc** the Hilbert calculus for CPL introduced in [6]. Logical axioms of **Hc** are:

(Ax1) $A \to (B \to A)$
(Ax2) $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$
(Ax3) $A \to (B \to (A \wedge B))$
(Ax4a) $(A \wedge B) \to A$ (Ax4b) $(A \wedge B) \to B$
(Ax5a) $A \to (A \vee B)$ (Ax5b) $B \to (A \vee B)$
(Ax6) $(A \to C) \to ((B \to C) \to ((A \vee B) \to C))$
(Ax7) $(A \to B) \to ((A \to \neg B) \to \neg A)$
(Ax8) $\neg\neg A \to A$

The only rule of **Hc** is MP (*Modus Ponens*): from $A \to B$ and A infer B.

 ISBN: 978-1-61208-709-2

Let Γ be a set of formulas and A a formula. A *deduction* of A from assumptions Γ is a finite sequence of formulas $\mathcal{D} = \langle A_1, \ldots, A_n \rangle$ (possibly with repetitions) such that $A_n = A$ and, for every A_i in the sequence, one of the following conditions holds:

(a) $A_i \in \Gamma$ (namely, A_i is an assumption);
(b) A_i is an instance of a logical axiom;
(c) A_i is obtained by applying MP to A_j and A_k, with $j < i$ and $k < i$.

We denote with $\mathrm{Fm}(\mathcal{D}) = \{A_1, \ldots, A_n\}$ the set of formulas occurring in $\mathcal{D}$. By $\mathcal{D} : \Gamma \vdash A$ we mean that $\mathcal{D}$ is a deduction of A from assumptions in Γ. Given two sets of formulas $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$, we write $\mathcal{D} : A_1, \ldots, A_n, B_1, \ldots, B_m \vdash A$ instead of $\mathcal{D} : \{A_1, \ldots, A_n\} \cup \{B_1, \ldots, B_m\} \vdash A$.

Given two deductions $\mathcal{D}_1 : \Gamma_1 \vdash A_1$ and $\mathcal{D}_2 : \Gamma_2 \vdash A_2$, we denote by $\mathcal{D}_1 \circ \mathcal{D}_2$ the deduction obtained by concatenating the sequences $\mathcal{D}_1$ and $\mathcal{D}_2$. One can easily check that $\mathcal{D}_1 \circ \mathcal{D}_2 : \Gamma \vdash A_2$, where $\Gamma \subseteq \Gamma_1 \cup \Gamma_2$. Note that Γ can be a proper subset of $\Gamma_1 \cup \Gamma_2$, since some of assumptions in Γ_2 could be obtained by applying MP in $\mathcal{D}_1 \circ \mathcal{D}_2$ (see the next example).

Example 1: Let us consider the derivations

$$\mathcal{D}_1 = \langle p_1,\ p_1 \to p_2 \rangle \qquad \mathcal{D}_2 = \langle p_2,\ p_2 \to (p_2 \vee q),\ p_2 \vee q \rangle$$

We have $\mathcal{D}_1 : p_1, p_1 \to p_2 \vdash p_1 \to p_2$. *In* $\mathcal{D}_2$, *the formula* $p_2 \to (p_2 \vee q)$ *is an instance of axiom* $(\mathrm{Ax}5a)$ *and* $p_2 \vee q$ *is obtained by applying* MP *to* $p_2 \to (p_2 \vee q)$ *and* p_2, *hence* $\mathcal{D}_2 : p_2 \vdash p_2 \vee q$. *In the concatenation* $\mathcal{D}_1 \circ \mathcal{D}_2$ *the formula* p_2 *can be obtained by applying* MP *to* $p_1 \to p_2$ *and* p_1, *hence* $\mathcal{D}_1 \circ \mathcal{D}_2 : p_1, p_1 \to p_2 \vdash p_2 \vee q$.

In the following proposition, we introduce some deduction schemas we use later on.

Lemma 1: For all formulas A, B and C, the following deductions can be constructed:

(i) $\mathcal{D}_{\mathrm{MP}}(A \to B, A) : A \to B, A \vdash B$;

(ii) $\mathcal{D}_{\neg\neg\mathrm{E}}(A) : \neg\neg A \vdash A$;

(iii) $\mathcal{D}_{\mathrm{EF}}(A, B) : A, \neg A \vdash B$;

(iv) $\mathcal{D}_{\mathrm{EF}\to}(A, B) : \neg A \to B, \neg A \to \neg B \vdash A$;

(v) $\mathcal{D}_{\mathrm{EF}\to\neg}(A, B) : A \to B, A \to \neg B \vdash \neg A$;

(vi) $\mathcal{D}_{\vee\mathrm{E}}(A, B, C) : A \to C, B \to C, A \vee B \vdash C$.

Proof: The following sequence of formulas proves Point (vi):

(1)	$A \to C$	Assumption
(2)	$B \to C$	Assumption
(3)	$A \vee B$	Assumption
(4)	$(A \to C) \to ((B \to C) \to ((A \vee B) \to C))$	(Ax6)
(5)	$(B \to C) \to ((A \vee B) \to C)$	MP (4) (1)
(6)	$(A \vee B) \to C$	MP (5) (2)
(7)	C	MP (6) (3)

■

A distinguishing feature of Hilbert calculi is that there are no rules to close assumptions. Thus, to prove the Deduction Lemma for **Hc** we have to rearrange a deduction $\mathcal{D}$ of $A, \Gamma \vdash B$ to get a deduction of $\Gamma \vdash A \to B$. An analogous issue holds for negation elimination and negation introduction. The following lemma provides the schemas of derivation transformations treating such cases:

Lemma 2 (Closing assumption lemma):

(i) Let $\mathcal{D} : A, \Gamma \vdash B$. Then, there exists a deduction $\mathcal{E}_{\mathrm{DL}}(\mathcal{D}, A) : \Gamma \vdash A \to B$ such that, for every $C \in \mathrm{Fm}(\mathcal{D})$, $A \to C \in \mathrm{Fm}(\mathcal{E}_{\mathrm{DL}}(\mathcal{D}))$.

(ii) Let $\mathcal{D} : \neg A, \Gamma \vdash K$ such that $\neg K \in \mathrm{Fm}(\mathcal{D})$. Then, there exists a deduction $\mathcal{E}_{\neg\mathrm{E}}(\mathcal{D}, \neg A) : \Gamma \vdash A$.

(iii) Let $\mathcal{D} : A, \Gamma \vdash K$ such that $\neg K \in \mathrm{Fm}(\mathcal{D})$. Then, there exists a deduction $\mathcal{E}_{\neg\mathrm{I}}(\mathcal{D}, A) : \Gamma \vdash \neg A$.

Proof: For Point (i) see the proof of the Deduction Lemma in [6]. As for Point (ii), by Point (i) there exists a deduction $\mathcal{E}_{\mathrm{DL}}(\mathcal{D}, \neg A) : \Gamma \vdash \neg A \to K$ such that $\neg A \to \neg K \in \mathrm{Fm}(\mathcal{E}_{\mathrm{DL}}(\mathcal{D}, \neg A))$. Let us consider the derivation $\mathcal{D}_{\mathrm{EF}\to}(A, K) : \neg A \to K, \neg A \to \neg K \vdash A$ defined in Lemma 1.(iv). We set $\mathcal{E}_{\neg\mathrm{E}}(\mathcal{D}, \neg A) : \Gamma \vdash A$ as the deduction $\mathcal{E}_{\mathrm{DL}}(\mathcal{D}, \neg A) \circ \mathcal{D}_{\mathrm{EF}\to}(A, K)$. The deduction $\mathcal{E}_{\neg\mathrm{I}}(\mathcal{D}, A)$ of Point (iii) is built in a similarly using $\mathcal{D}_{\mathrm{EF}\to\neg}(A, K)$ instead of $\mathcal{D}_{\mathrm{EF}\to}(A, K)$. ■

III. THE EVALUATION MECHANISM

Evaluation functions have been introduced in [9] to get a terminating proof-search procedure for Gentzen sequent calculus **LJ** for propositional intuitionistic logic. Here, we adapt this mechanism to the case of classical propositional logic.

Let $\mathcal{L}_{\mathtt{t},\mathtt{f}}$ be the language obtained by adding to $\mathcal{L}$ the constants `t` (true) and `f` (false), with the usual interpretation; let H be a formula of $\mathcal{L}_{\mathtt{t},\mathtt{f}}$ and let Γ be a set of formulas of $\mathcal{L}$. The *evaluation of* H *in* Γ, denoted by $\mathbf{eval}(H, \Gamma)$, is the formula of $\mathcal{L}_{\mathtt{t},\mathtt{f}}$ obtained by replacing every subformula K of H such that K is an axiom or $K \in \Gamma$ with `t` and then by performing some truth preserving boolean simplifications (for instance, any subformula of the kind $\mathtt{t} \vee K$ is replaced by `t`, while $\mathtt{f} \vee K$ is replaced by K). The function $\mathbf{eval}$ and the auxiliary function $\mathbf{simpl}$ are defined by mutual induction in Figure 1. We point out that $\mathbf{eval}(H, \Gamma)$ is either `t` or `f` or a formula of $\mathcal{L}$ (not containing the constants `t`, `f`). Moreover, $|\mathbf{eval}(H, \Gamma)| \leq |H|$.

One can easily prove that, assuming Γ, the formulas H, $\mathbf{eval}(H, \Gamma)$ and $\mathbf{simpl}(H, \Gamma)$ are classically equivalent, as stated in the next lemma

Lemma 3: Let Γ be a set of formulas of $\mathcal{L}$ and H a formula of $\mathcal{L}_{\mathtt{t},\mathtt{f}}$. Then:

1) $\Gamma \vDash H \leftrightarrow \mathbf{eval}(H, \Gamma)$;

2) $\Gamma \vDash H \leftrightarrow \mathbf{simpl}(H, \Gamma)$.

By Lemma 3 we immediately get:

Theorem 1: Let Γ be a set of formulas of $\mathcal{L}$, let A be a formula of $\mathcal{L}$ and $\mathcal{M}$ a model such that $\mathcal{M} \vDash \Gamma$.

1) $\mathbf{eval}(A, \Gamma) = \mathtt{t}$ implies $\mathcal{M} \vDash A$.

 ISBN: 978-1-61208-709-2

$$\mathbf{eval}(H,\Gamma) \;=\; \begin{cases} \mathtt{t} & \text{if } H \text{ is an axiom or } H \in \Gamma \\ \mathtt{f} & \text{if } \neg H \in \Gamma \\ H & \text{if } H \in (\mathrm{Pv} \setminus \Gamma) \cup \{\mathtt{t},\mathtt{f}\} \\ \mathbf{simpl}(H,\Gamma) & \text{otherwise} \end{cases}$$

$$\text{in the following } K_1 = \mathbf{eval}(H_1,\Gamma) \text{ and } K_2 = \mathbf{eval}(H_2,\Gamma)$$

$$\mathbf{simpl}(\neg H_1,\Gamma) \;=\; \begin{cases} \mathtt{t} & \text{if } K_1 = \mathtt{f} \\ \mathtt{f} & \text{if } K_1 = \mathtt{t} \\ \neg K_1 & \text{otherwise} \end{cases} \qquad \mathbf{simpl}(H_1 \wedge H_2,\Gamma) \;=\; \begin{cases} \mathtt{t} & \text{if } K_1 = \mathtt{t} \text{ and } K_2 = \mathtt{t} \\ \mathtt{f} & \text{if } K_1 = \mathtt{f} \text{ or } K_2 = \mathtt{f} \\ K_1 \wedge K_2 & \text{otherwise} \end{cases}$$

$$\mathbf{simpl}(H_1 \vee H_2,\Gamma) \;=\; \begin{cases} \mathtt{t} & \text{if } K_1 = \mathtt{t} \text{ or } K_2 = \mathtt{t} \\ \mathtt{f} & \text{if } K_1 = \mathtt{f} \text{ and } K_2 = \mathtt{f} \\ K_1 \vee K_2 & \text{otherwise} \end{cases} \qquad \mathbf{simpl}(H_1 \to H_2,\Gamma) \;=\; \begin{cases} \mathtt{t} & \text{if } K_1 = \mathtt{f} \text{ or } K_2 = \mathtt{t} \\ \mathtt{f} & \text{if } K_1 = \mathtt{t} \text{ and } K_2 = \mathtt{f} \\ K_1 \to K_2 & \text{otherwise} \end{cases}$$

Figure 1. Definition of $\mathbf{eval}$

2) $\mathbf{eval}(A,\Gamma) = \mathtt{f}$ implies $\mathcal{M} \vDash \neg A$.

By $\mathcal{L}^{\neg}$ we denote the set of formulas H such that H is a literal or $H = \neg(A \vee B)$. A set of formulas Γ is *reduced* iff the following properties hold:

- $\Gamma \subseteq \mathcal{L}^{\neg}$;
- for every $p \in \mathrm{Pv}$, $p \in \Gamma$ implies $\neg p \notin \Gamma$;
- for every $H = \neg(A \vee B) \in \Gamma$, $\mathbf{eval}(H, \Gamma \setminus \{H\}) = \mathtt{t}$.

By Theorem 1, we get:

Theorem 2: Let Γ be a reduced set of formulas. Then, $\Gamma \cap \mathrm{Pv}$ is a model of Γ.

Proof: Let Θ be the subset of Γ containing all the formulas of the kind $\neg(A \vee B)$ of Γ. We prove the theorem by induction on the cardinality of Θ. If Θ is empty, then Γ only contains propositional variables or formulas $\neg p$ such that $p \in \mathrm{Pv} \setminus \Gamma$; this implies that $\Gamma \cap \mathrm{Pv}$ is a model of Γ. Let $\neg(A \vee B) \in \Theta$, let $\Gamma_1 = \Gamma \setminus \{\neg(A \vee B)\}$ and let $\mathcal{M}$ be the model $\Gamma \cap \mathrm{Pv}$. By induction hypothesis, $\mathcal{M}$ is a model of Γ_1. Since Γ is reduced, we have $\mathbf{eval}(\neg(A \vee B), \Gamma_1) = \mathtt{t}$; by Theorem 1 we get $\mathcal{M} \vDash \neg(A \vee B)$, hence $\mathcal{M}$ is a model of Γ. ∎

A set of formulas Γ is *contradictory* iff there exists a formula X such that $\{X, \neg X\} \subseteq \Gamma$. In the proof of termination of the proof-search procedure **Hp**, we need the following properties of $\mathbf{eval}$.

Lemma 4: Let Γ be a non-contradictory set of formulas, let H be a formula and let us assume that $\mathbf{eval}(H,\Gamma) = \tau$, where $\tau \in \{\mathtt{t},\mathtt{f}\}$. Let $K \in \Gamma$ and $\Delta = \Gamma \setminus \{K\}$. Then:

1) If $K = \neg\neg A$ and $\Delta \cup \{A\}$ is not contradictory, then $\mathbf{eval}(H,\ \Delta \cup \{A\}) = \tau$.
2) If $K = A \wedge B$ and $\Delta \cup \{A, B\}$ is not contradictory, then $\mathbf{eval}(H,\ \Delta \cup \{A, B\}) = \tau$.
3) If $A_0 \vee A_1 \in \Gamma$ and , $\Delta \cup \{A_k\}$ is not contradictory, then $\mathbf{eval}(H,\ \Delta \cup \{A_k\}) = \tau$.
4) If $K = A \to B$ and $\Delta \cup \{\neg A\}$ is not contradictory, then $\mathbf{eval}(H,\ \Delta \cup \{\neg A\}) = \tau$.
5) If $K = A \to B$ and , $\Delta \cup \{B\}$ is not contradictory, then $\mathbf{eval}(H,\ \Delta \cup \{B\}) = \tau$.
6) If $K = \neg(A_0 \wedge A_1)$ and $k \in \{0,1\}$ and $\Delta \cup \{\neg A_k\}$ is not contradictory, then $\mathbf{eval}(H,\ \Delta \cup \{\neg A_k\}) = \tau$.
7) If $K = \neg(A_0 \vee A_1)$ and $\mathbf{eval}(K, \Delta) = \mathtt{t}$ and $k \in \{0,1\}$ and $\Delta \cup \{\neg A_k\}$ is not contradictory, then $\mathbf{eval}(H,\ \Delta \cup \{\neg A_k\}) = \tau$.
8) If $K = \neg(A_0 \vee A_1)$ and $\mathbf{eval}(A_0, \Delta) = \mathbf{eval}(A_1, \Delta) = \mathtt{f}$, then $\mathbf{eval}(H,\ \Delta) = \tau$.
9) If $\neg(A \to B) \in \Gamma$ and $\Delta \cup \{A, \neg B\}$ is not contradictory, then $\mathbf{eval}(H,\ \Delta \cup \{A, \neg B\}) = \tau$.

IV. Procedure **Hp**

We present the procedure **Hp** to search for a deduction in **Hc**. Let Γ be a set of formulas and let G be the goal formula or the special symbol $\Box$ representing the empty goal. We define the procedure **Hp** satisfying the following properties:

(H1) If $G \in \mathcal{L}$, $\mathbf{Hp}(\Gamma, G)$ returns either a deduction $\mathcal{D} : \Gamma \vdash G$ or a model of $\Gamma \cup \{\neg G\}$.
(H2) $\mathbf{Hp}(\Gamma, \Box)$ returns either a deduction $\mathcal{D} : \Gamma \vdash K$ such that $\neg K \in \mathrm{Fm}(\mathcal{D})$ or a model of Γ.

Procedure **Hp** (Γ, G)

(1) if G is an axiom or $G \in \Gamma$
    return $\langle G \rangle$
(2) if there is $\{\neg A, A\} \subseteq \Gamma$
    if $G \neq \Box$, then return the proof $\mathcal{D}_{\mathrm{EF}}(A, G)$
    else return $\langle \neg A, A \rangle$
(3) if there is $\neg\neg A \in \Gamma$
    let $\mathcal{D} = \mathbf{Hp}(\,(\Gamma \setminus \{\neg\neg A\}) \cup \{A\},\ G\,)$.
    if $\mathcal{D}$ is a model, then return $\mathcal{D}$
    else return $\langle \neg\neg A, \neg\neg A \to A \rangle \circ \mathcal{D}$
(4) if there is $A \wedge B \in \Gamma$

let $\mathcal{D} = \mathbf{Hp}(\,(\Gamma \setminus \{A \wedge B\}) \cup \{A, B\},\, G\,)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$
else return $\langle A \wedge B, (A \wedge B) \to A, (A \wedge B) \to B\rangle \circ \mathcal{D}$

(5) if there is $A_0 \vee A_1 \in \Gamma$ and $G = \Box$
for $i = 0, 1$, let $\mathcal{D}_i = \mathbf{Hp}(\,(\Gamma \setminus \{A_0 \vee A_1\}) \cup \{A_i\},\, \Box\,)$
if there exists $i \in \{0, 1\}$ such that $\mathcal{D}_i$ is a model, then return $\mathcal{D}_i$
else for $i = 0, 1$, let K_i the last formula of $\mathcal{D}_i$
let $\mathcal{D}'_1 = \mathcal{D}_1 \circ \mathcal{D}_{\mathrm{EF}}(K_1, \neg K_0) \circ \mathcal{D}_{\mathrm{EF}}(K_1, K_0)$
let $\mathcal{E}_0 = \mathcal{E}_{\mathrm{DL}}(\mathcal{D}_0, A_0)$
let $\mathcal{E}_1 = \mathcal{E}_{\mathrm{DL}}(\mathcal{D}'_1, A_1)$
return $\mathcal{E}_0 \circ \mathcal{E}_1 \circ \mathcal{D}_{\vee\mathrm{E}}(A_0, A_1, \neg K_0) \circ \mathcal{D}_{\vee\mathrm{E}}(A_0, A_1, K_0)$

(6) if there is $A_0 \vee A_1 \in \Gamma$ and $G \neq \Box$
for $i = 0, 1$, let $\mathcal{D}_i = \mathbf{Hp}((\Gamma \setminus \{A_0 \vee A_1\}) \cup \{A_i\}, G)$
if there exists $i \in \{0, 1\}$, such that $\mathcal{D}_i$ is a model, then return $\mathcal{D}_i$
else for $i = 0, 1$, let $\mathcal{D}'_i = \mathcal{E}_{\mathrm{DL}}(\mathcal{D}_i, A_i)$
return $\mathcal{D}'_0 \circ \mathcal{D}'_1 \circ \mathcal{D}_{\vee\mathrm{E}}(A_0, A_1, G)$

(7) if ($G = A_0 \vee A_1$ or $G \in \mathrm{Pv}$) and $\mathbf{eval}(G, \Gamma) \neq \mathtt{f}$
let $\mathcal{D} = \mathbf{Hp}(\,\Gamma \cup \{\neg G\}, \Box\,)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$
else return $\mathcal{E}_{\neg\mathrm{E}}(\mathcal{D}, \neg G)$

(8) if $G = A_0 \wedge A_1$
for $i = 0, 1$, let $\mathcal{D}_i = \mathbf{Hp}(\Gamma, A_i)$
if there exists $i \in \{0, 1\}$, such that $\mathcal{D}_i$ is a model, then return $\mathcal{D}_i$
else let $\mathcal{D}_2 = \langle\ A_0 \to (A_1 \to (A_0 \wedge A_1)),\ A_1 \to (A_0 \wedge A_1), A_0 \wedge A_1\rangle$
return $\mathcal{D}_0 \circ \mathcal{D}_1 \circ \mathcal{D}_2$

(9) if $G = A \to B$
let $\mathcal{D} = \mathbf{Hp}(\Gamma \cup \{A\}, B)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$, else return $\mathcal{E}_{\mathrm{DL}}(\mathcal{D}, A)$

(10) if $G = \neg A$
let $\mathcal{D} = \mathbf{Hp}(\Gamma \cup \{A\}, \Box)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$, else return $\mathcal{E}_{\neg\mathrm{I}}(\mathcal{D}, A)$

(11) if $G = A_0 \vee A_1$
// *here* $\mathbf{eval}(A_0 \vee A_1, \Gamma) = \mathtt{f}$
if $\mathbf{eval}(A_0, \Gamma) \neq \mathtt{f}$
let $\mathcal{D} = \mathbf{Hp}(\Gamma, A_0)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$
else return $\mathcal{D} \circ \langle A_0 \to (A_0 \vee A_1), A_0 \vee A_1\rangle$
if $\mathbf{eval}(A_1, \Gamma) \neq \mathtt{f}$
let $\mathcal{D} = \mathbf{Hp}(\Gamma, A_1)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$
else return $\mathcal{D} \circ \langle A_1 \to (A_0 \vee A_1), A_0 \vee A_1\rangle$
// *here* $\mathbf{eval}(A_0, \Gamma) = \mathbf{eval}(A_1, \Gamma) = \mathtt{f}$
let $\mathcal{D} = \mathbf{Hp}(\Gamma, \Box)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$
else let K be the last formula of $\mathcal{D}$
return $\mathcal{D} \circ \mathcal{D}_{\mathrm{EF}}(K, G)$

(12) if $G \in \mathrm{Pv}$ and $\neg G \in \Gamma$
let $\mathcal{D} = \mathbf{Hp}(\Gamma, \Box)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$
else let K be the last formula of $\mathcal{D}$
return $\mathcal{D} \circ \mathcal{D}_{\mathrm{EF}}(K, G)$
// *here* $G = \Box$

(13) if there is $\neg A \in \Gamma$ such that $A = B_0 \vee B_1$ and $\mathbf{eval}(\neg A, \Gamma \setminus \{\neg A\}) \neq \mathtt{t}$
let $\mathcal{D} = \mathbf{Hp}(\Gamma, A)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$
else return $\langle \neg A\rangle \circ \mathcal{D}$

(14) if there is $\neg A \in \Gamma$ such that $A \notin \mathrm{Pv}$ and $A \neq B_0 \vee B_1$
let $\mathcal{D} = \mathbf{Hp}(\,\Gamma \setminus \{\neg A\},\, A\,)$
if $\mathcal{D}$ is a model, then return $\mathcal{D}$
else return $\langle \neg A\rangle \circ \mathcal{D}$

(15) if there is $A \to B \in \Gamma$
let $\mathcal{D}_0 = \mathbf{Hp}(\,\Gamma \setminus \{A \to B\},\, A\,)$
let $\mathcal{D}_1 = \mathbf{Hp}(\,(\Gamma \setminus \{A \to B\}) \cup \{B\},\, G\,)$
if, for some $i \in \{0, 1\}$, $\mathcal{D}_i$ is a model, then return $\mathcal{D}_i$
else return $\mathcal{D}_0 \circ \langle A \to B\rangle \circ \mathcal{D}_1$
// *here* Γ *is reduced*

(16) return the model $\Gamma \cap \mathrm{Pv}$

Before discussing the properties of $\mathbf{Hp}$, we give a high-level overview of the proof-search procedure. Steps (1), (2) and (16) are the base cases of $\mathbf{Hp}$ and immediately return a value. In particular, Step (1) returns a derivation of G when it is either an axiom or an assumption and Step (2) returns the derivation of G when Γ is contradictory. As for Step (16), it is executed when steps (1)-(15) cannot be applied; thus, Γ is reduced and G is the empty goal $\Box$. As we discuss later (see Theorem 4), the returned value $\Gamma \cap \mathrm{Pv}$ is a model of Γ. Steps (3)-(6) act on an assumption $K \in \Gamma$; according with the form of K, these steps reduce the problem to prove $\Gamma \vdash G$ to the problem to prove $\Gamma' \vdash G$, where Γ' is obtained by replacing K with its relevant subformulas. These steps exploit axioms $(\mathrm{Ax}8)$, $(\mathrm{Ax}4a)$, $(\mathrm{Ax}4b)$, $(\mathrm{Ax}6)$ to decompose the assumption. Steps (7)-(12) act on the goal formula G. Step (7) essentially corresponds to an application of *reductio ad absurdum*. Note that this case only applies when G is an atomic or a disjunctive formula; how we discuss later (see Point (P6) below), the side condition $\mathbf{eval}(G, \Gamma) \neq \mathtt{f}$ has an essential role to guarantee termination. Steps (8)-(12) exploit axioms $(\mathrm{Ax}3)$, $(\mathrm{Ax}5a)$ and the closing assumption schemas of Lemma 2 to decompose the goal formula. Steps (13) and (14) handle the case of negated assumptions. We note that in Step (13) the treated assumption $\neg(B_0 \vee B_1)$ is retained in the recursive call; also in this case the firing condition involving the evaluation function plays an essential role to guarantee termination (see Point (P8) below). Finally, Step (15) handles the case of implicative assumptions.

Termination and completeness

To search for a derivation of a goal G_0 from assumptions Γ_0, we have to compute $\mathbf{Hp}(\Gamma_0, G_0)$. We show that the call $\mathbf{Hp}(\Gamma_0, G_0)$ terminates. The stack of recursive calls involved in the computation of $\mathbf{Hp}(\Gamma_0, G_0)$ can be represented by a chain of the form $\Gamma_0 \vdash G_0 \mapsto \Gamma_1 \vdash G_1 \mapsto \cdots \mapsto \Gamma_n \vdash G_n$ where, for every $k \geq 0$, Γ_k is a set of formulas and the goal G_k is a formula or $\Box$. Each sequent $\sigma_k = \Gamma_k \vdash G_k$ in the chain represents a call $\mathbf{Hp}(\Gamma_k, G_k)$ performed along the computation of $\mathbf{Hp}(\Gamma_0, G_0)$. By $\mathcal{G}_k$ we denote the set $\Gamma_k \cup \{\neg G_k\}$ if $G_k \neq \Box$ and the set Γ_k if $G_k = \Box$.

We have to prove that, for every set of formulas Γ_0 and goal G_0, every chain starting from $\sigma_0 = \Gamma_0 \vdash G_0$ is finite (namely, the chain contains finitely many sequents). Let us assume, by absurd, that there exists an infinite chain $\mathcal{C}(\sigma_0) = \sigma_0 \mapsto \sigma_1 \mapsto \sigma_2 \ldots$; we show that we get a contradiction. To this aim, we state some properties of $\mathcal{C}(\sigma_0)$ (the related proofs are only sketched).

(P1) For every $k \geq 0$, G_k is not an axiom.
(P2) For every $k \geq 0$, the set $\mathcal{G}_k$ is not contradictory.

Let us assume, by contradiction, that there exists $k \geq 0$ such that G_k is an axiom. Then the call $\mathbf{Hp}(\Gamma_k, G_k)$ immediately ends at step (1), hence σ_k would be the last sequent of $\mathcal{C}(\sigma_0)$, against the assumption that $\mathcal{C}(\sigma_0)$ is infinite. This proves (P1).

To prove (P2), let us assume that there exists $k \geq 0$ such that $\mathcal{G}_k$ is contradictory. Then, there is a formula X such that $\{X, \neg X\} \subseteq \mathcal{G}_k$. If $\{X, \neg X\} \subseteq \Gamma_k$ or $G_k = \Box$, then the call $\mathbf{Hp}(\Gamma_k, G_k)$ immediately ends at step (2), a contradiction. Thus, let us assume $G_k \neq \Box$ and either $G_k \in \Gamma_k$ or $\neg\neg G_k \in \Gamma_k$. In the first case, the call $\mathbf{Hp}(\Gamma_k, G_k)$ immediately ends at step (1). If $\neg\neg G_k \in \Gamma_k$, by inspecting $\mathbf{Hp}$ one can check that there exists $j > k$ such that $G_k \in \Gamma_j$ and $G_j = G_k$ (see step (3)), hence σ_j would be the last sequent of $\mathcal{C}(\sigma_0)$, a contradiction. This concludes the proof of (P2).

By (P2) and the properties of $\mathbf{eval}$, we can prove that the evaluation of a formula is preserved along the chain, as stated in the next property:

(P3) Let $\mathbf{eval}(A, \mathcal{G}_k) = \tau$, with $\tau \in \{\mathtt{t}, \mathtt{f}\}$. Then, for every $j \geq k$, $\mathbf{eval}(A, \mathcal{G}_j) = \tau$.

To prove (P3), we can exploit Lemma 4, since, by Point (P2), all the sets $\mathcal{G}_k$ are non-contradictory.

By $\mathrm{Sf}^{\neg}(\sigma_0)$ we denote the set of the formulas H and $\neg H$ such that H is a subformula of a formula in σ_0. By inspecting the definition of $\mathbf{Hp}$, one can easily check that:

(P4) For every $k \geq 0$, $\mathcal{G}_k \subseteq \mathrm{Sf}^{\neg}(\sigma_0)$.
(P5) For every $i \leq j$, $\mathcal{G}_i \cap \mathcal{L}^{\neg} \subseteq \mathcal{G}_j \cap \mathcal{L}^{\neg}$.

We now state some crucial properties of disjunctive goals:

(P6) Let $G_k = A \vee B$ and $\mathbf{eval}(\Gamma_k, A) \neq \mathtt{f}$. Then, there exists $j > k$ such that $\mathbf{eval}(\Gamma_j, A) = \mathtt{f}$.
(P7) Let $G_k = A \vee B$ and $\mathbf{eval}(\Gamma_k, A) = \mathtt{f}$ and $\mathbf{eval}(\Gamma_k, B) \neq \mathtt{f}$. Then, there exists $j > k$ such that $\mathbf{eval}(\Gamma_j, A) = \mathbf{eval}(\Gamma_j, B) = \mathtt{f}$.
(P8) Let $H = \neg(A \vee B)$ such that $H \in \Gamma_k$ and $\mathbf{eval}(H, \Gamma_k \setminus \{H\}) \neq \mathtt{t}$. Then, there exists $j > k$ such that $H \in \Gamma_j$ and $\mathbf{eval}(H, \Gamma_j \setminus \{H\}) = \mathtt{t}$.

Let us consider Point (P6). If $G_k = A \vee B$ and $\mathbf{eval}(\Gamma_k, A) \neq \mathtt{f}$, then there exists $l \geq k$ such that chain starting from the sequent σ_l has the form

$$\Gamma_l \vdash A \vee B \mapsto \Gamma_{l+1} \vdash A \mapsto \cdots ; \mapsto \Gamma_{l+m} \vdash H$$

where the formula H satisfies one of the following properties:

(a) $H = H_0 \vee H_1$ and $\mathbf{eval}(H_0 \vee H_1, \Gamma_{l+m}) \neq \mathtt{f}$;
(b) H is a literal.

Indeed, after a (possibly empty) initial phase where some formulas in Γ_k are handled, the goal formula $A \vee B$ is eagerly decomposed until a disjunctive formula H satisfying (a) or a literal H is obtained. In this phase, whenever we get a goal $G_j = B \to C$, the formula B is added to the left-hand side and the next sequent of the chain is $\Gamma_j, B \vdash C$ (see step (9)); this might introduce a sequence of steps to decompose the formula B and its subformulas. By exploiting (P3), one can prove that:

(c) $\mathbf{eval}(H, \Gamma_{l+m+1}) = \mathtt{f}$ implies $\mathbf{eval}(A, \Gamma_{l+m+1}) = \mathtt{f}$.

If H satisfies (a), the next sequent of the chain is $\sigma_{l+m+1} = \Gamma_{l+m}, \neg H \vdash \Box$ (see step (7)). Since $\mathbf{eval}(H, \Gamma_{l+m+1}) = \mathtt{f}$, by (c) we get $\mathbf{eval}(A, \Gamma_{l+m+1}) = \mathtt{f}$, hence Point (P6) holds. Similarly, if H is a propositional variable, then $H \notin \Gamma_{l+m}$, otherwise the set $\mathcal{G}_{l+m}$ would be contradictory, against Point (P2). Thus, the next sequent of the chain is $\sigma_{l+m+1} = \Gamma_{l+m}, \neg H \vdash \Box$ (see step (7)), and this proves Point (P6). The case $H = \neg p$, with $p \in \mathrm{Pv}$, is similar (see step (10)). The proofs of points (P7) and (P8) are similar.

By the above points, it follows that we eventually get a sequent σ_m such that Γ_m is reduced and $G_m = \Box$. Indeed, by (P4), the formulas occurring in $\mathcal{C}(\sigma_0)$ are subformulas of $\mathrm{Sf}^{\neg}(\sigma_0)$. Along the chain, formulas H occurring in Γ_k such that $H \notin \mathcal{L}^{\neg}$ are eventually decomposed, formulas $H \in \mathcal{L}^{\neg}$ are preserved (see (P5)). Formulas of the kind $\neg p$, with $p \in \mathrm{Pv}$, do not threaten termination. Let us take a formula $H = \neg(A \vee B)$ occurring in some Γ_k. By (P8), there exists $l > k$ such that $\sigma_l = \Gamma_l \vdash G_l$ and $\mathbf{eval}(H, \Gamma_l \setminus \{H\}) = \mathtt{t}$. By (P3), for every $j \geq l$ it holds that $\mathbf{eval}(H, \Gamma_j \setminus \{H\}) = \mathtt{t}$. Since the formulas of the kind $\neg(A \vee B)$ that can occur in the sets Γ_k are finitely many, it follows that we eventually get a sequent σ_m such that Γ_m is reduced and $G_m = \Box$. Since the only step applicable to the call $\mathbf{Hp}(\Gamma_m, G_m)$ is (16), σ_m should be the last sequent of $\mathcal{C}(\sigma_0)$, a contradiction. This proves that $\mathcal{C}(\sigma_0)$ is finite, thus:

Theorem 3 (Termination): Let Γ be a finite set of formulas of $\mathcal{L}$ and G a formula of $\mathcal{L}$ or $\Box$. Then, $\mathbf{Hp}(\Gamma, G)$ terminates.

Now, we prove that $\mathbf{Hp}$ is correct, namely:

Theorem 4 (Correctness): Let Γ be a finite set of formulas of $\mathcal{L}$ and G a formula of $\mathcal{L}$ or $\Box$. Then, $\mathbf{Hp}(\Gamma, G)$ satisfies properties (H1) and (H2).

Proof: By induction on the length l of $\mathcal{C}(\Gamma \vdash G)$. If $l = 0$, then the computation of $\mathbf{Hp}(\Gamma, G)$ does not require any recursive call. Accordingly, one of the steps (1), (2) or (16) is executed. In the first two cases, a derivation satisfying properties (H1) and (H2) is returned. In the latter case, the model $\mathcal{M} = \Gamma \cap \mathrm{Pv}$ is returned. Since none of the conditions in the if-statements of cases (1)–(15) holds, the set Γ is reduced and $G = \Box$; by Theorem 2, we conclude $\mathcal{M} \models \Gamma$. Let $l > 1$ and let $\mathcal{C}(\Gamma \vdash G) = \Gamma \vdash G \mapsto \Gamma_1 \vdash G_1 \mapsto \cdots$. Since the length of $\mathcal{C}(\Gamma_1 \vdash G_1)$ is $l - 1$, by induction hypothesis the call $\mathbf{Hp}(\Gamma_1, G_1)$ satisfies properties (H1) and (H2). By a case analysis, one can easily check that $\mathbf{Hp}(\Gamma, G)$ satisfies (H1) and (H2) as well. For instance, let us assume that $G = A_0 \vee A_1$ and that step (11) is executed. If $\mathbf{eval}(A_0, \Gamma) = \mathtt{f}$, the recursive call $\mathbf{Hp}(\Gamma, A_0)$ is executed. If a deduction $\mathcal{D}$ is returned, by induction hypothesis $\mathcal{D}$ is a deduction of $\Gamma \vdash A_0$. This implies that the deduction $\mathcal{D} \circ \langle A_0 \to (A_0 \vee A_1), A_0 \vee A_1 \rangle$ returned by $\mathbf{Hp}(\Gamma, A_0 \vee A_1)$ is a deduction of $\Gamma \vdash A_0 \vee A_1$ (note that $A_0 \to (A_0 \vee A_1)$ is an instance of the axiom $(\mathrm{Ax}5a)$). Let us assume that $\mathbf{Hp}(\Gamma, A_0)$ returns a model $\mathcal{M}$. By induction hypothesis, we have $\mathcal{M} \models \Gamma \cup \{\neg A_0\}$. Moreover, since the if-condition in step (7) is false, it holds that $\mathbf{eval}(A_0 \vee A_1, \Gamma) = \mathtt{f}$. By Theorem 1, we get $\mathcal{M} \models \neg(A_0 \vee A_1)$, hence $\mathcal{M} \models \Gamma \cup \{\neg(A_0 \vee A_1\}$. The proof of the remaining subcases is similar. ■

To conclude this section we provide two examples, the first one showing an example of an execution of the proof-search

procedure returning a counter model, the second one returning an $\mathbf{Hc}$-derivation.

Example 2: Let us consider the case where the goal formula is $p \vee q$ and the set of the assumptions is empty. We describe the chain of recursive calls applied to compute the goal, referring to the steps in the definition of $\mathbf{Hp}$.

(c1) $\mathbf{Hp}(\emptyset, p \vee q)$
Since $\mathbf{eval}(p \vee q, \emptyset) = p \vee q$*, case (7) is applied and the recursive call* $\mathbf{Hp}(\{\neg(p \vee q)\}, \Box)$ *is executed.*

(c2) $\mathbf{Hp}(\{\neg(p \vee q)\}, \Box)$
Since $\mathbf{eval}(\neg(p \vee q), \emptyset) = \neg(p \vee q)$*, case (13) is selected and* $\mathbf{Hp}(\{\neg(p \vee q)\}, p \vee q)$ *is executed.*

(c3) $\mathbf{Hp}(\{\neg(p \vee q)\}, p \vee q)$
Since $\mathbf{eval}(p \vee q, \{\neg(p \vee q)\}) = \mathtt{f}$*, case (11) is selected. Moreover, since* $\mathbf{eval}(p, \{\neg(p \vee q)\}) = p$*, the call* $\mathbf{Hp}(\{\neg(p \vee q)\}, p)$ *is executed.*

(c4) $\mathbf{Hp}(\{\neg(p \vee q)\}, p)$
Since $\mathbf{eval}(p, \{\neg(p \vee q)\}) = p$*, case (7) is selected and* $\mathbf{Hp}(\{\neg(p \vee q), \neg p\}, \Box)$ *is executed.*

(c5) $\mathbf{Hp}(\{\neg(p \vee q), \neg p\}, \Box)$
Since $\mathbf{eval}(\neg(p \vee q), \{\neg p\}) = \neg q$*, case (13) is selected and* $\mathbf{Hp}(\{\neg(p \vee q), \neg p\}, p \vee q)$ *is executed.*

(c6) $\mathbf{Hp}(\{\neg(p \vee q), \neg p\}, p \vee q)$
Since $\mathbf{eval}(p \vee q, \{\neg(p \vee q), \neg p\}) = \mathtt{f}$*, case (11) is selected. Moreover, since* $\mathbf{eval}(p, \{\neg(p \vee q), \neg p\}) = \mathtt{f}$ *and* $\mathbf{eval}(q, \{\neg(p \vee q), \neg p\}) = q$*,* $\mathbf{Hp}(\{\neg(p \vee q), \neg p\}, q)$ *is executed.*

(c7) $\mathbf{Hp}(\{\neg(p \vee q), \neg p\}, q)$
Since $\mathbf{eval}(q, \{\neg(p \vee q), \neg p\}) = q$*, case (7) is selected and* $\mathbf{Hp}(\{\neg(p \vee q), \neg p, \neg q\}, \Box)$ *is executed.*

(c8) $\mathbf{Hp}(\{\neg(p \vee q), \neg p, \neg q\}, \Box)$
Since $\mathbf{eval}(\neg(p \vee q), \{\neg p, \neg q\}) = \mathtt{t}$ *and* $\neg p$ *and* $\neg q$ *are literals, case (16) is executed and the model* $\emptyset$ *is returned.*

Since the recursive call in (c8) returns the model $\emptyset$, any of the recursive calls in points (c7)-(c1) returns the same model, which indeed is a model of $\{\neg(p \vee q)\}$.

Example 3: Let us consider the case where the goal formula is $p \vee \neg p$ and the set of assumptions is empty.

(c1) $\mathbf{Hp}(\emptyset, p \vee \neg p)$
Since $\mathbf{eval}(p \vee \neg p, \emptyset) = p \vee \neg p$*, case (7) is selected and the recursive call* $\mathbf{Hp}(\{\neg(p \vee \neg p)\}, \Box)$ *is executed.*

(c2) $\mathbf{Hp}(\{\neg(p \vee \neg p)\}, \Box)$
Since $\mathbf{eval}(\neg(p \vee \neg p), \emptyset) = \neg(p \vee \neg p)$*, case (13) is selected and* $\mathbf{Hp}(\{\neg(p \vee \neg p)\}, p \vee \neg p)$ *is executed.*

(c3) $\mathbf{Hp}(\{\neg(p \vee \neg p)\}, p \vee \neg p)$
Since $\mathbf{eval}(p \vee \neg p, \{\neg(p \vee \neg p)\}) = \mathtt{f}$*, case (11) is selected and since* $\mathbf{eval}(p, \{\neg(p \vee \neg p)\}) = p$*, the call* $\mathbf{Hp}(\{\neg(p \vee \neg p)\}, p)$ *is executed.*

(c4) $\mathbf{Hp}(\{\neg(p \vee \neg p)\}, p)$
Since $\mathbf{eval}(p, \{\neg(p \vee \neg p)\}) = p$ *case (7) is selected and* $\mathbf{Hp}(\{\neg(p \vee \neg p), \neg p\}, \Box)$ *is executed.*

(c5) $\mathbf{Hp}(\{\neg(p \vee \neg p), \neg p\}, \Box)$
Since $\mathbf{eval}(\neg(p \vee \neg p), \{\neg p\}) = \mathtt{f}$*, case (13) is selected and* $\mathbf{Hp}(\{\neg(p \vee \neg p), \neg p\}, p \vee \neg p)$ *is executed.*

(c6) $\mathbf{Hp}(\{\neg(p \vee \neg p), \neg p\}, p \vee \neg p)$
Since $\mathbf{eval}(p \vee \neg p, \{\neg(p \vee \neg p), \neg p\}) = \mathtt{t}$*, case (11) is selected. Moreover, we have* $\mathbf{eval}(p, \{\neg(p \vee \neg p), \neg p\}) = \mathtt{f}$ *and* $\mathbf{eval}(\neg p, \{\neg(p \vee \neg p), \neg p\}) = \mathtt{t}$*; as a consequence* $\mathbf{Hp}(\{\neg(p \vee \neg p), \neg p\}, \neg p)$ *is executed.*

(c7) $\mathbf{Hp}(\{\neg(p \vee \neg p), \neg p\}, \neg p)$
Since $\neg p \in \{\neg(p \vee \neg p), \neg p\}$ *case (1) is selected and the* $\mathbf{Hc}$*-derivation* $\langle \neg p \rangle$ *is returned.*

Since the recursive call in (c7) returns a derivation, any of the calls in points (c6)-(c1) returns the derivation built according with the receipt described in the corresponding case.

V. Conclusions

In this paper, we have presented a deterministic and terminating proof-search procedure $\mathbf{Hp}$ for a Hilbert calculus for classical propositional logic (a Prolog implementation is available[10]). $\mathbf{Hp}$ is a procedure that builds Hilbert proofs, if any, in one-pass, that is during the proof-search phase. Such proofs can be seen as derivations based on a sequent calculus having at most one formula on the right. The machinery of evaluations is used to get completeness and termination.
As regards future works, we plan to provide terminating procedures returning Hilbert proofs for intermediate logics by applying the results in [9][11]. We also aim to extend the proof-search procedure for $\mathbf{Hc}$ to treat some modal logics, as, e.g., $S4$ and $S5$ [12] and conditional logics [13].

References

[1] A. Troelstra and H. Schwichtenberg, Basic Proof Theory, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2000, vol. 43.

[2] M. D'Agostino, "Classical natural deduction," in We Will Show Them!, S. A. et al., Ed. College Publications, 2005, pp. 429–468.

[3] W. Sieg and J. Byrnes, "Normal natural deduction proofs (in classical logic)," Studia Logica, vol. 60, no. 1, 1998, pp. 67–106.

[4] M. Ferrari and C. Fiorentini, "Proof-search in natural deduction calculus for classical propositional logic," in TABLEAUX 2015, ser. LNCS, H. D. Nivelle, Ed., vol. 9323. Springer, 2015, pp. 237–252.

[5] ——, "Goal-oriented proof-search in natural deduction for intuitionistic propositional logic," Journal of Automated Reasoning, vol. 62, no. 1, Jan 2019, pp. 127–167. [Online]. Available: https://doi.org/10.1007/s10817-017-9427-3

[6] S. Kleene, Introduction to Metamathematics. Van Nostrand, 1952.

[7] E. Eder, "Automated Deduction in Frege-Hilbert Calculi," in 8th WEAS International Conference on Applied Computer and Applied Computational Science, Chen, SY and Li, Q, Ed. World Scientific and Engineering Acad and Soc, 2009, pp. 21–25.

[8] J. Toriz, I. Ruiz, and J. R. Arrazola-Ramírez, "On automatic theorem proving with ML," in 13th Mexican International Conference on Artificial Intelligence, MICAI 2014, 2014, pp. 231–236.

[9] M. Ferrari, C. Fiorentini, and G. Fiorino, "An evaluation-driven decision procedure for G3i," ACM Transactions on Computational Logic (TOCL), vol. 6, no. 1, 2015, pp. 8:1–8:37.

[10] "Implementation of **Hp** ", 2019, URL: https://drive.google.com/open?id=1nGvqKx_Rj0y2IscL4jldH2jPMWzDB-0X.

[11] A. Avellone, P. Miglioli, U. Moscato, and M. Ornaghi, "Generalized tableau systems for intermediate propositional logics," in TABLEAUX 1997, ser. LNAI, D. Galmiche, Ed., vol. 1227. Springer-Verlag, 1997, pp. 43–61.

[12] A. Chagrov and M. Zakharyaschev, Modal Logic. Oxford University Press, 1997.

[13] D. Lewis, Counterfactuals. John Wiley & Sons, 2013.

 ISBN: 978-1-61208-709-2