Efficient SAT-based Proof Search in Intuitionistic Propositional Logic

Camillo Fiorentini

Department of Computer Science
University of Milan, Italy

CADE 2021

The 28th International Conference on Automated Deduction

https://www.cs.cmu.edu/~mheule/CADE28/

July, 12th, online

Motivations

- In 2015, Claessen and Rosén introduced `intuit`, an efficient decision procedure for IPL (Intuitionistic Propositional Logic) based on a Satisfiability Modulo Theories (SMT) approach.

  K. Claessen and D. Rosén. SAT Modulo Intuitionistic Implications, LPAR 2015

  The `intuit` decision procedure exploits an incremental SAT-solver.

- On the top of `intuit`, we have implemented `intuitR` (`intuit` with Restart), obtaining significant advantages.

`intuit`: specification

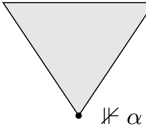
- Input

  A formula α

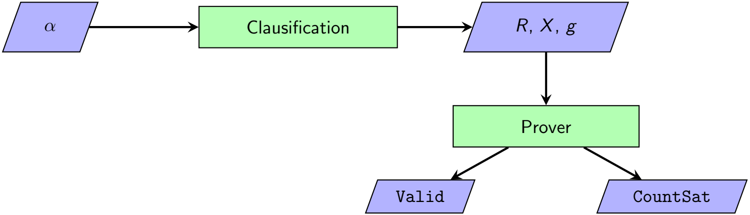
- Output

  - `Valid` if α is intuitionistically valid
  - `CountSat` (counter-satisfiable) if α is not intuitionistically valid

    Thus, there exists a countermodel for α, namely:
    a Kripke model such that at its root α is not forced



intuit: architecture



Two main modules:

- Clausification

  Pre-processing of the input formula α:

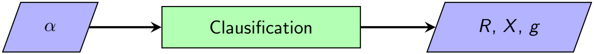
  √ the validity of α is reduced to the validity of a sequents of the kind $R, X \Rightarrow g$, where R, X and g have a simple form.

- Prover

  Decide the validity of $R, X \Rightarrow g$.

  Most of the computation is performed by an incremental SAT-solver.

intuit: clausification



- R is a set of flat clauses φ of the form

$$\varphi \;:=\; \bigwedge A_1 \to \bigvee A_2 \qquad A_1,\, A_2\text{: sets of atoms}$$

Flat clauses are actively used in classical reasoning (SAT-solver)

- X is a set of implication clauses λ of the form

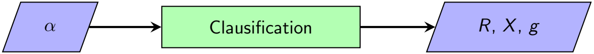
$$\lambda \;:=\; (a \to b) \to c \qquad a,\, b,\, c\text{: atoms}$$

- g is an atom

We also assume that (R, X) is $\to$-closed:

$$(a \to b) \to c \in X \quad \Longrightarrow \quad b \to c \in R$$

intuit: clausification



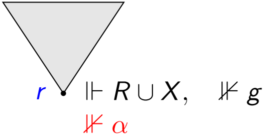
R : set of flat clauses φ
X : set of impl. clauses λ
$\varphi \coloneqq \bigwedge A_1 \to \bigvee A_2$
$\lambda \coloneqq (a \to b) \to c$
a, b, c, g : atoms
$A_1,\ A_2$: sets of atoms

(1) $\alpha \in \mathrm{IPL}$ iff $R, X \vdash_{\mathrm{i}} g$ $\quad\quad$ $\vdash_{\mathrm{i}}$: intuitionistic provability

(2) Let $\mathcal{K}$ be a countermodel for the sequent $R, X \Rightarrow g$, namely:

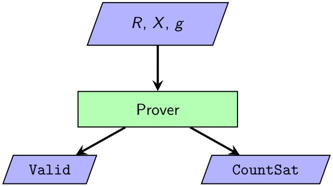
- at the root r of $\mathcal{K}$, all the formulas in R and X are forced and g is not forced.

Then, $\mathcal{K}$ is a countermodel for α.



Clausification is performed by applying standard rewriting steps.

intuit: prover



R : set of flat clauses φ
X : set of impl. clauses λ
$\varphi \;:=\; \bigwedge A_1 \to \bigvee A_2$
$\lambda \;:=\; (a \to b) \to c$
a, b, c, g : atoms
A_1, A_2 : set of atoms

`Valid`	implies $R, X \vdash_{\mathrm{i}} g$
`CountSat`	implies $R, X \nvdash_{\mathrm{i}} g$

- Decision algorithm: a variant of the DPLL($\mathcal{T}$) procedure.
- Most of the computation is performed by an incremental SAT-solver:

  √ Learning mechanism:

  During the computation, flat clauses φ of the form

  $$\varphi := \bigwedge A \to b$$

  (with A a set of atoms) are learned and permanently added to the solver (learned clauses)

intuit

- Implemented in Haskell
- intuit outperforms the best state-of-the-art provers for IPL

But

- YES/NO procedure (no informative output)
- The procedure seems to be far away from the traditional techniques for deciding IPL validity.

Recent work (joint paper with S. Graham-Lengrand and R. Goré):

> *A Proof-Theoretic Perspective on SMT-Solving for Intuitionistic Propositional Logic, Tableaux 2019.*

We unveil a close and surprising connection between the intuit decision procedure based on SMT and the known proof-theoretic methods.

- the decision procedure mimics a standard root-first proof search strategy for the sequent calculus $\mathrm{LJT}_{\mathtt{SAT}}$, a variant of Dyckhoff's calculus LJT

The calculus $\mathrm{LJT}_{\mathtt{SAT}}$

We consider r-sequents (reduced sequents) of the form:

$$R, X \Rightarrow g \qquad \begin{array}{l} R\text{: set of fat clauses } \bigwedge A_1 \to \bigvee A_2 \\ X\text{: set of impl. clauses } (a \to b) \to c \\ g\text{: atom} \end{array}$$

The calculus $\mathrm{LJT}_{\mathtt{SAT}}$ is sound and complete w.r.t. IPL.

$$\vdash_{\mathrm{LJT}_{\mathtt{SAT}}} R, X \Rightarrow g \qquad \text{iff} \qquad R, X \vdash_{\mathrm{i}} g$$

Axiom rule

$$\frac{R \vdash_{\mathrm{c}} g}{R, X \Rightarrow g}\ \mathrm{cpl}_0 \qquad \vdash_{\mathrm{c}}: \text{ classical provability}$$

The soundness follows by this well-known property:

- $R \vdash_{\mathrm{c}} g$ iff $R \vdash_{\mathrm{i}} g$

To check the condition $R \vdash_{\mathrm{c}} g$ we use the SAT-solver.

The calculus $\mathrm{LJT}_{\mathtt{SAT}}$

Rule for left $\to$

Let us consider Dyckhoff rule adapted to r-sequents:

$$\frac{R,\, X,\, b \to c,\, a \Rightarrow b \qquad R,\, X,\, c \Rightarrow g}{R,\, X,\, (a \to b) \to c \Rightarrow g}$$

We are assuming $b \to c \in R$, thus it can be rewritten as

$$\frac{R,\, X,\, a \Rightarrow b \qquad R,\, X,\, c \Rightarrow g}{R,\, X,\, (a \to b) \to c \Rightarrow g}$$

Generalization

- multiplicative contexts (split R into R_1, R_2 and X into X_1, X_2)
- Replace the atom a in the left premise with a set of atoms A

$$\frac{R_1, X_1,\, A \Rightarrow b \qquad R_2, X_2, ??? \Rightarrow g}{R_1, R_2,\, X_1, X_2,\, (a \to b) \to c \Rightarrow g}$$

The calculus $\mathrm{LJT}_{\mathtt{SAT}}$

Rule ljt for left implication:

$$\frac{R_1, X_1, A \Rightarrow b \qquad \varphi, R_2, X_2, \underline{(a \to b) \to c} \Rightarrow g}{R_1, R_2, X_1, X_2, \underline{(a \to b) \to c} \Rightarrow g}$$

A is any set of atoms

$\varphi = \bigwedge(A \setminus \{a\}) \to c$

In the right premise:

- the main formula $(a \to b) \to c$ is kept
- the flat clause φ is added

Intuitively:

- R_1, R_2 are the clauses in the SAT-solver
- φ is the learned clause to be added to the SAT-solver.

Proof of soundness

$$\lambda = (a \to b) \to c$$

$$\dfrac{\dfrac{\dfrac{\dfrac{R_1, X_1, A \vdash_{\mathrm{i}} b}{R_1, X_1, A \setminus \{a\} \vdash_{\mathrm{i}} a \to b}\ R\to \qquad \dfrac{}{\lambda, a \to b \vdash_{\mathrm{i}} c}\ \mathrm{MP}}{R_1, X_1, \lambda, A \setminus \{a\} \vdash_{\mathrm{i}} c}\ \mathrm{cut}}{R_1, X_1, \lambda \vdash_{\mathrm{i}} \varphi}\ R\to \qquad \varphi, R_2, X_2, \lambda \vdash_{\mathrm{i}} g}{R_1, R_2, X_1, X_2, \lambda \vdash_{\mathrm{i}} g}\ \mathrm{cut}$$

The calculus $\mathrm{LJT}_{\mathtt{SAT}}$

$$\frac{R \vdash_{\mathrm{c}} g}{R, X \Rightarrow g}\ \mathrm{cpl}_0$$

$$\frac{R_1,\ X_1,\ A \Rightarrow b \qquad \varphi,\ R_2,\ X_2,\ (a \to b) \to c \Rightarrow g}{R_1, R_2,\ X_1, X_2,\ (a \to b) \to c \Rightarrow g}\ \mathrm{ljt} \qquad \varphi = \bigwedge(A \setminus \{a\}) \to c$$

We also need a cut rule

$$\frac{R_1, X_1 \vdash_{\mathrm{i}} \varphi \qquad \varphi, R_2, X_2 \Rightarrow q}{R_1, R_2, X_1, X_2 \Rightarrow q}\ \mathrm{cut}$$

In [Tableuax 2019], we formalize the `intuit` decision procedure so that, given an r-sequent $\sigma = R, X \Rightarrow g$, it outputs either a derivation of σ in $\mathrm{LJT}_{\mathtt{SAT}}$ or a countermodel for σ.

The end of the story?

Beyond `intuit`

We have enhanced the Haskell `intuit` code by implementing the derivation/countermodel extraction procedures

We experimented some unexpected and weird phenomena:

- derivations are often convoluted and contain applications of the cut rule which cannot be trivially eliminated.
- countermodels have lots of redundancies.

To overcome these issues:

- we introduce the sequent calculus $C^{\to}$, a lightweight variant of $\mathrm{LJT}_{\mathtt{SAT}}$
- we redesign the `intuit` decision procedure, using $C^{\to}$ instead of $\mathrm{LJT}_{\mathtt{SAT}}$

  We call the new prover `intuitR` (`intuit` with Restart)

The calculus $C^{\to}$

The calculus $C^{\to}$ only consists of two rules:

- Axiom rule

  Same axiom rule as in $\mathrm{LJT}_{\mathtt{SAT}}$

$$\frac{R \vdash_{\mathrm{c}} g}{R, X \Rightarrow g}\ \mathrm{cpl}_0$$

- Left implication

  A simplified version of the rule ljt of $\mathrm{LJT}_{\mathtt{SAT}}$ (rule cpl_1 in next slide).

There is no need for cut rule.

The calculus $C^{\to}$: rule for left $\to$

Let us consider the additive variant of rule ljt (A is any set of atoms):

$$\frac{R,\, X,\, A \Rightarrow b \qquad \varphi,\, R,\, X,\, (a \to b) \to c \Rightarrow g}{R,\, X,\, (a \to b) \to c \Rightarrow g} \qquad \varphi \coloneqq \bigwedge(A \backslash \{a\}) \to c$$

We require that the left premise has a trivial proof:

$$\frac{\dfrac{R,\, A \vdash_{\mathrm{c}} b}{R,\, X,\, A \Rightarrow b}\,\mathrm{cpl}_0 \qquad \varphi, R,\, X,\, (a \to b) \to c \Rightarrow g}{R,\, X,\, (a \to b) \to c \Rightarrow g}$$

We get the rule cpl_1:

$$\frac{R,\, A \vdash_{\mathrm{c}} b \qquad \varphi,\, R,\, X,\, (a \to b) \to c \Rightarrow g}{R,\, X,\, (a \to b) \to c \Rightarrow g}\,\mathrm{cpl}_1 \qquad \varphi \coloneqq \bigwedge(A \backslash \{a\}) \to c$$

Very simple rule: one premise, one side condition involving classical provability (thus it can be checked by a SAT-solver)

Derivations of $C^{\to}$ have a plain linear structure (one branch):

$$
\dfrac{R_{m-1},\, A_{m-1} \vdash_{\mathrm{c}} b_{m-1} \qquad \dfrac{R_m \vdash_{\mathrm{c}} g}{R_m,\, X \Rightarrow g}}{R_{m-1},\, X \Rightarrow g}\;\lambda_{m-1}
$$

$$
\vdots
$$

$$
\dfrac{R_0,\, A_0 \vdash_{\mathrm{c}} b_0 \qquad \dfrac{R_1,\, A_1 \vdash_{\mathrm{c}} b_1 \qquad \overbrace{\varphi_1,\, R_1}^{R_2},\, X \Rightarrow g}{\overbrace{\varphi_0,\, R_0}^{R_1},\, X \Rightarrow g}\;\lambda_1}{R_0,\, X \Rightarrow g}\;\lambda_0
$$

$$
\begin{aligned}
\lambda_k &:= (a_k \to b_k) \to c_k \in X \\
\varphi_k &:= \bigwedge(A_k \setminus \{a_k\}) \to c_k \\
R_{k+1} &:= R_k \cup \{\varphi_k\}
\end{aligned}
$$

Rule names are omitted, we display the main formulas λ_k of cpl_1 applications.

The calculus $C^{\rightarrow}$: derivations

$$
\dfrac{R_{m-1},\, A_{m-1} \vdash_{\mathrm{c}} b_{m-1} \qquad \dfrac{R_m \vdash_{\mathrm{c}} g}{R_m,\, X \Rightarrow g}}{R_{m-1},\, X \Rightarrow g}\ \lambda_{m-1}
$$

$$
\vdots
$$

$$
\dfrac{R_0,\, A_0 \vdash_{\mathrm{c}} b_0 \qquad \dfrac{R_1,\, A_1 \vdash_{\mathrm{c}} b_1 \qquad R_2,\, X \Rightarrow g}{R_1,\, X \Rightarrow g}\ \lambda_1}{R_0,\, X \Rightarrow g}\ \lambda_0
$$

Note that the sets R_k are increasing

$$
R_0 \subseteq R_1 \subseteq R_2 \subseteq \cdots \subseteq R_m
$$

Accordingly, to check the conditions

$$
R_0, A_0 \vdash_{\mathrm{c}} b_0, \quad R_1, A_2 \vdash_{\mathrm{c}} b_1, \quad \ldots \quad R_m \vdash_{\mathrm{c}} g
$$

we can use an incremental SAT-solver

- R_0:, R_1, …: clauses stored in the SAT-solver
- A_0, A_1, …, b_0, b_1, …: local variables

Bottom-up proof search

Goal

Search for a derivation of $R_0, X \Rightarrow g$ in $C^{\rightarrow}$.

Bottom-up proof search procedure by exploiting an incremental SAT-solver to check classical validity.

(1) Add the clauses R_0 to the SAT-solver and check:

$$R_0 \vdash_{\mathrm{c}} g\,?$$

- If $R_0 \vdash_{\mathrm{c}} g$, then build the derivation:

$$\frac{R_0 \vdash_{\mathrm{c}} g}{R_0, X \Rightarrow g}\ \mathrm{cpl}_0$$

- Otherwise, choose $\langle \lambda_0, A_0 \rangle$ such that:

$$\lambda_0 = (a_0 \rightarrow b_0) \rightarrow c_0 \in X \qquad\qquad R_0, A_0 \vdash_{\mathrm{c}} b_0$$

  and apply:

$$\frac{R_0,\ A_0 \vdash_{\mathrm{c}} b_0 \qquad \overbrace{\varphi_0,\ R_0}^{R_1},\ X \Rightarrow g}{R_0,\ X \Rightarrow g}\ \lambda_0 \qquad \varphi_0 = \bigwedge(A_0 \setminus \{a_0\}) \rightarrow c_0$$

Bottom-up proof search

(2) We continue with the sequent

$$R_1,\, X \Rightarrow g \qquad R_1 \;=\; R_0 \cup \{\varphi_0\}$$

Add φ_0 to the SAT-solver and check

$$R_1 \vdash_{\mathrm{c}} g\,?$$

- If $R_1 \vdash_{\mathrm{c}} g$, then:

$$\dfrac{R_0,\, A_0 \vdash_{\mathrm{c}} b_0 \qquad \dfrac{R_1 \vdash_{\mathrm{c}} g}{R_1,\, X \Rightarrow g}}{R_0,\, X \Rightarrow g}\ \lambda_0$$

- Otherwise, choose $\langle \lambda_1, A_1 \rangle$ such that

$$\lambda_1 = (a_1 \to b_1) \to c_1 \in X \qquad\qquad R_1, A_1 \vdash_{\mathrm{c}} b_1$$

$$\dfrac{R_0,\, A_0 \vdash_{\mathrm{c}} b_0 \qquad \dfrac{R_1,\, A_1 \vdash_{\mathrm{c}} b_1 \qquad \overbrace{\varphi_1,\, R_1}^{R_2},\, X \Rightarrow g}{R_1,\, X \Rightarrow g}\ \lambda_1}{R_0,\, X \Rightarrow g}\ \lambda_0$$

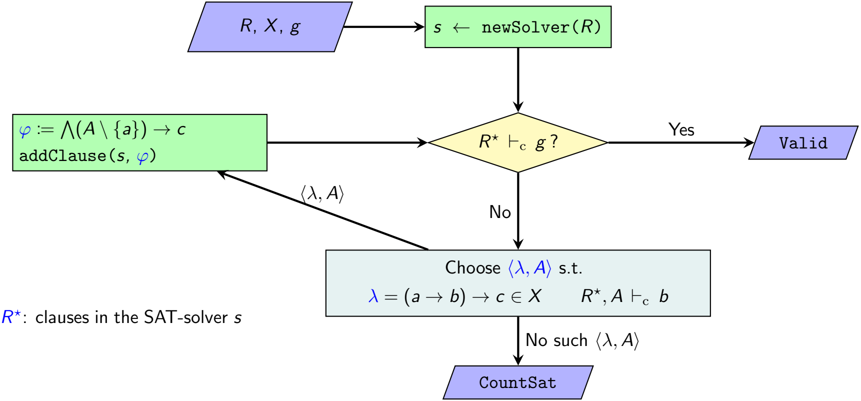
Bottom-up proof search

Input Assumptions

R : set of flat clauses $\varphi = \bigwedge A_1 \to \bigvee A_2$
X : set of impl. clauses $\lambda = (a \to b) \to c$
g : atom

Output Properties

`Valid`	implies $R, X \vdash_{\mathrm{i}} g$
`CountSat`	implies $R, X \nvdash_{\mathrm{i}} g$



$R^\star$: clauses in the SAT-solver s

Bottom-up proof search

Problem

A blind choice of $\langle \lambda, A \rangle$ might lead to non-termination

Example

Current sequent:

$$b \to c,\ (a \to b) \to c \Rightarrow g$$

Selected $\langle \lambda, A \rangle$:

$$\lambda \;=\; (a \to b) \to c \qquad\qquad A \;=\; \{b\}$$

Application of rule cpl_1:

$$\frac{b \to c,\ \overbrace{\{b\}}^{A} \vdash_{\mathrm{c}}\ b \qquad \varphi,\ b \to c,\ (a \to b) \to c \Rightarrow g}{b \to c,\ (a \to b) \to c \Rightarrow g}$$

$$\varphi \;=\; \bigwedge(A \setminus \{a\}) \to c \;=\; b \to c$$

Bottom-up proof search

We get a non-terminating loop!

$$\dfrac{b \to c,\, \{b\} \vdash_{\mathrm{c}} b \qquad \dfrac{b \to c,\, \{b\} \vdash_{\mathrm{c}} b \qquad \begin{array}{c}\vdots \\ \boxed{b \to c,\, (a \to b) \to c \Rightarrow g}\end{array}}{\boxed{b \to c,\, (a \to b) \to c \Rightarrow g}}}{\boxed{b \to c,\, (a \to b) \to c \Rightarrow g}}$$

Can we get an informed choice of $\langle \lambda, A \rangle$?

Bottom-up proof search

Current goal: prove the sequent $\sigma_k = R_k, X \Rightarrow g$

$$
\dfrac{R_0,\ A_0 \vdash_{\mathrm{c}}\ b_0 \qquad \dfrac{\begin{array}{c}\vdots \\ \dfrac{R_{k-1},\ A_{k-1} \vdash_{\mathrm{c}}\ b_{k-1} \qquad \overset{???}{R_k,\ X \Rightarrow g}}{R_{k-1},\ X \Rightarrow g}\ \lambda_{k-1}\end{array}}{R_1,\ X \Rightarrow g}}{R_0,\ X \Rightarrow g}\ \lambda_0 \qquad R_0 \subseteq R_1 \subseteq \cdots \subseteq R_k
$$

We search for a countermodel $\mathcal{K}$ for σ_k.

- If we find $\mathcal{K}$ then:
  $\mathcal{K}$ is a countermodel for $R_0, X \Rightarrow g$ (indeed, $R_0 \subseteq R_k$)
  We conclude `CountSat` (namely, $R_0, X \nvdash_{\mathrm{i}}\ g$)
- Othewise
  From the failure, we learn the proper choice of $\langle \lambda_k, A_k \rangle$

Countermodels

A Kripke model can be seen as a set of interpretations.

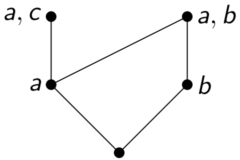
Let W be a finite set of interpretations with minimum M_0
(namely: $M_0 \subseteq M$, for every $M \in W$).
Then, $\mathcal{K}(W)$ is the Kripke model such that:

- The set of worlds is W;
- $M_1 \leq M_1$ (in $\mathcal{K}(W)$) iff $M_1 \subseteq M_2$;
- the root of $\mathcal{K}(W)$ is M_0;
- $M \Vdash p$ iff $p \in M$.

Example

$$W \;=\; \{\,\emptyset\,,\,\{a\}\,,\,\{b\}\,,\,\{a,c\}\,,\,\{a,b\}\,\}$$

$\mathcal{K}(W) \;=$



Countermodels

Let W be a finite set of interpretations.
We introduce the following realizability relation $\triangleright_W$ $(M \in W)$:

$$M \triangleright_W (a \to b) \to c \quad \text{iff} \quad a \in M \text{ or } b \in M \text{ or } c \in M \text{ or}$$
$$\exists M' \in W \text{ s.t.}$$
$$(M \subset M') \text{ and } a \in M' \text{ and } b \notin M'$$

Main property of $\triangleright_W$

Let W be a finite set of interpretations with minimum M_0.
Then, $\mathcal{K}(W)$ is a countermodel for $R, X \Rightarrow g$ iff:

- $g \notin M_0$;
- for every $M \in W$:

$$\begin{array}{lll} M \models R & \text{namely:} & M \models \varphi,\ \forall \varphi \in R \\ M \triangleright_W X & \text{namely:} & M \triangleright_W \lambda,\ \forall \lambda \in X \end{array}$$

We use the property to build countermodels.

Countermodels

Let $R, X \Rightarrow g$ be the current sequent to be proved in the main loop.

- If $R \vdash_{\mathrm{c}} g$, then there exists a derivation of $R, X \Rightarrow g$.
- Otherwise, the SAT-solver yields a model M s.t. $M \models R$ and $M \not\models g$. We set

$$W = \{M\}$$

We try to turn $\mathcal{K}(W)$ into a countermodel for $R, X \Rightarrow g$ by running a saturation process:

√ we add to W the worlds needed to fulfill the main property (inner loop).

Key point

- Suppose that there exists a pair $\langle w, \lambda \rangle$ such that

$$w \in W \qquad \lambda = (a \to b) \to c \in X \qquad w \not\triangleright_W \lambda$$

Then, we search for an interpretation w' s.t.:

$$w \subseteq w' \quad \text{and} \quad w' \models R \quad \text{and} \quad a \in w' \quad \text{and} \quad b \notin w'$$

√ If such a w' exists, we add w' to W and we continue to saturate

√ Otherwise, there is no countermodel for $R, X \Rightarrow g$.

Countermodels

How can we search for an interpretation w' s.t.:

$$w \subseteq w' \quad \text{and} \quad w' \models R \quad \text{and} \quad a \in w' \quad \text{and} \quad b \notin w' \ ?$$

Ask to the SAT-solver:

$$R, w, a \nvdash_{\mathrm{c}} b \ ?$$

Possible outcomes

- $\mathrm{Yes}(A)$

  This means that

  $$A \subseteq w \cup \{a\} \quad \text{and} \quad R, A \vdash_{\mathrm{c}} b$$

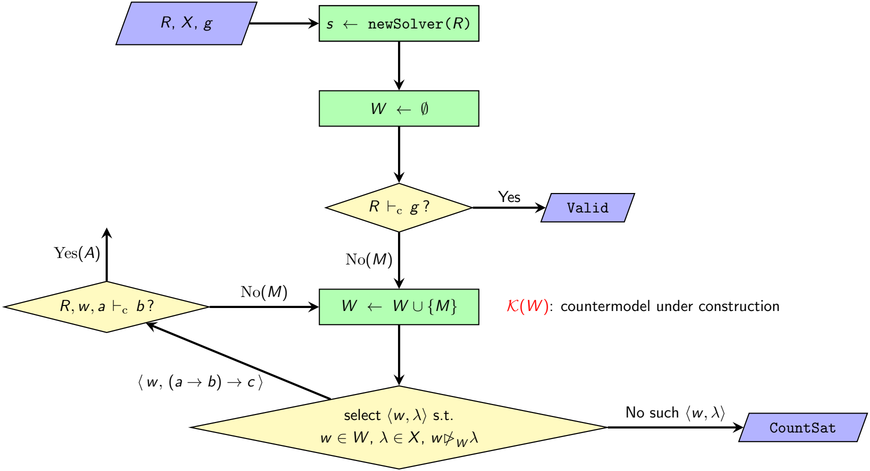
  Thus, $R, w, a \vdash_{\mathrm{c}} b$ and such a w' does not exist.
  Accordingly, the construction of the countermodel fails.
- $\mathrm{No}(M')$

  This means that

  $$M' \models R \cup w \cup \{a\} \quad \text{and} \quad M' \not\models b$$

  We set $w' = M'$.

Countermodels



Countermodels

Suppose that, after having chosen the pair $\langle w, \lambda \rangle$, the inner loop ends with $\mathrm{Yes}(A)$, meaning that $R, A \vdash_{\mathrm{c}} b$ (R: flat clauses in the SAT-solver).

Then, λ and A are the main formula and the local assumptions to be used:

$$\frac{R, A \vdash_{\mathrm{c}} b \qquad \varphi, R, X \Rightarrow g}{R, X \Rightarrow g}\ \lambda \qquad \begin{array}{l} \lambda \in X \\ \varphi = \bigwedge(A \setminus \{a\}) \to c \end{array}$$

- The learned clause φ is added to the SAT-solver
- We empty the set W (namely, we discard the current countermodel) and we perform a new iteration of the main loop (Restart)
  - √ At each restart, we execute the procedure from scratch.

    However, at each restart the SAT-solver is more powerful (we have added a new learned clause φ to it).

We call the obtained prover `intuitR` (`intuit` with Restart).

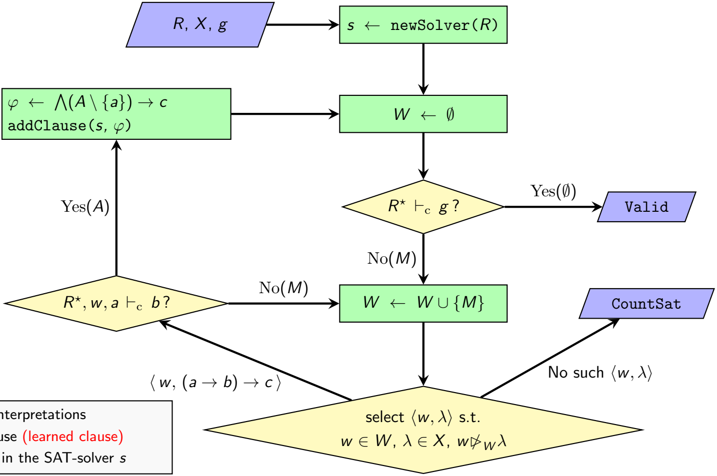
intuitR

Input Assumptions

R : set of flat clauses $\varphi = \bigwedge A_1 \to \bigvee A_2$
X : set of impl. clauses $\lambda = (a \to b) \to c$
g : atom

Output Properties

`Valid`	implies $R, X \vdash_{\mathrm{i}} g$
`CountSat`	implies $R, X \nvdash_{\mathrm{i}} g$



W:	set of interpretations
φ:	flat clause (learned clause)
$R^\star$:	clauses in the SAT-solver s

intuitR implementation

- We have implemented `intuitR` in Haskell on the top of `intuit`; we have added some useful features (e.g., trace of computations, construction of derivations/countermodels).
- As in `intuit`, we exploit the module MiniSat, a Haskell bundle of the MiniSat SAT-solver (but in principle we can use any incremental SAT-solver).
- The `intuitR` implementation can be downloaded at

  https://github.com/cfiorentini/intuitR.

intuitR vs intuit

- The proof search procedure of `intuitR` has a plain and intuitive presentation, consisting of two nested loops.
- Derivations have a linear structure, formalized by the calculus $C^{\rightarrow}$. Basically, a derivation in $C^{\rightarrow}$ is a cut-free derivation in $\mathrm{LJT}_{\texttt{SAT}}$ (the calculus of `intuit`) having only one branch.
- The countermodels obtained by `intuitR` are in general smaller than the ones obtained by `intuit`, since at every restart the model is reset.
- We have replicated the experiments performed for `intuit`: `intuitR` outperforms `intuit`.

Example 1 (Valid formula)

$$\chi = ((\eta_{12} \to \gamma) \wedge (\eta_{23} \to \gamma) \wedge (\eta_{31} \to \gamma)) \to \gamma$$

$$\eta_{ij} = p_i \leftrightarrow p_j \qquad \gamma = p_1 \wedge p_2 \wedge p_3$$

first instance of problem class SYJ201 from the ILTP library

Clausification

R_0: 17 flat clauses, X: 6 implication clauses

intuitR

14 call to the SAT-solver (6 Yes, 8 No), 6 Restart

$$\dfrac{R_0, p_2, \tilde{p}_6 \vdash_{c} p_1 \qquad \dfrac{R_1, p_3, \tilde{p}_1 \vdash_{c} p_1 \qquad \dfrac{R_2, p_1, \tilde{p}_{10} \vdash_{c} p_2 \qquad \dfrac{R_3, p_3 \vdash_{c} p_2 \qquad \dfrac{R_4, p_1, \tilde{p}_1 \vdash_{c} p_3 \qquad \dfrac{R_5, p_2 \vdash_{c} p_3 \qquad \dfrac{R_6 \vdash_{c} \tilde{g}}{R_6, X \Rightarrow \tilde{g}}}{R_5, X \Rightarrow \tilde{g}}\,\lambda_2}{R_4, X \Rightarrow \tilde{g}}\,\lambda_4}{R_3, X \Rightarrow \tilde{g}}\,\lambda_0}{R_2, X \Rightarrow \tilde{g}}\,\lambda_5}{R_1, X \Rightarrow \tilde{g}}\,\lambda_1}{R_0, X \Rightarrow \tilde{g}}\,\lambda_3$$

intuit vs intuitR: derivations

intuit

14 calls to the SAT-solver (7 Yes, 6 No)

$$
\begin{prooftree}
\AxiomC{$R_0\,,\,..\ \vdash_{\mathrm{c}}\ p_1$}
\UnaryInfC{$R_0\,,\,..\ \Rightarrow\ p_1$}
\AxiomC{$R_1\,,\,..\ \vdash_{\mathrm{c}}\ p_2$}
\UnaryInfC{$R_1\,,\,..\ \Rightarrow\ p_2$}
\AxiomC{$R_2\,,\,..\ \vdash_{\mathrm{c}}\ p_2$}
\UnaryInfC{$R_2\,,\,..\ \Rightarrow\ p_2$}
\RightLabel{λ_5}
\BinaryInfC{$R_1\,,\,..\ \Rightarrow\ p_2$}
\RightLabel{λ_3}
\BinaryInfC{$R_0\,,\,..\ \Rightarrow\ p_2$}
\AxiomC{$R_3\,,\,..\ \vdash_{\mathrm{c}}\ p_1$}
\UnaryInfC{$R_3\,,\,...\ \Rightarrow\ p_1$}
\AxiomC{$R_4\,,\,..\ \vdash_{\mathrm{c}}\ p_3$}
\UnaryInfC{$R_4\,,\,..\ \Rightarrow\ p_3$}
\AxiomC{$R_5\,,\,..\ \vdash_{\mathrm{c}}\ p_3$}
\UnaryInfC{$R_5\,,\,..\ \Rightarrow\ p_3$}
\RightLabel{λ_4}
\BinaryInfC{$R_4\,,\,..\ \Rightarrow\ p_3$}
\AxiomC{$R_6\ \vdash_{\mathrm{c}}\ \tilde{g}$}
\UnaryInfC{$R_6\,,\,..\ \Rightarrow\ \tilde{g}$}
\RightLabel{λ_2}
\BinaryInfC{$R_5\,,\,..\ \Rightarrow\ \tilde{g}$}
\RightLabel{λ_1}
\BinaryInfC{$R_3\,,\,..\ \Rightarrow\ \tilde{g}$}
\RightLabel{λ_0}
\BinaryInfC{$R_0\,,\,\varphi_0\,,\,\varphi_1\,,\,\varphi_4\,,\,X\ \Rightarrow\ \tilde{g}$}
\doubleLine
\RightLabel{cut (3 times)}
\UnaryInfC{$R_0\,,\,X\ \Rightarrow\ \tilde{g}$}
\end{prooftree}
$$

Cuts are needed to drip out the extra learned clauses φ_0, φ_1, φ_4.

Actually, one can prove that each learned clause φ_k satisfies

$$R_0, X \vdash_{\mathrm{i}} \varphi_k$$

intuit vs intuitR: countermodels

Example 2 (CountSat formula)

$$\psi \quad = ((\eta_{12} \to \gamma) \wedge (\eta_{23} \to \gamma) \wedge (\eta_{34} \to \gamma) \wedge (\eta_{41} \to \gamma)) \;\to\; (p_0 \vee \neg p_0 \vee \gamma)$$

$$\eta_{ij} = p_i \leftrightarrow p_j \qquad \gamma = \; p_1 \wedge p_2 \wedge p_3 \wedge p_4$$

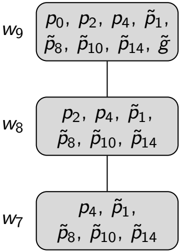
first instance of problem class SYJ207 from the ILTP library

Clausification

R_0: 24 flat clauses, X: 9 implication clauses

intuitR

14 call to the SAT-solver (4 Yes, 10 No), 4 Restart

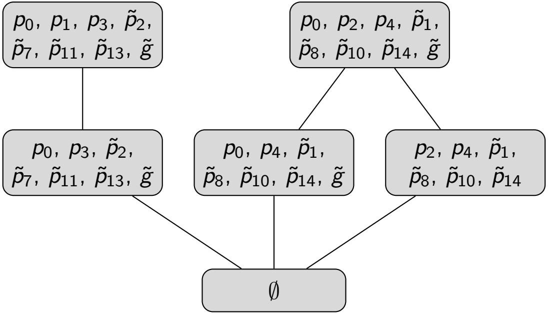


$\mathcal{K}(\{w_7, w_8, w_9\})$

intuit vs intuitR: countermodels

intuit

31 calls to the SAT-solver (24 No, 7 Yes)



6 worlds

intuit vs intuitR: experiments

We compare intuitR with the state-of-the-art provers for IPL:

- intuit
- fCube [Ferrari et al. LPAR 2010]

  Standard tableaux calculus with simplification rules
- intHistGC [Goré et al., IJCAR 2014]

  Sequent calculus with histories, dependency directed backtracking for global caching

Benchmarks

- 1200 problems (498 Valid and 702 CountSat)
- timeout: 600 seconds.

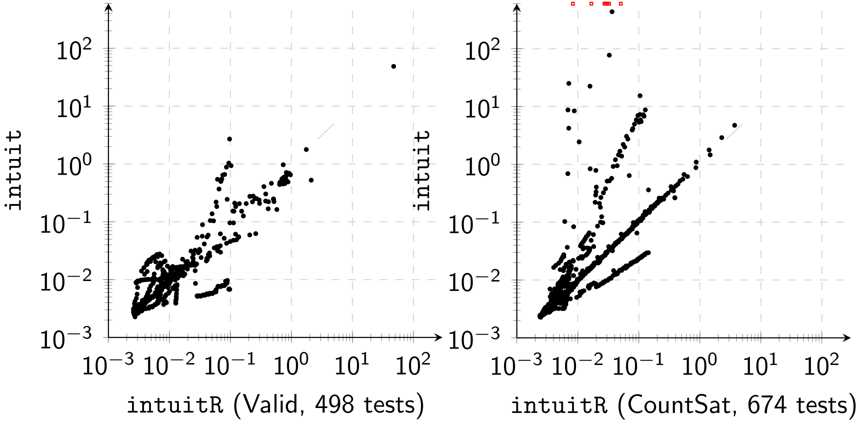
Outcome

- intuitR solve more problems than its competitors
- In families SYJ201 (Valid formulas) and SYJ207 (CountSat formulas) of ILTP library, intuitR outperforms its rivals
- In all the other cases (except 3 families), intuitR is comparable to the best prover (which is intuit in most cases).

intuit vs intuitR: experiments

Class (# problems)	intuitR	intuit	fCube	intHistGC
SYJ201(50)	50 (2.259)	50 (11.494)	50 (259.776)	50 (39.466)
SYJ202(38)	$10^\star$ (49.265)	$10^\star$ (50.658)	$9^\star$ (176.984)	$6^\star$ (324.673)
SYJ203(50)	50 (0.250)	50 (0.335)	50 (1.671)	50 (0.293)
SYJ204(50)	50 (0.442)	50 (0.477)	50 (0.972)	50 (0.203)
SYJ205(50)	50 (0.500)	50 (0.730)	50 (1.317)	50 (4.129)
SYJ206(50)	50 (0.303)	50 (0.348)	50 (0.759)	50 (0.112)
SYJ207(50)	50 (2.291)	50 (109.919)	50 (138.546)	50 (1014.476)
SYJ208(38)	38 (5.225)	38 (5.479)	$29^\star$ (2.755)	38 (497.715)
SYJ209(50)	50 (0.226)	50 (0.278)	50 (1.690)	50 (0.254)
SYJ210(50)	50 (0.272)	50 (0.252)	50 (0.988)	50 (0.288)
SYJ211(50)	50 (0.462)	50 (1.251)	50 (1.073)	50 (63.686)
SYJ212(50)	50 (0.669)	$42^\star$ (587.794)	50 (2.698)	50 (1.624)
EC(100)	100 (2.738)	100 (0.821)	100 (6.183)	100 (0.651)
negEC(100)	100 (3.614)	100 (1.116)	100 (13.733)	100 (5.807)
cross(4)	4 (0.100)	4 (0.097)	4 (3.417)	$2^\star$ (0.005)
jm_cross(4)	4 (0.120)	4 (0.090)	4 (5.404)	$3^\star$ (4.324)
jm_lift(3)	3 (0.170)	3 (0.133)	3 (6.847)	$2^\star$ (0.028)
lift(3)	3 (0.119)	3 (0.102)	3 (6.494)	$2^\star$ (0.012)
mapf(4)	4 (0.187)	4 (0.400)	4 (446.921)	$3^\star$ (0.043)
portia(100)	100 (32.878)	100 (22.596)	100 (3255.818)	100 (3200.135)
negportia(100)	100 (7.956)	100 (8.309)	$98^\star$ (3826.011)	100 (28.289)
negportiav2(100)	100 (8.081)	100 (8.411)	$98^\star$ (1264.103)	100 (3212.293)
nishimura2(28)	28 (9.784)	28 (12.285)	$27^\star$ (141.326)	28 (7.616)
Unsolved	28	36	43	38

intuit vs intuitR: experiments



Comparison between intuitR and intuit (1172 problems, the 28 problems where both provers run out of time have been omitted); time axis are logarithmic, the 8 red squares indicates that intuit has exceeded the timeout

Conclusions

- `intuitR` can be extended to deal with some superintuitionistic logics.

  Key idea:

  - ✓ if the countermodel under construction is not a model of the logic, the inner loop fails, and we run a new iteration of the main loop.
  - ✓ From the failure, we learn new clauses to add to the SAT-solver (corresponding to instances of the axiom schema of the logic).

- Other generalizations suggested in [Claessen&Rosen,LPAR 2015] (modal logics, fragments of first-order logic) seem to be more challenging.
- The `intuitR` implementation and other additional material (e.g., the omitted proofs, a detailed report on experiments) can be downloaded at

  https://github.com/cfiorentini/intuitR.