intuitR: A Theorem Prover for Intuitionistic Propositional Logic

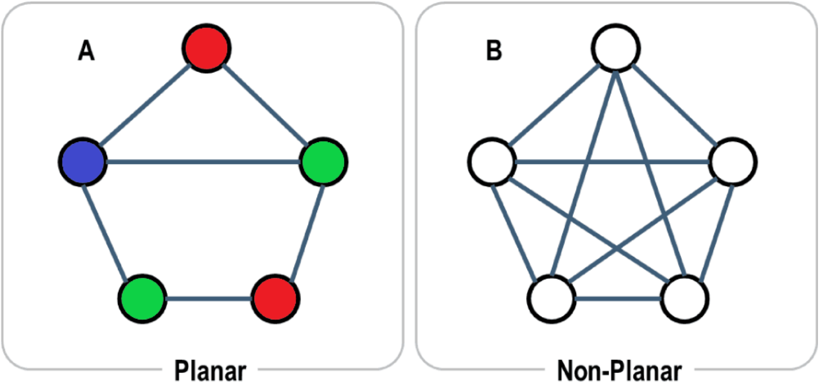
Erik Rauer

Division of Science and Mathematics
University of Minnesota, Morris

April 14, 2022

Motivation

Want to write program that assigns each vertex of a planar graph a color, such that no two adjacent vertices are the same color, using smallest number of different colors.



Motivation

Want to write program that assigns each vertex of a planar graph a color, such that no two adjacent vertices are the same color, using smallest number of different colors.

Four Color Theorem

No more than 4 colors are needed to color a planar graph in this way.

Motivation

Want to write program that assigns each vertex of a planar graph a color, such that no two adjacent vertices are the same color, using smallest number of different colors.

Four Color Theorem

No more than 4 colors are needed to color a planar graph in this way.

Proof gives step by step details of how to a color any graph in this way, can write our algorithm based on the proof.
Call this a *constructive proof*.

Non-Constructive Proof Example

Want to show the existence of irrational numbers a and b, such that a^b is rational.

Non-Constructive Proof Example

Want to show the existence of irrational numbers a and b, such that a^b is rational.

Assume $\sqrt{2}^{\sqrt{2}}$ is either rational or irrational.

Non-Constructive Proof Example

Want to show the existence of irrational numbers a and b, such that a^b is rational.

Assume $\sqrt{2}^{\sqrt{2}}$ is either rational or irrational.

$\Downarrow$

Either $\sqrt{2}^{\sqrt{2}}$ or $\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}}$ is rational.

But don't know which.

Intuitionistic Logic

Logic system that attempts to emulate/force constructive proofs

Same as classical logic, except it doesn't allow:

- Law of Excluded Middle: $p \vee \neg p$
- Double Negation Elimination: $\neg\neg p \equiv p$

Intuitionistic Propositional Logic (IPL)

Propositional logic form of intuitionistic logic, i.e. no quantifiers ($\forall$ and $\exists$)

Only $\wedge, \vee, \neg, \rightarrow, \bot, \top$

Difficult to determine IPL-validity, so use IPL-provers like *intuitR* by Fiorentini (2021) or *intuit* by Claessen and Rosén (2015)

Outline

Outline

How to decide if a formula α is IPL-valid?

Kripke Models

$(W, \delta, \leq, r)$

How to decide if a formula α is IPL-valid?

Kripke Models

$(W, \delta, \leq, r)$

- W set of worlds

How to decide if a formula α is IPL-valid?

Kripke Models

$(W, \delta, \leq, r)$

- W set of worlds
- δ mapping from W to set of propositional variables

How to decide if a formula α is IPL-valid?

Kripke Models

$(W, \delta, \leq, r)$

- W set of worlds
- δ mapping from W to set of propositional variables
- $\leq$ ordering of worlds such that for all $k \leq k'$, $\delta(k) \subseteq \delta(k')$

How to decide if a formula α is IPL-valid?

Kripke Models

$(W, \delta, \leq, r)$

- W set of worlds
- δ mapping from W to set of propositional variables
- $\leq$ ordering of worlds such that for all $k \leq k'$, $\delta(k) \subseteq \delta(k')$
- r minimum world

How to decide if a formula α is IPL-valid?

Kripke Models

$(W, \delta, \leq, r)$

- W set of worlds
- δ mapping from W to set of propositional variables
- $\leq$ ordering of worlds such that for all $k \leq k'$, $\delta(k) \subseteq \delta(k')$
- r minimum world

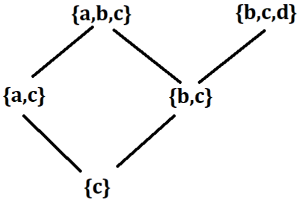


Figure: Visual Representation of a Kripke Model

Forcing

For $k \in W$ and logical formula α, $k \vDash \alpha$ based on the following rules (from Moschovakis (2021)):

1. $k \vDash p, \text{ for every } p \in \delta(k)$
2. $k \nvDash \perp$

Forcing

For $k \in W$ and logical formula α, $k \vDash \alpha$ based on the following rules (from Moschovakis (2021)):

1. $k \vDash p, \text{ for every } p \in \delta(k)$
2. $k \nvDash \perp$
3. $k \vDash P \wedge Q, \text{ if } k \vDash P \text{ and } k \vDash Q$
4. $k \vDash P \vee Q, \text{ if } k \vDash P \text{ or } k \vDash Q$

Forcing

For $k \in W$ and logical formula α, $k \vDash \alpha$ based on the following rules (from Moschovakis (2021)):

1. $k \vDash p$, for every $p \in \delta(k)$
2. $k \nvDash \perp$
3. $k \vDash P \wedge Q$, if $k \vDash P$ and $k \vDash Q$
4. $k \vDash P \vee Q$, if $k \vDash P$ or $k \vDash Q$
5. $k \vDash \neg P$, if for all $k' \geq k, k' \nvDash P$

Forcing

For $k \in W$ and logical formula α, $k \vDash \alpha$ based on the following rules (from Moschovakis (2021)):

1. $k \vDash p, \text{ for every } p \in \delta(k)$
2. $k \nvDash \bot$
3. $k \vDash P \wedge Q, \text{ if } k \vDash P \text{ and } k \vDash Q$
4. $k \vDash P \vee Q, \text{ if } k \vDash P \text{ or } k \vDash Q$
5. $k \vDash \neg P, \text{ if for all } k' \geq k, k' \nvDash P$
6. $k \vDash P \rightarrow Q, \text{ if for every } k' \geq k, \text{ if } k' \vDash P, \text{ then } k' \vDash Q$

Determining IPL-validity

A formula α is IPL-valid iff for every kripke model with root r, $r \vDash \alpha$.

Call a kripke model where the root $r \nvDash \alpha$ a *countermodel* for α

Example

Countermodel for $p \vee \neg p$:

$(\{k, k'\}, \delta, \leq, k)$, where

- $\delta(k) = \emptyset$
- $\delta(k') = \{p\}$
- $k < k'$

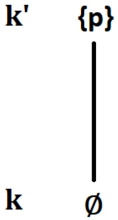


Figure: Visual representation of Kripke Countermodel for $p \vee \neg p$

Outline

SAT solver

Program that solves the *Boolean Satisfiability Problem*:
Given a propositional formula α is there an assignment of variables, such that α is true?

SAT solver

Program that solves the *Boolean Satisfiability Problem*:
Given a propositional formula α is there an assignment of variables, such that α is true?

When in form $\alpha \rightarrow p$, can reinterpret as "Does α being true make p true?" or "Does α prove p classically?"

Methods Required for *intuitR*

- *newSolver()*
  - Create a new SAT solver

Methods Required for *intuitR*

- *newSolver()*
  - Create a new SAT solver
- *addClause(s, ρ)*
  - Add clause ρ to SAT solver's existing clauses $R(s)$

Methods Required for *intuitR*

- *newSolver()*
  - Create a new SAT solver
- *addClause(s, ρ)*
  - Add clause ρ to SAT solver's existing clauses $R(s)$
- *satProve(s, A, g)*
  - Use SAT solver s to prove g based on the clauses $R(s)$ that have already been added and the set of propositional variables A that are assumed to be true

Methods Required for *intuitR*

- *newSolver()*
  - Create a new SAT solver
- *addClause(s, ρ)*
  - Add clause ρ to SAT solver's existing clauses $R(s)$
- *satProve(s, A, g)*
  - Use SAT solver s to prove g based on the clauses $R(s)$ that have already been added and the set of propositional variables A that are assumed to be true
  - Returns:
    - *YES(A′)*, if $R(s)$ and $A' \subseteq A$ being true makes g true

Methods Required for *intuitR*

- *newSolver()*
  - Create a new SAT solver
- *addClause(s, ρ)*
  - Add clause ρ to SAT solver's existing clauses $R(s)$
- *satProve(s, A, g)*
  - Use SAT solver s to prove g based on the clauses $R(s)$ that have already been added and the set of propositional variables A that are assumed to be true
  - Returns:
    - *YES(A')*, if $R(s)$ and $A' \subseteq A$ being true makes g true
    - *NO(M)*, if $R(s)$ and the set of propositional variables $M \supseteq A$ are both true, but g is false

Outline

Goal

Convert formula to an *r-sequent*, denoted $R, X \Rightarrow g$:

$(\bigwedge R \wedge \bigwedge X) \rightarrow g$

Goal

Convert formula to an *r-sequent*, denoted $R, X \Rightarrow g$:

$(\bigwedge R \wedge \bigwedge X) \rightarrow g$

where:

- R a set of flat clauses: $(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \rightarrow (b_1 \vee b_2 \vee \ldots \vee b_m)$

Goal

Convert formula to an *r-sequent*, denoted $R, X \Rightarrow g$:

$(\bigwedge R \wedge \bigwedge X) \rightarrow g$

where:

- R a set of flat clauses: $(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \rightarrow (b_1 \vee b_2 \vee \ldots \vee b_m)$
- X a set of implication clauses: $(a \rightarrow b) \rightarrow c$

Goal

Convert formula to an *r-sequent*, denoted $R, X \Rightarrow g$:

$(\bigwedge R \wedge \bigwedge X) \to g$

where:

- R a set of flat clauses: $(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \to (b_1 \vee b_2 \vee \ldots \vee b_m)$
- X a set of implication clauses: $(a \to b) \to c$
- for each $(a \to b) \to c \in X$, $b \to c \in R$

Goal

Convert formula to an *r-sequent*, denoted $R, X \Rightarrow g$:

$$(\bigwedge R \wedge \bigwedge X) \rightarrow g$$

where:

- R a set of flat clauses: $(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \rightarrow (b_1 \vee b_2 \vee \ldots \vee b_m)$
- X a set of implication clauses: $(a \rightarrow b) \rightarrow c$
- for each $(a \rightarrow b) \rightarrow c \in X$, $b \rightarrow c \in R$
- g a propositional variable

Example

For $p \vee \neg p$:

- $R = \{p \to g, \bot \to g\}$
- $X = \{(p \to \bot) \to g\}$
- $g = g$ (introduced during clausification)

Or $(p \to g) \wedge (\bot \to g) \wedge ((p \to \bot) \to g) \to g$

Outline

cpl_0 and cpl_1

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

cpl_0 and cpl_1

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

$$\frac{R,\ A \vdash_c b \qquad R,\ \varphi,\ X \Rightarrow g}{R,\ X \Rightarrow g}\ \mathrm{cpl}_1 \qquad \begin{array}{l} (a \to b) \to c \in X \\ A \subseteq V \\ \varphi\ =\ \bigwedge(A \setminus \{a\}) \to c \end{array}$$

Outline

proveR

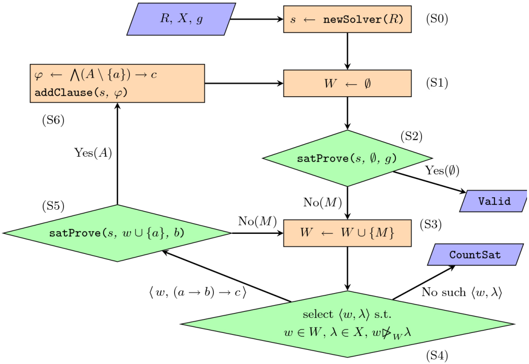
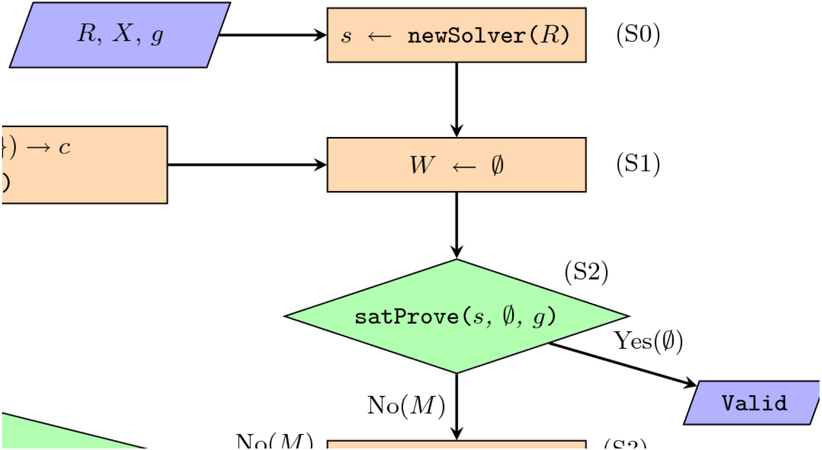
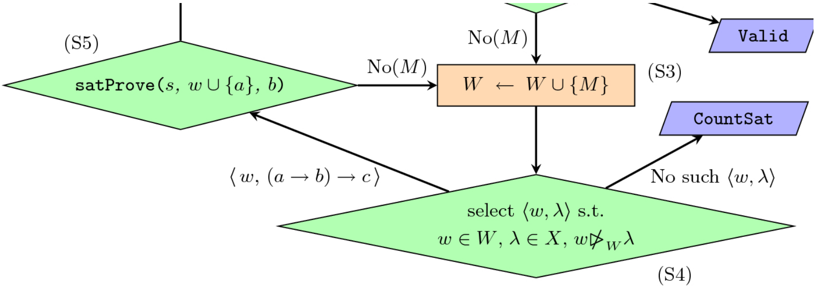


Figure: Flowchart of the *proveR* algorithm

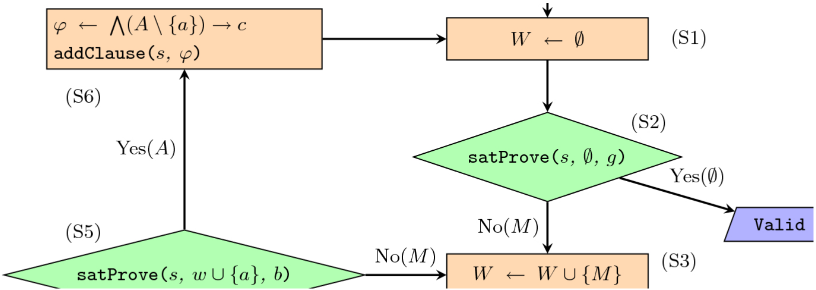
Initialization



Inner Loop


Valid
(S5)
No(M)
No(M)
satProve(s, $w \cup \{a\}$, b)
$W \leftarrow W \cup \{M\}$
(S3)
CountSat
$\langle w, (a \to b) \to c \rangle$
No such $\langle w, \lambda \rangle$
select $\langle w, \lambda \rangle$ s.t.
$w \in W, \lambda \in X, w \not\triangleright_W \lambda$
(S4)

Restart Outer Loop



Outline

Experiment

intuitR was compared to three other IPL-provers:
intuit, *fCube*, and *intHistGC*

Ran on a benchmark set of 1200 problems, split into 32 groups
498 problems were IPL-valid, 702 were not

Outline

Significant Results

Problem Set (Number of Problems)	intuitR	intuit	fCube	intHistGC
SYJ201 (50)	**50 (2.259)**	50 (11.494)	50 (259.776)	50 (39.466)
SYJ207 (50)	**50 (2.291)**	50 (109.919)	50 (138.546)	50 (1014.476)
SYJ211 (50)	**50 (0.462)**	50 (1.251)	50 (1.073)	50 (63.686)
SYJ212 (50)	**50 (0.669)**	42 (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)
portia (100)	100 (32.878)	**100 (22.596)**	100 (3255.818)	100 (3200.135)
Total Unsolved	**28**	36	43	38
Total Time (For These Problems)	**44.911**	734.991	3677.827	4325.845

Table: Most significant results from Fiorentini (2021). Number of problems solved, followed by the time taken to solve said problems (in seconds). Fastest prover highlighted.

Conclusion

Questions?

References

Koen Claessen and Dan Rosén. 2015. SAT Modulo Intuitionistic Implications. In *Logic for Programming, Artificial Intelligence, and Reasoning*, Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 622–637.

Camillo Fiorentini. 2021. Efficient SAT-based Proof Search in Intuitionistic Propositional Logic. In *Automated Deduction – CADE 28*, André Platzer and Geoff Sutcliffe (Eds.). Springer International Publishing, Cham, 217–233.

Joan Moschovakis. 2021. Intuitionistic Logic. In *The Stanford Encyclopedia of Philosophy* (Fall 2021 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.