A New Approach to Clausification for Intuitionistic Propositional Logic

Camillo Fiorentini[1], Mauro Ferrari[2]

[1]Dep. of Computer Science, Università degli Studi di Milano, Italy
[2]Dep. of Theoretical and Applied Sciences, Università degli Studi dell'Insubria, Italy

Abstract

In recent years some papers have addressed the problem of the validity in Intuitionistic Propositional Logic and in some intermediate propositional logics using the approach proposed by Claessen and Rosén of reduction to Satisfiability Modulo Theories (SMT). This approach depends on an initial pre-processing phase that reduces the input formula in the intuitionistic language to an equivalent sequent in the language of clauses. In this work we present an extension of the clauses used by Claessen and Rosén that allows us to define a natural relationship between the semantics of the extended clauses and Kripke semantics. As an application, we show how Answer Set Programming can be used to check the intuitionistic validity of a formula and to generate Kripke countermodels for countersatisfiable formulas.

Keywords

Intuitionistic Propositional Logic, countermodels construction, Answer Set Programming.

1. Introduction

In [1] Claessen and Rosén have introduced `intuit`, a decision procedure for Intuitionistic Propositional Logic (IPL) that exploits advanced techniques in automated theorem proving; actually, the decision problem is formalized as a Satisfiability Modulo Theories (SMT) search procedure, where the bulk of the computation is performed by an incremental SAT-solver. This approach requires a pre-processing phase to reduce the input formula to a semantically equivalent sequent $\Delta \Rightarrow g$, where Δ is a set of clauses and g an atom. To capture IPL semantics, two kinds of clauses are used: *flat clauses* and *implication clauses*. The former have the form $\bigwedge A_1 \to \bigvee A_2$, where A_1 and A_2 are sets of atoms and are understood as classical clauses. The latter have the form $(a \to b) \to c$, where a, b, c are atoms and $\to$ is the intuitionistic implication. In the SMT approach, implication clauses are part of the theory of the SMT search procedure. The sequent $\sigma = \Delta \Rightarrow g$ obtained by clausifying an input formula α enjoys some relevant properties: first of all, α is valid in IPL if and only if the sequent σ is provable in IPL. Secondly, clausification preserves countermodels: if $\mathcal{K}$ is a countermodel for σ (namely, $\mathcal{K}$ is a Kripke model such that at its root all the formulas in Δ are forced and g is not forced), then $\mathcal{K}$ is a countermodel for α as well (α is not forced at the root of $\mathcal{K}$).

The proof-theoretical counterpart of the above approach has been studied in [2] where it is

CILC'23: 38th Italian Conference on Computational Logic, June 21–23, 2023, Udine, Italy
fiorentini@di.unimi.it (C. Fiorentini); mauro.ferrari@uninsubria.it (M. Ferrari)
0000-0003-2152-7488 (C. Fiorentini); 0000-0002-7904-1125 (M. Ferrari)

shown that there is a close connection between the `intuit` approach and the known proof-theoretic methods. Actually, the `intuit` decision procedure mimics a standard root-first proof search strategy for $\mathrm{LJT}_{\mathtt{SAT}}$, a variant of Dyckhoff's calculus LJT [3] (alias G4ip). In [2], the `intuit` search procedure is rephrased so that, given a sequent σ, a formal derivation of σ in $\mathrm{LJT}_{\mathtt{SAT}}$ or a countermodel for σ is returned.

The `intuit` approach has been refined in [4] where the decision procedure has been re-designed; the obtained prover is called `intuitR` (`intuit` with Restart). Differently from `intuit`, the `intuitR` procedure has a simple structure, consisting of two nested loops and its performances improve those of `intuit`. Finally, in [5, 6] the `intuit` approach is extended to some intermediate and non-classical logics.

Despite these significant improvements of the original `intuit` procedure, many aspects have not been fully investigated. Undoubtedly, a central aspect in the definition of `intuit` is related to the form of clauses, specifically the role of implication clauses. At first glance, it is not clear why such clauses are necessary and one may wonder if other clausal forms are possible, in order to get a clearer and more explicit semantic mapping, as well as more efficient clausification procedures. Moreover, in a more general sense, it is worth exploring whether there exist clausal forms that can be applied to other logics with Kripke-style semantics. For this reason, in this paper we introduce a different approach to clausification based on two phases.

In the first phase we translate a formula α in a clausal form for intuitionistic logic based on the notion of IPL-*clause*. The IPL-clausification procedure translates a formula α in an intuitionistically equivalent set of IPL-clauses. This is the only phase which introduces new propositional variables and hence the role of the new propositional variables can be studied in a purely intuitionistic setting.

In the second step we translate IPL-clauses in the classical setting using the notion of general clause. *General clauses* extend classical ones by introducing literals of the form $a \not\to b$, where a and b are atoms and $\not\to$ is a new operator; intuitively, $a \not\to b$ corresponds to the intuitionistic negation of $a \to b$. A general clause is a disjunction of general literals, where a general literal is either a classical literal of the form a or $\sim a$ or a literal of the form $a \not\to b$; note that general clauses not containing $\not\to$-literals are classical clauses. For general clauses we can define a semantics, we call *realizability*, based on sets of classical interpretations with minimum, which admit a natural translation in Kripke models. Such a semantics is the bridge to apply standard techniques for testing classical satisfiability to check IPL validity and to generate countermodels for non-valid formulas. Even if in this paper we only consider the case of intuitionistic logic, in principle the above approach can be also applied to other logics with a Kripke semantics, provided that we can define a clausal form for the logic and the relationship between the semantics of general clauses and the semantics of the logic at hand.

Here, as an application of our approach, following the ideas of [7], we tackle the problem of intuitionistic validity using a strategy based on model generation formalized in the Answer Set Programming (ASP) setting [8, 9]. To check the intuitionistic validity of a formula α, we apply the clausification procedure to generate a sequent $\sigma = \Delta \Rightarrow g$, where Δ is a set of general clauses and g is an atom, which is equivalent to α. Then we define an ASP program Π_σ such that an answer set of Π_σ corresponds to a countermodel for σ (and hence a countermodel for α); if no answer set for Π_σ exists, there is no countermodel for σ and this implies that α is valid in IPL. The implementation, based on the Potassco tool Clingo [10], is available

at https://github.com/cfiorentini/clausificationIPL.

2. Basic Definitions

Formulas, denoted by lowercase Greek letters, are built from an enumerable set of propositional variables $\mathcal{V}$, the constants $\top$, $\bot$ and the connectives $\wedge$, $\vee$, $\rightarrow$; moreover, $\neg\alpha$ (intuitionistic negation) stands for $\alpha \rightarrow \bot$ and $\alpha \leftrightarrow \beta$ stands for $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$. We refer to the above language as the intuitionistic language to stress the fact that this is the language for which we provide an intuitionistic interpretation over Kripke models. In Sec. 4 we introduce general clauses, an extension of classical clauses; to define them, we use the classical connectives $\sim$, $|$ and the non-classical binary connective $\not\rightarrow$. Elements of the set $\mathcal{V} \cup \{\bot, \top\}$ are called *atoms* and are denoted by lowercase Roman letters, uppercase Greek letters denote sets of formulas. By $\mathcal{V}_\alpha$ we denote the set of propositional variables occurring in α. The notation is extended to sets of formulas: $\mathcal{V}_\Omega$ is the union of $\mathcal{V}_\alpha$ such that $\alpha \in \Omega$; $\mathcal{V}_{\Omega,\Omega'}$ and $\mathcal{V}_{\Omega,\alpha}$ stand for $\mathcal{V}_{\Omega\cup\Omega'}$ and $\mathcal{V}_{\Omega\cup\{\alpha\}}$ respectively.

A *(classical) interpretation* M is a subset of $\mathcal{V}$, identifying the propositional variables assigned to true. By $M \models \alpha$ we mean that α is true in M. Let Ω be a set of formulas; by $M \models \Omega$ we mean that $M \models \alpha$ for every $\alpha \in \Omega$. Classical Propositional Logic (CPL) is the set of formulas true in every interpretation.

A (rooted) Kripke model [11] is a quadruple $\langle W, \leq, r, \vartheta\rangle$ where W is a finite and non-empty set (the set of *worlds*), $\leq$ is a reflexive and transitive binary relation over W, the world r (the *root* of $\mathcal{K}$) is the minimum of W w.r.t. $\leq$, and ϑ (the *valuation* function) is a map from W to $2^{\mathcal{V}}$ obeying the persistence condition: for every pair of worlds w_1 and w_2 of $\mathcal{K}$, $w_1 \leq w_2$ implies $\vartheta(w_1) \subseteq \vartheta(w_2)$. The valuation ϑ is extended to a *forcing* relation between worlds of $\mathcal{K}$ and formulas as follows:

$\mathcal{K}, w \Vdash p$ iff $p \in \vartheta(w), \forall p \in \mathcal{V}$ $\quad$ $\mathcal{K}, w \Vdash \top$ $\quad$ $\mathcal{K}, w \nVdash \bot$

$\mathcal{K}, w \Vdash \alpha \wedge \beta$ iff $\mathcal{K}, w \Vdash \alpha$ and $\mathcal{K}, w \Vdash \beta$ $\quad$ $\mathcal{K}, w \Vdash \alpha \vee \beta$ iff $\mathcal{K}, w \Vdash \alpha$ or $\mathcal{K}, w \Vdash \beta$

$\mathcal{K}, w \Vdash \alpha \rightarrow \beta$ iff $\forall w' \geq w, \mathcal{K}, w' \Vdash \alpha$ implies $\mathcal{K}, w' \Vdash \beta$.

Note that, since $\neg\alpha$ stands for $\alpha \rightarrow \bot$, we have $\mathcal{K}, w \Vdash \neg\alpha$ iff $\forall w' \geq w, \mathcal{K}, w' \nVdash \alpha$. Given a set of formulas Ω, by $\mathcal{K}, w \Vdash \Omega$ we mean that $\mathcal{K}, w \Vdash \alpha$ for every $\alpha \in \Omega$. A formula α is *valid* in the model $\mathcal{K} = \langle W, \leq, r, \vartheta\rangle$ iff $\mathcal{K}, r \Vdash \alpha$. Propositional Intuitionistic Logic (IPL) is the set of formulas valid in all Kripke models. Accordingly, if there is a model $\mathcal{K}$ such that $\mathcal{K}, r \nVdash \alpha$ (where r is the root of $\mathcal{K}$), then α is not IPL-valid; we call $\mathcal{K}$ a *countermodel* for α.

The intuitionistic consequence relation, denoted by $\models_{\mathrm{i}}$, is defined as follows (Ω is a set of formulas, α is a formula):

- $\Omega \models_{\mathrm{i}} \alpha$ iff, for every model $\mathcal{K} = \langle W, \leq, r, \vartheta\rangle$, if $\mathcal{K}, r \Vdash \Omega$ then $\mathcal{K}, r \Vdash \alpha$.

Note that α is IPL-valid iff $\models_{\mathrm{i}} \alpha$ (namely, $\emptyset \models_{\mathrm{i}} \alpha$). The next lemma shows that the decision problem $\models_{\mathrm{i}} \alpha$ can be reduced to the problem $\alpha \rightarrow g \models_{\mathrm{i}} g$, with g a new propositional variable.

Lemma 1 *Let α be a formula and $g \notin \mathcal{V}_\alpha$. Then, $\models_{\mathrm{i}} \alpha$ iff $\alpha \rightarrow g \models_{\mathrm{i}} g$.*

$$\top \wedge \alpha \mapsto \alpha \qquad \alpha \wedge \top \mapsto \alpha \qquad \bot \wedge \alpha \mapsto \bot \qquad \alpha \wedge \bot \mapsto \bot$$
$$\top \vee \alpha \mapsto \top \qquad \alpha \vee \top \mapsto \top \qquad \bot \vee \alpha \mapsto \alpha \qquad \alpha \vee \bot \mapsto \alpha$$
$$\top \to \alpha \mapsto \alpha \qquad \alpha \to \top \mapsto \top \qquad \bot \to \alpha \mapsto \top$$

Figure 1: Boolean reductions.

Proof. The proof that $\models_{\mathrm{i}} \alpha$ implies $\alpha \to g \models_{\mathrm{i}} g$ is immediate. Conversely, let us assume $\not\models_{\mathrm{i}} \alpha$. Then, there exists a Kripke model $\mathcal{K} = \langle W, \leq, r, \vartheta \rangle$ such that $\mathcal{K}, r \nVdash \alpha$. We define the model $\mathcal{K}'$ obtained from $\mathcal{K}$ by modifying the valuation ϑ so that g in $\mathcal{K}'$ simulates the forcing of α in $\mathcal{K}$. Formally, $\mathcal{K}' = \langle W, \leq, r, \vartheta' \rangle$ where: $\vartheta'(w) = \vartheta(w) \cup \{g\}$ if $\mathcal{K}, w \Vdash \alpha$; $\vartheta'(w) = \vartheta(w) \setminus \{g\}$ if $\mathcal{K}, w \nVdash \alpha$. One can easily check that, for every world w in W and every formula β, the following property holds: (1) if $g \notin \mathcal{V}_\beta$, then $\mathcal{K}', w \Vdash \beta$ iff $\mathcal{K}, w \Vdash \beta$. We show that $\mathcal{K}', r \Vdash \alpha \to g$. Let $w \in W$ be such that $\mathcal{K}', w \Vdash \alpha$. By (1), it holds that $\mathcal{K}, w \Vdash \alpha$; by definition of ϑ' we get $\mathcal{K}', w \Vdash g$; this shows that $\mathcal{K}', r \Vdash \alpha \to g$. Since $\mathcal{K}', r \Vdash \alpha \to g$ and $\mathcal{K}', r \nVdash g$ (indeed, $\mathcal{K}, r \nVdash \alpha$), the model $\mathcal{K}'$ ascertains that $\alpha \to g \not\models_{\mathrm{i}} g$. □

To streamline the presentation, we only consider simplified formulas. Formally, a formula α is *simplified* if none of the boolean reductions in Fig. 1 can be applied to subformulas of α. We stress that, given a formula β, by repeatedly applying such reductions, we eventually get a simplified formula α; moreover, since each rewriting step preserves intuitionistic validity, it holds that $\models_{\mathrm{i}} \alpha \leftrightarrow \beta$.

3. Clauses for IPL

We introduce IPL-clauses, a clausal form for intuitionistic formulas, and a procedure `ClauIPL` to turn a simplified formula into an equivalent set of IPL-clauses.

An IPL-clause η is defined by the following grammar:

$$\begin{aligned} \eta \quad ::= \quad & \bot \mid d_1 \vee \cdots \vee d_l \mid \\ & \mid (c_1 \wedge \cdots \wedge c_m \wedge (a_1 \to b_1) \wedge \cdots \wedge (a_n \to b_n)) \to \bot \\ & \mid (c_1 \wedge \cdots \wedge c_m \wedge (a_1 \to b_1) \wedge \cdots \wedge (a_n \to b_n)) \to (d_1 \vee \cdots \vee d_l) \\ & l \geq 1,\ m+n \geq 1 \qquad a_i \in \mathcal{V},\ b_j \in \mathcal{V} \cup \{\bot\},\ c_k \in \mathcal{V},\ d_x \in \mathcal{V} \end{aligned}$$

We call *non-classical* the IPL-clauses containing at least two occurrences of $\to$. We remark that IPL-clauses are a generalizations of the clauses introduced in [1]; indeed, in [1] only non-classical clauses of the form $(a \to b) \to c$ are admitted. To clausify a formula, we exploit the clausification procedure `ClauIPL` defined in Fig. 2 and discussed below. Given a simplified formula α, `ClauIPL` computes a (possibly empty) set of IPL-clauses Θ which is equivalent to α in the sense stated by the following theorem:

Theorem 2 *Let α be a formula and let $\Omega \cup \{\beta\}$ be a set of formulas such that $\mathcal{V}_{\Omega,\beta} \subseteq \mathcal{V}_\alpha$. Then, $\Omega, \alpha \models_{\mathrm{i}} \beta$ iff $\Omega, \mathtt{ClauIPL}(\alpha) \models_{\mathrm{i}} \beta$.*

Since $\alpha \models_{\mathrm{i}} \alpha$, as an immediate consequence of Th. 2 we get:

Proposition 3 $\texttt{ClauIPL}(\alpha) \models_{\mathrm{i}} \alpha$.

Clausification allows us to reduce the validity problem $\models_{\mathrm{i}} \alpha$, where α is a simplified formula, to the problem $\Theta \models_{\mathrm{i}} g$, where Θ is a finite set of IPL-clauses and g a propositional variable. Indeed, let α be any simplified formula and let g be a new propositional variable (namely, $g \notin \mathcal{V}_\alpha$). By Lemma 1, $\models_{\mathrm{i}} \alpha$ if and only if $\alpha \to g \models_{\mathrm{i}} g$. Let $\Theta = \texttt{ClauIPL}(\alpha \to g)$; by Th. 2, $\alpha \to g \models_{\mathrm{i}} g$ iff $\Theta \models_{\mathrm{i}} g$. Thus:

Theorem 4 *Let α be a simplified formula and $g \in \mathcal{V}$ such that $g \notin \mathcal{V}_\alpha$. Then, $\models_{\mathrm{i}} \alpha$ iff $\texttt{ClauIPL}(\alpha \to g) \models_{\mathrm{i}} g$.*

IPL-**clausification** The clausification procedure `ClauIPL` is defined in Fig. 2, by a case analysis on the input formula α; we stress that α is assumed to be simplified, and this reduces the number of cases to consider. We write that $\alpha \equiv \alpha'$ iff α' can be obtained from α by permuting the order of conjuncts or disjuncts in subformulas of α of the kind $\beta_1 \wedge \beta_2$ and $\beta_1 \vee \beta_2$; for instance $p_0 \to (p_1 \wedge (p_2 \vee p_3) \wedge p_4) \equiv p_0 \to ((p_3 \vee p_2) \wedge p_4 \wedge p_1)$.

In the clausification process, we introduce new propositional variables to represent some of the subformulas of α; we write $\tilde{p}_\delta$ to denote the propositional variable associated with the subformula δ of α. If $\alpha = \top$, then Θ is the empty set; if α is an IPL-clause, then Θ is the set $\{\alpha\}$. Otherwise, $\texttt{ClauIPL}(\alpha)$ is defined by distinguishing some cases. One can easily check that, since α is simplified, the list of cases is exhaustive; moreover, the cases do not overlap. In the computation of $\texttt{ClauIPL}(\alpha)$, we have to define the values of $\tilde{p}_\delta$. Given δ, whenever $\tilde{p}_\delta$ is mentioned and $\tilde{p}_\delta$ has not been defined yet, a new propositional variable q must be chosen (namely, q does not occur in δ and in any of the formulas processed so far) and we set $\tilde{p}_\delta = q$.

Example 1 Let α be the simplified formula $\neg\neg(a \vee \neg a)$. Note that α has the form $(\alpha_1 \to \bot) \to \bot$, where $\alpha_1 = a \vee \neg a$. We get:

$$\begin{aligned}
\texttt{ClauIPL}(\alpha) &= \{\, (\tilde{p}_{\alpha_1} \to \bot) \to \bot \,\} \cup \texttt{ClauIPL}(\tilde{p}_{\alpha_1} \to \alpha_1) \\
\texttt{ClauIPL}(\tilde{p}_{\alpha_1} \to \alpha_1) &= \{\, \tilde{p}_{\alpha_1} \to (a \vee \tilde{p}_{\neg a}) \,\} \cup \texttt{ClauIPL}(\tilde{p}_{\neg a} \to \neg a) \\
\texttt{ClauIPL}(\tilde{p}_{\neg a} \to \neg a) &= \texttt{ClauIPL}((\tilde{p}_{\neg a} \wedge a) \to \bot) \;=\; \{\, (\tilde{p}_{\neg a} \wedge a) \to \bot \,\} \\
\texttt{ClauIPL}(\alpha) &= \{\, (\tilde{p}_{\alpha_1} \to \bot) \to \bot,\ \tilde{p}_{\alpha_1} \to (a \vee \tilde{p}_{\neg a}),\ (\tilde{p}_{\neg a} \wedge a) \to \bot \,\}
\end{aligned}$$

We point out that $\tilde{p}_{\alpha_1}$ can be any propositional variable different from a and $\tilde{p}_{\neg a}$ must be distinct from a and $\tilde{p}_{\alpha_1}$. ◊

Now we show that `ClauIPL` terminates. The *size of a formula* α, denoted by $|\alpha|$, and of *weight of a logical operator* $\odot$, denoted by $\mathrm{wg}(\odot)$ ($\odot \in \{\wedge, \vee, \to\}$) are defined as follows:

$$|\alpha| = \begin{cases} 0 & \text{if } \alpha \in \mathcal{V} \cup \{\bot, \top\} \\ |\alpha_1| + |\alpha_2| + \mathrm{wg}(\odot) & \text{if } \alpha = \alpha_1 \odot \alpha_2 \end{cases} \qquad \begin{aligned} \mathrm{wg}(\wedge) &= 1 \\ \mathrm{wg}(\vee) &= 3 \\ \mathrm{wg}(\to) &= 2 \end{aligned}$$

The weights of logical operators have been chosen so to guarantee the following property:

α	Θ
$\top$	$\emptyset$
α is an IPL-clause	$\{\alpha\}$
$\alpha_1 \wedge \alpha_2$	$\texttt{ClauIPL}(\alpha_1) \cup \texttt{ClauIPL}(\alpha_2)$
$\alpha \equiv \eta_2 \vee \beta_1 \vee \cdots \vee \beta_y$ $y \geq 1$	$\{\, \eta_2 \vee \tilde{p}_{\beta_1} \vee \cdots \vee \tilde{p}_{\beta_y} \,\} \cup \Theta_1 \cup \cdots \cup \Theta_y$ $\Theta_j = \texttt{ClauIPL}(\tilde{p}_{\beta_j} \rightarrow \beta_j), j \in \{1, \ldots, y\}$
$\alpha_1 \rightarrow (\beta_1 \wedge \beta_2)$	$\texttt{ClauIPL}(\alpha_1 \rightarrow \beta_1) \cup \texttt{ClauIPL}(\alpha_1 \rightarrow \beta_2)$
$\alpha_1 \rightarrow (\beta_1 \rightarrow \beta_2)$	$\texttt{ClauIPL}((\alpha_1 \wedge \beta_1) \rightarrow \beta_2)$
$(\alpha_1 \vee \alpha_2) \rightarrow \beta$	$\texttt{ClauIPL}(\alpha_1 \rightarrow \beta) \cup \texttt{ClauIPL}(\alpha_2 \rightarrow \beta)$
$(\alpha_1 \rightarrow \alpha_2) \rightarrow \beta$	$\{\, (\tilde{\alpha}_1 \rightarrow \tilde{\alpha}_2) \rightarrow \tilde{\beta} \,\} \cup \Theta_1 \cup \Theta_2 \cup \Theta_3$ $\tilde{\alpha}_1 = \begin{cases} \alpha_1 & \text{if } \alpha_1 \in \mathcal{V} \cup \{\bot\} \\ \tilde{p}_{\alpha_1} & \text{otherwise} \end{cases}$ $\tilde{\alpha}_2 = \begin{cases} \alpha_2 & \text{if } \alpha_2 \in \mathcal{V} \cup \{\bot\} \\ \tilde{p}_{\alpha_2} & \text{otherwise} \end{cases}$ $\tilde{\beta} = \begin{cases} \beta & \text{if } \beta \in \mathcal{V} \cup \{\bot\} \\ \tilde{p}_{\beta} & \text{otherwise} \end{cases}$ $\Theta_1 = \begin{cases} \emptyset & \text{if } \alpha_1 \in \mathcal{V} \cup \{\bot\} \\ \texttt{ClauIPL}(\tilde{p}_{\alpha_1} \rightarrow \alpha_1) & \text{otherwise} \end{cases}$ $\Theta_2 = \begin{cases} \emptyset & \text{if } \alpha_2 \in \mathcal{V} \cup \{\bot\} \\ \texttt{ClauIPL}(\alpha_2 \rightarrow \tilde{p}_{\alpha_2}) & \text{otherwise} \end{cases}$ $\Theta_3 = \begin{cases} \emptyset & \text{if } \beta \in \mathcal{V} \cup \{\bot\} \\ \texttt{ClauIPL}(\tilde{p}_{\beta} \rightarrow \beta) & \text{otherwise} \end{cases}$
$\alpha \equiv (\eta_1 \wedge \alpha_1 \wedge \cdots \wedge \alpha_x)$ $\rightarrow (\eta_2 \vee \beta_1 \vee \cdots \vee \beta_y)$ $x + y \geq 1$	$\{\, (\, \eta_1 \wedge \tilde{p}_{\alpha_1} \wedge \cdots \wedge \tilde{p}_{\alpha_x} \,) \rightarrow (\, \eta_2 \vee \tilde{p}_{\beta_1} \vee \cdots \vee \tilde{p}_{\beta_y} \,) \,\}$ $\cup \Theta'_1 \cup \cdots \cup \Theta'_x \cup \Theta''_1 \cup \cdots \cup \Theta''_y$ $\Theta'_i = \texttt{ClauIPL}(\alpha_i \rightarrow \tilde{p}_{\alpha_i}), i \in \{1, \ldots, x\}$ $\Theta''_j = \texttt{ClauIPL}(\tilde{p}_{\beta_j} \rightarrow \beta_j), j \in \{1, \ldots, y\}$

$\eta_1 = c_1 \wedge \cdots \wedge c_m \wedge (a_1 \rightarrow b_1) \wedge \cdots \wedge (a_n \rightarrow b_n) \qquad \eta_2 = d_1 \vee \cdots \vee d_l$

$\alpha_1, \ldots, \alpha_x, \beta_1, \ldots, \beta_y$ are non-atomic formulas

Figure 2: Definition of the recursive procedure $\texttt{ClauIPL}(\alpha)$, where α is simplified.

.

Lemma 5 *Let α be a simplified formula and let $\texttt{ClauIPL}(\alpha')$ be any of the recursive calls invoked by $\texttt{ClauIPL}(\alpha)$ (see Fig. 2). Then, $|\alpha'| < |\alpha|$.*

Proof. The assertion can be easily proved by a case analysis. As an example, let $\alpha \equiv (\eta_1 \wedge \alpha_1 \wedge \cdots \wedge \alpha_x) \rightarrow (\eta_2 \vee \beta_1 \vee \cdots \vee \beta_y)$ (last case in Fig. 2), and let $\texttt{ClauIPL}(\alpha')$ be any of the recursive calls in the definition of $\texttt{ClauIPL}(\alpha)$. We have two possible cases:

(a) $\alpha' = \alpha_i \to \tilde{p}_{\alpha_i}$, where $i \in \{1, \ldots, x\}$; (b) $\alpha' = \tilde{p}_{\beta_j} \to \beta_j$, where $j \in \{1, \ldots, y\}$.

In Case (a), the formula α contains at least one of the displayed occurrences of $\wedge$, thus $|\alpha| \geq |\alpha_i| + 1 + 2$ (the latter is the weight of the main operator $\to$ of α). On the other hand $|\alpha'| = |\alpha_i| + 2$, and this proves that $|\alpha'| < |\alpha|$. The discussion of Case (b) is similar. □

By Lemma 5, it immediately follows that `ClauIPL` is terminating. Finally, we remark that the procedure `ClauIPL` satisfies Th. 2 (see the proof in the Appendix available at https://github.com/cfiorentini/clausificationIPL).

4. General Clauses

We introduce general clauses, an extension of classical clauses that can be used to capture non-classical logics, such as intermediate and modal logics. Moreover, we exhibit a one-to-one translation of IPL-clauses into general clauses.

Classical clauses are built over propositional variables, using the the operators $\sim$(classical negation) and $|$ (classical disjunction). A *classical literal* is a formula of the kind a or $\sim a$, where a is a propositional variable. A *classical clause* γ is a set of classical literals $\{l_1, \ldots, l_n\}$, denoted by $l_1 \mid \ldots \mid l_n$; if $n = 0$, then γ is the empty clause, denoted by $\Box$. We point out that the order of the literals occurring in a clause is immaterial; for instance, $a \mid \sim b$ and $\sim b \mid a$ denote the same clause. Given two classical clauses $\gamma = l_1 \mid \ldots \mid l_m$ and $\gamma' = l'_1 \mid \ldots \mid l'_n$, where $m \geq 0$ and $n \geq 0$, by $\gamma \mid \gamma'$ we denote the classical clause $l_1 \mid \ldots \mid l_m \mid l'_1 \mid \ldots \mid l'_n$. Given a classical interpretation M and a classical clause γ, the relation $M \models \gamma$ is defined as follows:

$$M \not\models \Box \quad M \models a \text{ iff } a \in M \quad M \models \sim a \text{ iff } a \notin M \quad M \models l_1 \mid \ldots \mid l_n \text{ iff } \exists k : M \models l_k.$$

We extend classical clauses by introducing the binary operator $\not\to$, which is a sort of negation of intuitionistic implication. A *general literal* is either a classical literal or a $\not\to$-formula of the kind $a \not\to b$, where $a \in \mathcal{V}$ and $b \in \mathcal{V} \cup \{\bot\}$. A *general clause* is a set of general literals $\{l_1, \ldots, l_m\}$, denoted by $l_1 \mid \ldots \mid l_m$; we stress that the order of the literals is immaterial. We call *non-classical* a general clause containing at least one occurrence of $\not\to$.

Semantics of general clauses. We introduce a semantics for general clauses, we call *realizability semantics*, based on sets of classical interpretations, that plays a crucial role to establish the relationship between general clauses and IPL-clauses.

Let W be a nonempty set of classical interpretations; we define the realizability relations $\rhd^0_W$ and $\rhd_W$ between the interpretations in W and general clauses. Given an interpretation M in W and a general clause $\delta = \gamma \mid a_1 \not\to b_1 \mid \ldots \mid a_n \not\to b_n$, where γ is a classical clause and $n \geq 0$, the relations $M \rhd^0_W \delta$ and $M \rhd_W \delta$ are defined as follows:

- $M \rhd^0_W \delta$ iff $M \models \gamma$, or there is $M' \in W$ and $i \in \{1, \ldots, n\}$ such that $M \subseteq M'$ and $M' \models a_i$ and $M' \not\models b_i$.
- $M \rhd_W \delta$ iff, for every $M' \in W$, if $M \subseteq M'$ then $M' \rhd^0_W \delta$.

$$\Psi(\eta) \;=\; \begin{cases} \Box & \text{if } \eta = \bot \\ d_1 \mid \ldots \mid d_l & \text{if } \eta = d_1 \vee \cdots \vee d_l \\ \sim c_1 \mid \ldots \mid \sim c_m \mid a_1 \not\to b_1 \mid \ldots & \text{if } \eta = \; (\, c_1 \wedge \cdots \wedge c_m \wedge (a_1 \to b_1) \wedge \ldots \\ \qquad \ldots \mid a_n \not\to b_n & \qquad\qquad \cdots \wedge (a_n \to b_n)\,) \;\to\; \bot \\ d_1 \mid \ldots \mid d_l \mid \sim c_1 \mid \ldots & \text{if } \eta = \; (\, c_1 \wedge \cdots \wedge c_m \wedge (a_1 \to b_1) \wedge \ldots \\ \qquad \ldots \mid \sim c_m \mid a_1 \not\to b_1 \mid \ldots \mid a_n \not\to b_n & \qquad\qquad \cdots \wedge (a_n \to b_n)\,) \;\to\; (d_1 \vee \cdots \vee d_l) \end{cases}$$

$$\delta \;=\; d_1 \mid \ldots \mid d_l \mid \sim c_1 \mid \ldots \mid \sim c_m \mid a_1 \not\to b_1 \mid \ldots \mid a_n \not\to b_n$$

$$\Phi(\delta) \;=\; \begin{cases} \bot & \text{if } \delta = \Box \; (l = m = n = 0) \\ \neg(c_1 \wedge \cdots \wedge c_m) & \text{if } l = n = 0 \text{ and } m > 0 \\ d_1 \vee \cdots \vee d_l & \text{if } l > 0 \text{ and } m = n = 0 \\ (\, c_1 \wedge \cdots \wedge c_m \,) \;\to\; (d_1 \vee \cdots \vee d_l) & \text{if } l > 0, m > 0 \text{ and } n = 0 \\ \neg(c_1 \wedge \cdots \wedge c_m \wedge (a_1 \to b_1) \wedge \cdots \wedge (a_n \to b_n)) & \text{if } l = 0, m > 0 \text{ and } n > 0 \\ (\, c_1 \wedge \cdots \wedge c_m \wedge (a_1 \to b_1) \wedge \cdots \wedge (a_n \to b_n)\,) \;\to\; (d_1 \vee \cdots \vee d_l) & \text{otherwise} \end{cases}$$

Figure 3: Translations between IPL-clauses and general clauses.

In general $\triangleright^0_W$ is not monotone w.r.t. the inclusion relation $\subseteq$. For instance, let $W = \{\, \emptyset,\, \{a\}\,\}$; then $\emptyset \triangleright^0_W \sim a$ and $\emptyset \subseteq \{a\}$, but $\{a\} \not\triangleright^0_W \sim a$. Instead, $\triangleright_W$ is monotone by definition. Given a set of general clauses Δ, by $M \triangleright^0_W \Delta$ (resp., $M \triangleright_W \Delta$) we mean $M \triangleright^0_W \delta$ (resp., $M \triangleright_W \delta$) for every δ in Δ.

Let W be a finite set of classical interpretations; the *minimum* of W is the minimum element r of W with respect to the subset relation $\subseteq$ (namely, $r \subseteq M$ for every $M \in W$). Let W be a finite set of interpretations with minimum r. By $\mathcal{K}(W)$ we denote the structure $\langle W, \leq, r, \vartheta \rangle$ where: $\leq$ coincides with the subset relation $\subseteq$; ϑ is the identity map. It is easy to check that $\mathcal{K}(W)$ is a Kripke model such that $\mathcal{K}(W), w \Vdash p$ iff $p \in w$.

From IPL-clauses to general clauses. In Fig. 3 we display the bijective map Ψ between IPL-clauses and general clauses and its inverse Φ; note that non-classical IPL-clauses are mapped to non-classical general clauses, and vice versa. We investigate the relationship between the semantics of IPL-clauses (Kripke models) and the semantics of general clauses (realizability). The next proposition shows that the forcing relation on $\mathcal{K}(W)$ mirrors the realizability relation $\triangleright_W$.

Proposition 6 *Let δ be a general clause, let W be a set of classical interpretations having minimum element. For every w in W, $w \triangleright_W \delta$ iff $\mathcal{K}(W), w \Vdash \Phi(\delta)$.*

Proof. We only consider the most general case where:

$$\begin{array}{lcl} \delta & = & (\underbrace{c_1 \wedge \cdots \wedge c_m \wedge (a_1 \rightarrow b_1) \wedge \ldots (a_n \rightarrow b_n)}_{\alpha}) \rightarrow \underbrace{(d_1 \vee \cdots \vee d_l)}_{\beta} \\ \Psi(\delta) & = & \underbrace{d_1 \mid \ldots \mid d_l \mid\!\sim c_1 \mid \ldots \mid\!\sim c_m}_{\gamma} \mid a_1 \not\rightarrow b_1 \mid \ldots \mid a_n \not\rightarrow b_n \\ & & \text{where } l \geq 1,\ m \geq 1,\ n \geq 1 \end{array}$$

Let us assume $w \rhd_W \delta$; we show $w \Vdash \Phi(\delta)$ [1], namely $w \Vdash \alpha \rightarrow \beta$. We proceed by induction on the height of w in $\mathcal{K}(W)$, defined as the length of the longest path from w to a maximal world of $\mathcal{K}(W)$ w.r.t. $\leq$. We start by considering the base case $\mathrm{h}(w) = 0$ namely, w is a maximal world in $\mathcal{K}(W)$ w.r.t. $\leq$; we point out that w is maximal in W w.r.t. $\subseteq$. Let us assume $w \Vdash \alpha$, we prove $w \Vdash \beta$. We have: $w \Vdash c_i$, for every $i \in \{1, \ldots, m\}$ and $w \nVdash a_j$ or $w \Vdash b_j$, for every $j \in \{1, \ldots, n\}$. This implies that: $w \models c_i$, for every $i \in \{1, \ldots, m\}$ and $w \not\models a_j$ or $w \models b_j$, for every $j \in \{1, \ldots, n\}$. Since $w \rhd^0_W \delta$ and w is maximal in W w.r.t. $\subseteq$, there exists $k \in \{1, \ldots, l\}$ such that $w \models d_k$. It follows that $w \Vdash d_k$, which implies $w \Vdash \beta$. This concludes the proof of $w \Vdash \alpha \rightarrow \beta$ in the base case. Now, let $\mathrm{h}(w) > 0$ and let $w' \in W$ such that $w \leq w'$ and $w' \Vdash \alpha$; we show that $w' \Vdash \beta$. If $w < w'$, then $w \subset w'$ and $\mathrm{h}(w') < \mathrm{h}(w)$. Since $\rhd_W$ is monotone, it holds that $w' \rhd_W \delta$; by the induction hypothesis, we get $w' \Vdash \alpha \rightarrow \beta$, and this implies $w' \Vdash \beta$. On the other hand, if $w' = w$ then $w \Vdash \alpha$, hence: $w \Vdash c_i$, for every $i \in \{1, \ldots, m\}$ and $w' \nVdash a_j$ or $w' \Vdash b_j$, for every $w' \in W$ s.t. $w \leq w'$ and every $j \in \{1, \ldots, n\}$. This implies that: $w \models c_i$, for every $i \in \{1, \ldots, m\}$ and $w' \not\models a_j$ or $' \models b_j$, for every $w' \in W$ s.t. $w \leq w'$ and every $j \in \{1, \ldots, n\}$. Since $w \rhd^0_W \delta$, there is $k \in \{1, \ldots, l\}$ such that $w \models d_k$, hence $w \Vdash \beta$. We have proved that $w \Vdash \alpha \rightarrow \beta$, namely $w \Vdash \Phi(\delta)$; accordingly, $w \rhd_W \delta$ implies $w \Vdash \Phi(\delta)$.

To prove the converse, we show that:

(1) If $w \Vdash \Phi(\delta)$, then $w \rhd^0_W \delta$.

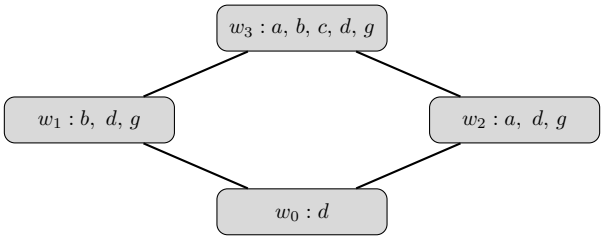
We assume $w \Vdash \Phi(\delta)$, namely $w \Vdash \alpha \rightarrow \beta$, and we show $w \rhd^0_W \delta$. If $w \models \gamma$, we immediately get $w \rhd^0_W \delta$. Let us assume $w \not\models \gamma$; then: $w \models c_i$, for every $i \in \{1, \ldots, m\}$ and $w \not\models d_k$, for every $k \in \{1, \ldots, l\}$. This implies that: $w \Vdash c_i$, for every $i \in \{1, \ldots, m\}$ and $w \nVdash d_k$, for every $k \in \{1, \ldots, l\}$. Since $w \Vdash \alpha \rightarrow \beta$ and $w \nVdash \beta$, it holds that $w \nVdash \alpha$; hence there is $j \in \{1, \ldots, n\}$ such that $w \nVdash a_j \rightarrow b_j$. It follows that there is $w' \in W$ such that $w \leq w'$ and $w' \Vdash a_j$ and $w' \nVdash b_j$, hence $w \subseteq w'$ and $w' \models a_j$ and $w' \not\models b_j$. We get $w \rhd^0_W \delta$, and this proves point (1).

Let us assume $w \Vdash \Phi(\delta)$ and let $w' \in W$ such that $w \leq w'$. Since $w' \Vdash \Phi(\delta)$, by (1) we get $w' \rhd^0_W \delta$. This proves that $w \rhd_W \delta$, and this concludes the proof of the proposition. □

Sequents. According with Th. 4, the problem of IPL-validity of a formula α can be reduced to the decidability of $\Theta \mid\!\!=_{\mathrm{i}} g$, where g is a propositional variable and Θ is a set of IPL-clauses; in the rest of the paper, we focus on the problem $\Theta \mid\!\!=_{\mathrm{i}} g$.

We can show that $\Theta \not\mid\!\!=_{\mathrm{i}} g$ by exhibiting a Kripke (counter)model $\mathcal{K} = \langle W, \leq, r, \vartheta \rangle$ such that $\mathcal{K}, r \Vdash \Theta$ and $\mathcal{K}, r \nVdash g$. By Prop. 6, this is equivalent to search for a set of interpretations W having minimum element r such that $r \rhd_W \Psi(\Theta)$ and $r \not\rhd_W \Psi(g)$, namely $g \notin r$. To formalize

[1] Here and below we leave understood the model $\mathcal{K}(W)$.



$$\begin{array}{rcl} \Delta & = & \{\, a \mid\!\sim c,\ \ b \mid\!\sim c,\ \ d \mid g,\ \ g \mid a \not\rightarrow b,\ \ g \mid b \not\rightarrow a,\ \ \sim d \mid c \not\rightarrow \bot \,\} \\ \Phi(\Delta) & = & \{\, c \rightarrow a,\ c \rightarrow b,\ d \vee g,\ (a \rightarrow b) \rightarrow g,\ (b \rightarrow a) \rightarrow g,\ \neg(d \wedge \neg c) \,\} \\ & & w_0 \rhd_W \Delta, \quad \mathcal{K}(W), w_0 \Vdash \Phi(\Delta) \end{array}$$

Figure 4: Countermodel $W = \{w_0, w_1, w_2, w_3\}$ for $\sigma = \Delta \Rightarrow g$ (see Ex. 2).

the latter problem, we introduce the notions of sequent and countermodel. A sequent $\Delta \Rightarrow g$ is a pair where Δ is a set of general clauses and g is an atom. A *countermodel* for $\Delta \Rightarrow g$ is a set of classical interpretations W with minimum r such that $r \rhd_W \Delta$ and $g \notin r$. A sequent σ is *countersatisfiable* iff there exists a countermodel for σ.

Example 2 In Fig. 4 we define a sequent $\sigma = \Delta \Rightarrow g$ and we show a countermodel for σ. The countermodel W consists of the classical interpretations w_0, w_1, w_2, w_3; for each interpretation w_k we list the propositional variables in w_k; thick lines represent the inclusion relation[2]. One can easily check that $w_0 \rhd_W \Delta$ and $g \notin w_0$; accordingly, W is countermodel for σ. It is easy to prove that there are not countermodels for σ having less than four interpretations. The same picture also represents the Kripke model $\mathcal{K}(W) = \langle W, \leq, w_0, \vartheta \rangle$; in this case each w_k must be understood as a world of the model (with the list of variables in $\vartheta(w_k)$) and thick lines denote the relation $\leq$. One can easily check that $w_0 \Vdash \Phi(\Delta)$ and $w_0 \nVdash g$, hence $\Phi(\Delta) \not\models_{\mathrm{i}} g$. $\Diamond$

The crucial property of countersatisfiable sequents is stated in the next proposition:

Proposition 7 *A sequent $\sigma = \Delta \Rightarrow g$ is countersatisfiable iff $\Phi(\Delta) \not\models_{\mathrm{i}} g$.*

Proof. Let σ be countersatisfiable, let W be a countermodel for σ and r the minimum of W. Since $r \rhd_W \Delta$, by Prop. 6 we get $\mathcal{K}(W), r \Vdash \Phi(\Delta)$; since $g \notin r$, it also holds that $\mathcal{K}(W), r \nVdash g$. Accordingly, the Kripke model $\mathcal{K}(W)$ witnesses that $\Phi(\Delta) \not\models_{\mathrm{i}} g$.

Conversely, let us assume $\Phi(\Delta) \not\models_{\mathrm{i}} g$ and let $\mathcal{K}$ be a Kripke model such that $\mathcal{K}, r \Vdash \Phi(\Delta)$ and $\mathcal{K}, r \nVdash g$, with r the root of $\mathcal{K}$. One can easily define a set of interpretations W such that the model $\mathcal{K}$ is isomorphic to the model $\mathcal{K}(W)$; note that the root of $\mathcal{K}$ is mapped to the minimum r of W. By Prop. 6, we get $r \rhd_W \Delta$; since $g \notin r$, W is a countermodel for σ, hence σ is countersatisfiable. $\Box$

[2]Redundant lines are omitted, e.g. self-loops, the line connecting w_0 with w_3.

```
% index of atoms
idxOfAtom(a,0). idxOfAtom(b,1). idxOfAtom(c,2). idxOfAtom(d,3). idxOfAtom(g,4).
%% classical clauses (3)
% c1 = a | ~c
clClause(c1). litOf(a, c1). litOf(neg(c), c1).
% c2 = b | ~c
clClause(c2). litOf(b, c2). litOf(neg(c), c2).
% c3 = d | g
clClause(c3). litOf(d, c3). litOf(g, c3).
%% non-classical clauses (3)
% gc1 = g | a-/->b
genClause(gc1). litOf(g, gc1). litOf(negImp(a, b), gc1).
% gc2 = g | b-/->a
genClause(gc2). litOf(g, gc2). litOf(negImp(b, a), gc2).
% gc3 = ~d | c-/->false
genClause(gc3). litOf(neg(d), gc3). litOf(negImp(c, false), gc3).
rightAtom(g).  % right atom

atomSet( w(0..1, 0..1, 0..1, 0..1, 0..1) ).   % shorthand for:
% atomSet(w(0,0,0,0,0)). atomSet(w(0,0,0,0,1)). ...  atomSet(w(1,1,1,1,1)).
```

Figure 5: The component $\mathtt{I}_\sigma$ corresponding to the sequent σ defined in Ex. 2.

5. ASP Encoding

We describe an ASP program Π_σ that, given a sequent σ, searches for a countermodel for σ. According with the ASP paradigm (see e.g. [12]), Π_σ is a Prolog-like program consisting of two components $\mathtt{I}_\sigma$ and `Gen`. The component $\mathtt{I}_\sigma$ encodes the instance of the problem determined by the input sequent σ; the component `Gen` implements the model generator. A solution to $\Pi_\sigma = \mathtt{Gen} \cup \mathtt{I}_\sigma$, called *answer set*, corresponds to a countermodel for σ; if Π_σ has no answers, then no countermodel for σ exists. We exploit the ASP solver `clingo` [13]. The clause language is encoded as follows: atoms are represented by Clingo constants, with $\bot$ denoted by `false`; atoms representing propositional variables are indexed by natural numbers using the `idxOfAtom`/2 predicate. General literals of the kind $\sim a$ and $a \not\to b$ are encoded using the function symbols `neg`/1 and `negImp`/2 respectively. We have to introduce names for classical clauses and general (non-classical) clauses by using the predicates `clClause`/1 and `genClause`/1. To define the literals belonging to a general clause (either classical or not), we use the predicate `litOf`/2. The right atom of a sequent is specified by the predicate `rightAtom`/1. In Fig. 5 we display the encoding of the sequent σ defined in Fig. 4.

Countermodels are sets of worlds, and each world is represented by a set of propositional variables (atomSet). Let n be the number of distinct propositional variables occurring in σ; an atomSet is represented by a term of the form $\mathtt{w}(b_0, ..., b_{n-1})$, where each argument b_k is a boolean value, namely $b_k = 0$ or $b_k = 1$. Let a_k be the atom having index k, as set by the predicate `idxOfAtom`/2; the term $\mathtt{w}(b_0, ..., b_{n-1})$ represents the set of a_k such that $b_k = 1$. For

instance, the encoding in Fig. 5 yields the following representation:

$\mathtt{w}(0,0,0,0,0) \mapsto \emptyset \qquad \mathtt{w}(0,0,0,0,1) \mapsto \{g\} \qquad \ldots \qquad \mathtt{w}(1,0,1,0,1) \mapsto \{a,c,g\} \quad \ldots$

The predicate `atomSet/1` enumerates all the atomSet's; since the arity of `w` depends on the input sequent σ, the definition of `atomSet/1` (as well as the definition of other auxiliary predicates) must be supplied by $\mathtt{I}_\sigma$ (see Fig. 5).

The countermodel generation algorithm is encoded in the component `Gen`. The classical validity of a literal in an atomSet (seen as a classical interpretation) is computed by the predicate `satisfies_lit/2` defined by the following clauses (the auxiliary predicate `atom/1` specifies the defined atoms, `member/2` implements the membership relation on atomSet's):

```
satisfies_lit(W,Atm) :- atomSet(W), atom(Atm), member(Atm,W).
satisfies_lit(W,neg(Atm)) :- atomSet(W), atom(Atm), not member(Atm,W).
```

The predicate `satisfies_clClause/2` checks whether an atomSet W satisfies a classical clause C, namely: W satisfies all the literals in C; such a condition is expressed by a cardinality constraint built over the aggregate `#count`. We also introduce the auxiliary predicate `satisfies_all_clClauses/1` to check whether an atomSet satisfies all the classical clauses.

```
satisfies_clClause(W,C) :-  atomSet(W),clClause(C),
    #count{ Lit : litOf(Lit,C), satisfies_lit(W,Lit) } > 0.
satisfies_all_clClauses(W) :-  atomSet(W),
    #count{ C : clClause(C), not satisfies_clClause(W,C) } = 0.
```

The generator has to *guess* a set a worlds, where a world is an atomSet satisfying all the classical clauses, and to *check* if the set of selected worlds is a countermodel for the input sequent σ. Worlds are denoted by the predicate `world/1` and are generated by the following rule, which selects zero or more worlds out of the atomSet's satisfying all the classical clauses:

```
{ world(W) : atomSet(W), satisfies_all_clClauses(W) }.
```

The set of selected worlds is a candidate solution, which must be checked by introducing suitable constraints. The predicate `realizes_genLit/2` encodes the realizability relation $\rhd^0$ between worlds and general clauses; below we only display the non-trivial case of the definition (`leq/2` encodes the subset relations between worlds).

```
realizes_genLit( W, negImp(Atm1,Atm2) ) :- world(W), atom(Atm1), atom(Atm2),
  #count{ W1 : world(W1), leq(W,W1), member(Atm1,W1) , not member(Atm2, W1) } > 0.
```

The predicate `realizes_genClause/2` encodes the realizability relation between worlds and general clauses, and is similar to `satisfies_clClause/2`. To rule out solutions where worlds do no satisfy all the general clauses, we introduce the following constraint:

```
:- world(W), #count{ C : genClause(C), not realizes_genClause(W,C) } > 0.
% it is false that W is a world and the number of C s.t. W does not realize C is > 0
```

We force that exactly one world, denoted by `root/1`, does not contain the right atom of the input sequent; we also require that the root world is the minimum w.r.t. subset relation (this rules out dead worlds).

```
:- rightAtom(G) , #count{ W : world(W), not member(G,W) } != 1.
% it is false that G is the rightAtom and the number of W non containing G is != 1
root(W) :- world(W),  rightAtom(G), not member(G,W).
:- root(W0),  world(W1),  not leq(W0,W1).
```

Let us consider the program $\Pi_\sigma = \texttt{Gen} \cup \texttt{I}_\sigma$. If $\mathcal{S}$ is a solution to Π_σ (answer set), then the set of worlds in $\mathcal{S}$ is a countermodel for σ. Conversely, if Π_σ has no solution, then there is no countermodel for σ. We remark that we can ask `clingo` to search for a minimal countermodel with respect to the number of worlds, by exploiting the `#minimize` feature:

```
numberOfWorlds(N) :-  #count{ W : world(W) } = N.  % N is the number of worlds
#minimize { N : numberOfWorlds(N) }.
```

In our implementation, the source file `generator.lp` encodes the component `Gen`, while the `#minimize` statement is set apart in the file `minimize.lp`. Assuming that the component $\texttt{I}_\sigma$ is encoded by `sigma.lp`, we can issue the commands :

```
clingo generator.lp sigma.lp              % (1) search for a solution
clingo generator.lp minimize.lp sigma.lp  % (2) search for a minimal solution
```

Example 3 Let `sigma.lp` be the instance $\texttt{I}_\sigma$ in Fig. 5. The command (2) yields:

```
world(w(0,0,0,1,0)) world(w(0,1,0,1,1)) world(w(0,0,0,1,1)) world(w(1,1,1,1,1))
memberOfWorld(d,w(0,0,0,1,0))  memberOfWorld(b,w(0,1,0,1,1)) ....
Optimization: 4
OPTIMUM FOUND
```

The computed answer set defines the worlds

$$w_0 = \texttt{w}(0,0,0,1,0) \quad w_1 = \texttt{w}(0,1,0,1,1) \quad w_2 = \texttt{w}(1,0,0,1,1) \quad w_3 = \texttt{w}(1,1,1,1,1)$$

The solution corresponds to the countermodel in Fig. 4. ◊

We can use Π_σ to decide IPL-validity. Indeed, let α be a simplified formula and let σ be the sequent $\Psi(\Theta) \Rightarrow g$, where $g \notin \mathcal{V}_\alpha$ and $\Theta = \texttt{ClauIPL}(\alpha \to g)$. If Π_σ has no answer, then the sequent σ is not countersatisfiable; by Prop. 7 it follows that $\Theta \models_{\mathrm{i}} g$ and, by Th. 4, we conclude $\models_{\mathrm{i}} \alpha$. Conversely, let us assume that Π_σ has a solution and let W be the set of worlds in one of the computed answer set; then W is a countermodel for σ, thus $r \rhd_W \Psi(\Theta)$ and $g \notin r$, where r is the minimum of W. Let us consider the Kripke model $\mathcal{K}(W)$; by Prop. 6 we get $\mathcal{K}(W), r \Vdash \Theta$ and $\mathcal{K}(W), r \nVdash g$. Since $\Theta \models_{\mathrm{i}} \alpha \to g$ (see Prop. 3), it follows that $\mathcal{K}(W), r \Vdash \alpha \to g$, thus $\mathcal{K}(W), r \nVdash \alpha$. We conclude that α is not IPL-valid and $\mathcal{K}(W)$ is a countermodel for α. Note that this approach presents an odd asymmetry: if α is not IPL-valid, we get a concrete certificate of it, namely a countermodel for α; on the contrary, if α is IPL-valid, we gain no evidence of it. We leave as future work the investigation of a more informative procedure.

Example 4 Let α be the formula $(a \to b) \vee (b \to a)$ and let $\Theta = \texttt{ClauIPL}(\alpha \to g)$. We have:

$$\begin{array}{rcl} \Theta & = & \texttt{ClauIPL}((a \to b) \to g) \cup \texttt{ClauIPL}((b \to a) \to g) \;=\; \{\, (a \to b) \to g,\ (b \to a) \to g \,\} \\ \Psi(\Theta) & = & \{\, g \mid a \not\to b,\ g \mid b \not\to a \,\} \end{array}$$

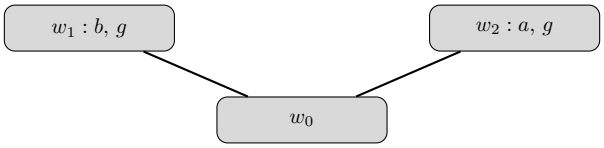


Figure 6: Countermodel $W = \{w_0, w_1, w_2\}$ for $\alpha = (a \to b) \vee (b \to a)$ (see Ex. 4).

Let $\sigma = \Psi(\Theta) \Rightarrow g$ and let σ be encoded by assigning the indexes 0, 1, 2 to a, b, g respectively. The program Π_σ generates a minimum countermodel W for σ consisting of the worlds $w_0 = \mathtt{w}(0,0,0)$, $w_1 = \mathtt{w}(0,1,1)$, $w_2 = \mathtt{w}(1,0,1)$ (see Fig. 6). It is immediate to check that $\mathcal{K}(W)$ is a countermodel for the input formula α. $\Diamond$

The implementation of the procedures presented in the paper are available at https://github.com/cfiorentini/clausificationIPL.

6. Conclusion and Future Work

In this paper we have presented a new approach to clausification for Intuitionistic Propositional Logic which differs from the one used in [1, 2, 4, 5, 6] and it is based on two phases. The first phase provides a clausification in the pure intuitionistic setting and this allows us to clarify the role of the propositional variables introduced by the clausification procedure. The second phase translates the intuitionistic clauses in the classical setting exploiting the notion of general clause. The realizability semantics of general clauses provides a natural bridge with the Kripke semantics of intuitionistic clauses.

As a future work, we aim to implement a prover for IPL exploiting the approach based on SMT along the lines of [1, 2, 4, 5, 6]. Moreover, we aim to apply our approach to other non-classical logics with Kripke semantics such as modal logics.

As for the application we have presented in this paper, we aim to investigate the role that intuitionistic simplifications, such as the ones defined in [14], can play in the reduction of the size of the models generated by the ASP program.

References

[1] K. Claessen, D. Rosén, SAT Modulo Intuitionistic Implications, in: M. Davis, A. Fehnker, A. McIver, A. Voronkov (Eds.), LPAR-20, volume 9450 of *LNCS*, Springer, 2015, pp. 622–637.

[2] C. Fiorentini, R. Goré, S. Graham-Lengrand, A Proof-Theoretic Perspective on SMT-Solving for Intuitionistic Propositional Logic, in: S. Cerrito, A. Popescu (Eds.), TABLEAUX 2019, volume 11714 of *LNCS*, Springer, 2019, pp. 111–129.

[3] R. Dyckhoff, Contraction-free sequent calculi for intuitionistic logic, J. Symb. Log. 57 (1992) 795–807.

[4] C. Fiorentini, Efficient SAT-based proof search in intuitionistic propositional logic, in: A. Platzer, G. Sutcliffe (Eds.), CADE 28, volume 12699 of *LNCS*, Springer, 2021, pp. 217–233.

[5] C. Fiorentini, M. Ferrari, SAT-based proof search in intermediate propositional logics, in: J. Blanchette, L. Kovács, D. Pattinson (Eds.), IJCAR, volume 13385 of *LNCS*, Springer, 2022, pp. 57–74.
[6] R. Goré, C. Kikkert, CEGAR-tableaux: Improved modal satisfiability via modal clause-learning and SAT, in: A. Das, S. Negri (Eds.), TABLEAUX 2021, volume 12842 of *LNCS*, Springer, Cham, 2021, pp. 74–91.
[7] C. Fiorentini, An ASP Approach to Generate Minimal Countermodels in Intuitionistic Propositional Logic, in: S. Kraus (Ed.), IJCAI, ijcai.org, 2019, pp. 1675–1681.
[8] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2010.
[9] E. Dantsin, T. Eiter, G. Gottlob, A. Voronkov, Complexity and expressive power of logic programming, ACM Comput. Surv. 33 (2001) 374–425.
[10] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Answer Set Solving in Practice, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2012.
[11] A. V. Chagrov, M. Zakharyaschev, Modal Logic, volume 35 of *Oxford logic guides*, Oxford University Press, 1997.
[12] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2010.
[13] M. Gebser, R. Kaminsk, B. Kaufmann, T. Schaub, Answer Set Solving in Practice, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers, 2012.
[14] M. Ferrari, C. Fiorentini, G. Fiorino, Simplification Rules for Intuitionistic Propositional Tableaux, ACM Trans. Comput. Log. 13 (2012) 14:1–14:23.