An ASP Approach to Generate Minimal Countermodels in Intuitionistic Propositional Logic

Camillo Fiorentini

DI, Univ. degli Studi di Milano, Milano, Italy

IJCAI 2019

Macao, August 14th, 2019

Intuitionistic Propositional Logic

Intuitionistic Propositional Logic (IPL) is a constructive non-classical logic.

- Non-classical: some classical tautologies are not valid in IPL

$$A \vee \neg A \qquad (A \to B) \vee (B \to A) \qquad \neg A \vee (A \to B)$$

- Constructive: IPL enjoys the Disjunction Property:

$$A \vee B \in \mathrm{IPL} \quad \Longrightarrow \quad A \in \mathrm{IPL} \text{ or } B \in \mathrm{IPL}$$

IPL is closely related to Propositional Classical Logic (CPL):

- $\mathrm{IPL} \subset \mathrm{CPL}$
- IPL can be embedded in CPL:

$$A \in \mathrm{IPL} \quad \Longrightarrow \quad \neg\neg A \in \mathrm{IPL}$$

Thus, the following principles are valid in IPL:

$$\neg\neg(A \vee \neg A) \qquad \neg\neg((A \to B) \vee (B \to A)) \qquad \neg\neg(\neg A \vee (A \to B))$$

Semantics

- CPL

  An interpretation $\mathcal{I}$ is a set of propositional variables.

  The validity of a formula w.r.t. $\mathcal{I}$ is defined according to the classical meaning of logical connectives (truth tables):

  - $\mathcal{I} \models p$ iff $p \in \mathcal{I}$, for p a propositional variable
  - $\mathcal{I} \models A \wedge B$ iff $\mathcal{I} \models A$ and $\mathcal{I} \models B$
  - ...

  CPL is the set of formulas valid in all the interpretations.

- IPL

  To get a sound semantics for IPL, we need a more refined semantics. A model is a set of classical interpretations, called worlds

  - √ Each world represents a knowledge state
  - √ Worlds are ordered by a partial order relation $\leq$
  - √ Validity is represented by forcing relation $\Vdash$ between worlds and formulas
  - √ Forcing is preserved by $\leq$:

$$w_1 \Vdash A \quad \wedge \quad w_1 \leq w_2 \quad \Longrightarrow \quad w_2 \Vdash A$$

This leads to Kripke frame semantics.

Semantics

A Kripke model is a structure $\mathcal{K} = \langle P, \leq, V \rangle$, where:

- P is a finite nonempty set of worlds
- $\leq$ is a partial order between worlds
- V assign to each world a classical interpretation, obeying truth preservation:

$$w_1 \leq w_2 \implies \mathcal{I}(w_1) \subseteq \mathcal{I}(w_2)$$

- The forcing relation $\Vdash$ between worlds and formulas is inductively defined as follows:
  - $w \nVdash \bot$
  - $w \Vdash p$ iff $V(w) \models p$ ($V(w)$ is the interpretation related to w)
  - $w \Vdash A \wedge B$ iff $w \Vdash A$ and $w \Vdash B$
  - $w \Vdash A \vee B$ iff $\alpha \Vdash A$ or $w \Vdash B$
  - $w \Vdash \neg A$ iff, for every $w' \geq w$, $w' \nVdash A$
  - $w \Vdash A \rightarrow B$ iff, for every $w' \geq w$, $w' \nVdash A$ or $w' \Vdash B$

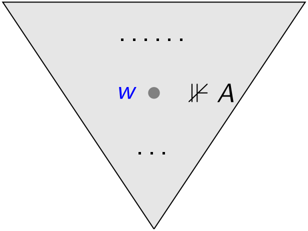
For formulas $\neg A$ and $A \rightarrow B$, forcing at w depends on the successors of w

Semantics

IPL is complete with respect to Kripke semantics, namely:

- $A \in \mathrm{IPL}$ iff A is forced in every world of every Kripke model

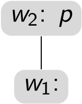
Accordingly, if $A \notin \mathrm{IPL}$, there exists a model $\mathcal{K}$ and a world w in $\mathcal{K}$ such that A is not forced at w.
We call $\mathcal{K}$ a countermodel for A



Semantics

Example

A countermodel for $p \vee \neg p$ is



$$V(w_1) = \emptyset \qquad v(w_2) = \{p\}$$

$w_1 \nVdash p$	since $p \notin V(w_1)$
$w_1 \nVdash \neg p$	since $w_1 \leq w_2$ and $w_2 \Vdash p$ $(p \in V(w_2))$
$w_1 \nVdash p \vee \neg p$	since $w_1 \nVdash p$ and $w_1 \nVdash \neg p$

At w_1, p is not forced.

The world w_1 is followed by a world w_2 and p is forced at w_1, thus $\neg p$ is not forced at w_2. Since forcing must be preserved through $\leq$, $\neg p$ is not forced at w_1.

We conclude that $p \vee \neg p$ is not forced at w_1.

Validity vs. non-validity

Let G be a goal formula

- The validity of G in IPL can be witnessed by a derivation in a sound calculus for IPL
  Hilbert calculus, natural deduction deduction, tableaux/sequent, ...

- The non-validity of G can be witnessed by a countermodel

Typically, the emphasis is on derivations and countermodels are obtained as a result of a failed proof-search for a derivation of G.

For almost all the the known tableaux/sequent calculi for IPL, we can define a proof-search procedure `ProofSearch` such that:

$$\texttt{ProofSearch}(G) = \begin{cases} \text{A derivation of } G & \text{If } G \in \mathrm{IPL} \\ \text{A countermodel for } G & \text{Otherwise} \end{cases}$$

Countermodels

- A countermodel can be understood as a certificate witnessing the non-validity of the goal formula G
- Countermodels can be used for diagnosis, to analyze why some property fails or to fix errors in formal specifications (see Property-Based Testing).
- It is critical that countermodels are minimal so as to convey a plain and concise representation of non-validity.

This issue has been scarcely investigated in the literature. Many proof-seearch procedures have been introduced, but all fail to build small countermodels

L. Pinto and R.Dyckhoff, Loop-free construction of counter-models for intuitionistic propositional logic. Symposia Gaussiana Conf, 1995

G. Corsi and G. Tassi. Intuitionistic logic freed of all metarules. JSL, 2007

M. Ferrari, C. Fiorentini, and G. Fiorino. Contraction-free linear depth sequent calculi for intuitionistic propositional logic with the subformula property and minimal depth counter-models. JAR, 2013.

D. Larchey-Wendling, D. Mry, and D. Galmiche. STRIP: Structural Sharing for Efficient Proof-Search. IJCAR, 2001.

V. Svejdar. On sequent calculi for intuitionistic propositional logic. Comment. Math. Univ. 2006

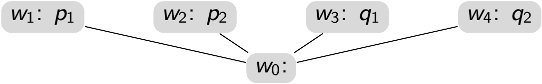
...

Countermodels

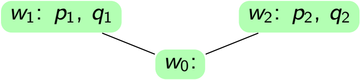
Example

$$G \;=\; (p_1 \to p_2) \vee (p_2 \to p_1) \vee (q_1 \to q_2) \vee (q_2 \to q_1)$$

Countermodel generated by `ProofSearch(G)` [Ferrari et al.,TOCL,2015] generating countermodels of minimal depth



The model has minimal height, but it is not mininal in the number of worlds. A minimum countermodel is:



Note that we cannot shrink the first model to get a minimum one!

Countermodel generation

Main contribution

We present a procedure to generate minimal countermoles:

- given a goal formula G, we try to build a countermodel for G by a model-search procedure guided by semantics.

A naive implementation of the process immediately blows-up; even for small goal formulas, model generation is not terminating.

We need a clever formalization of the problem.

Countermodel generation

- Model formalization

  We follow the approach of R. Goré et al. [IJCAR 2012, 2014]:

  - Worlds of models are represented by sets $\mathcal{W}$ of atomic subformulas H of G, namely:

    $$H ::= p \mid \neg A \mid A \rightarrow B \qquad p\text{: propositional variable}$$

  - We do not considers all possible sets $\mathcal{W}$ of atomic subformulas, but only the sets $\mathcal{W}$ satisfying some closure properties, we call p-worlds (possible worlds)

    For instance:

    $$\mathcal{W}_1 = \{\, p, \neg p \,\} \qquad \mathcal{W}_2 = \{\, p, p \rightarrow q \,\}$$

    $\mathcal{W}_1$ must be discarded since it is inconsistent
    $\mathcal{W}_1$ must be discarded since it is not closed under modus ponens ($q \notin \mathcal{W}_2$)

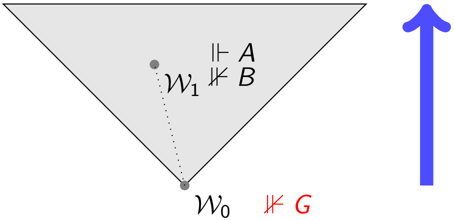
Countermodel generation

- The first selected p-world $\mathcal{W}_0$ is a putative world falsifying G.
- To get a well-defined Kripke model, we have to guarantee that atomic subformulas of G not belonging to $\mathcal{W}_0$ are not valid in $\mathcal{W}_0$, for instance:

$$A \to B \notin \mathcal{W}_0 \quad \Longrightarrow \quad \exists \mathcal{W}_1 (\, \mathcal{W}_0 \subseteq \mathcal{W}_1 \wedge A \in \mathcal{W}_1 \wedge B \notin \mathcal{W}_1 \,)$$

$\mathcal{W}_1$ is needed to witness the non-validity of $A \to B$ in $\mathcal{W}_0$.

This triggers a saturation process which successfully ends when all the needed witnesses have been generated, thus yielding a countermodel for G.



Countermodel generation

- Computation engine

  We formalize the search problem in Answer Set Programming (ASP) [Baral 2010].

  - √ ASP is a form of declarative programming based on the stable model semantics (answer sets),
  - √ ASP enables to solve hard search problems (in NP and in NP^{NP}) in a uniform way

- We define an ASP program Π_G such that an answer set of Π_G corresponds to a countermodel for G.
  If no answer exists, there is no countermodel for G, meaning that G is valid (in IPL).

- To compute answer sets, we exploit the Potassco tool `clingo` [Gebser et al.,2012].

- The minimization of models is delegated to `clingo`; however, it is critical to encode the problem so that even the first computed model is small, otherwise the minimization engine gets stuck.

Countermodel generation

Differently from other declarative formalisms, ASP allows for a quite modular formalization:

$$\Pi_G \;=\; \mathtt{Gen} \;+\; \mathtt{Goal}(G)$$

- `Gen` encodes the genereator and is independent of the goal formula
- $\mathtt{Goal}(G)$ encodes the goal formula

The generator can be easily extended to deal with other intermediate logics where the frame conditions can be expressed in ASP, such as:

- The Gödel-Dummett logic [Dummett,59], characterized by linear frames
- The logic of bound-depth frames
- Here and There logic [Pearce,97], well-known in ASP

Countermodel generation

Frame conditions can be freely composed:

- `lin.lp` encodes the contraint "the model is linear"

```
:-  world(W1), world(W2), W1 <> W2 ,  not le(W1,W2), not le(W2,W1).
```

- `bd2.lp` encodes the contraint "the model has depth at most 2"

```
:-  world(W1), world(W2), world(W3), W1 <> W2, W1 <> W3, W2 <> W3,
    le(W1,W2), le(W2,W3).
```

```
clingo  gen.lp  goal.lp  lin.lp          //  linear countermodels
clingo  gen.lp  goal.lp  bd2.lp     //  detpth <=2 countermodels

clingo  gen.lp  goal.lp  lin.lp  bd2.lp
 //  linear AND  detpth <=2 countermodels
```

This kind of modularity is not possible with derivations!

Countermodel generation

The program is efficiente with formulas containing few propositional variables.
For instance, let us consider the non-valid Nishimura formulas:

$$N_1 = p \qquad N_2 = \neg p$$
$$N_{2n+3} = N_{2n+1} \vee N_{2n+2} \qquad N_{2n+4} = N_{2n+3} \to N_{2n+1}$$

The cuntermodel for N_{17} is computed in few seconds: