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: $p \lor \neg p$
- 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 \textit{intuitR} by Fiorentini (2021) or \textit{intuit} by Claessen and Rosén (2015)
Outline

Introduction

Background
- Kripke Semantics and Models
- SAT Solvers

intuitR
- Clausification Procedure
- Logic Rules
- proveR Algorithm

Experiment and Results
- Experiment
- Results

Conclusion
Outline

Introduction

Background

- Kripke Semantics and Models
- SAT Solvers

intuitR

- Clausification Procedure
- Logic Rules

proveR Algorithm

Experiment and Results

- Experiment
- Results

Conclusion
How to decide if a formula $\alpha$ is IPL-valid?

Kripke Models

$(W, \delta, \leq, r)$
How to decide if a formula $\alpha$ is IPL-valid?

Kripke Models

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

- $W$ set of worlds
How to decide if a formula $\alpha$ is IPL-valid?

Kripke Models

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

- $W$ set of worlds
- $\delta$ mapping from $W$ to set of propositional variables
How to decide if a formula $\alpha$ is IPL-valid?

Kripke Models

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

- $W$ set of worlds
- $\delta$ 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 $\alpha$ is IPL-valid?

Kripke Models

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

- $W$ set of worlds
- $\delta$ 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
Kripke Semantics and Models

How to decide if a formula $\alpha$ is IPL-valid?

Kripke Models

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

- $W$ set of worlds
- $\delta$ 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 $\alpha$, $k \models \alpha$ based on the following rules (from Moschovakis (2021)):

1. $k \models p$, for every $p \in \delta(k)$
2. $k \not\models \bot$
Forcing

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

1. $k \models p$, for every $p \in \delta(k)$
2. $k \not\models \bot$
3. $k \models P \land Q$, if $k \models P$ and $k \models Q$
4. $k \models P \lor Q$, if $k \models P$ or $k \models Q$
Kripke Semantics and Models

Forcing

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

1. $k \models p$, for every $p \in \delta(k)$
2. $k \not\models \bot$
3. $k \models P \land Q$, if $k \models P$ and $k \models Q$
4. $k \models P \lor Q$, if $k \models P$ or $k \models Q$
5. $k \models \neg P$, if for all $k' \geq k$, $k' \not\models P$
Kripke Semantics and Models

Forcing

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

1. $k \models p$, for every $p \in \delta(k)$
2. $k \not\models \bot$
3. $k \models P \land Q$, if $k \models P$ and $k \models Q$
4. $k \models P \lor Q$, if $k \models P$ or $k \models Q$
5. $k \models \neg P$, if for all $k' \geq k$, $k' \not\models P$
6. $k \models P \rightarrow Q$, if for every $k' \geq k$, if $k' \models P$, then $k' \models Q$
Determining IPL-validity

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

Call a kripke model where the root $r \not\models \alpha$ a *countermodel* for $\alpha$. 
Example

Countermodel for $p \lor \neg p$:

$\langle \{k, k'\}, \delta, \leq, k \rangle$, where

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

**Figure**: Visual representation of Kripke Countermodel for $p \lor \neg p$
Outline

Introduction

Background
  Kripke Semantics and Models
  SAT Solvers

intuitR
  Clausification Procedure
  Logic Rules

proveR Algorithm

Experiment and Results
  Experiment
  Results

Conclusion
SAT solver

Program that solves the *Boolean Satisfiability Problem*: Given a propositional formula $\alpha$ is there an assignment of variables, such that $\alpha$ is true?
SAT solver

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

When in form $\alpha \rightarrow p$, can reinterpret as “Does $\alpha$ being true make $p$ true?” or “Does $\alpha$ 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, \( \rho \))**
  - Add clause \( \rho \) 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’ ⊆ A being true makes g true
    - **NO(M)**, if R(s) and the set of propositional variables M ⊇ A are both true, but g is false
Outline

Introduction

Background
  Kripke Semantics and Models
  SAT Solvers

intuitR
  Clausification Procedure
  Logic Rules

proveR Algorithm

Experiment and Results
  Experiment
  Results

Conclusion
Clausification Procedure

Goal

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

\[
(\bigwedge R \land \bigwedge X) \rightarrow g
\]
Goal

Convert formula to an \( r\)-sequent, denoted \( R, X \Rightarrow g \):

\[
(\bigwedge R \land \bigwedge X) \rightarrow g
\]

where:

- \( R \) a set of flat clauses: \( (a_1 \land a_2 \land \ldots \land a_n) \rightarrow (b_1 \lor b_2 \lor \ldots \lor b_m) \)
Clausification Procedure

Goal

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

$$(\land R \land \land X) \rightarrow g$$

where:

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

Goal

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

$$(\land R \land \land X) \rightarrow g$$

where:

- $R$ a set of flat clauses: $(a_1 \land a_2 \land ... \land a_n) \rightarrow (b_1 \lor b_2 \lor ... \lor b_m)$
- $X$ a set of implication clauses: $(a \rightarrow b) \rightarrow c$
- for each $(a \rightarrow b) \rightarrow c \in X, b \rightarrow c \in R$
Goal

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

$$(\land R \land \land X) \rightarrow g$$

where:

- $R$ a set of flat clauses: $(a_1 \land a_2 \land ... \land a_n) \rightarrow (b_1 \lor b_2 \lor ... \lor b_m)$
- $X$ 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 $p \lor \neg p$:

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

Or $(p \rightarrow g) \land (\bot \rightarrow g) \land ((p \rightarrow \bot) \rightarrow g) \rightarrow g$
Outline

Introduction

Background
  Kripke Semantics and Models
  SAT Solvers

intuitR
  Clausification Procedure
  Logic Rules
  proveR Algorithm

Experiment and Results
  Experiment
  Results

Conclusion

References
$cpl_0$ and $cpl_1$

$$\frac{R \vdash_c g}{R, X \Rightarrow g} \quad cpl_0$$
$cpl_0$ and $cpl_1$

\[
\frac{R \vdash_c g}{R, X \Rightarrow g} \quad cpl_0
\]

\[
\frac{R, A \vdash_c b \quad R, \varphi, X \Rightarrow g}{R, X \Rightarrow g} \quad cpl_1
\]

\[
(a \rightarrow b) \rightarrow c \in X \\
A \subseteq V \\
\varphi = \bigwedge(A \setminus \{a\}) \rightarrow c
\]
Outline

Introduction

Background
  Kripke Semantics and Models
  SAT Solvers

intuitR
  Clausification Procedure
  Logic Rules

proveR Algorithm

Experiment and Results
  Experiment
  Results

Conclusion
proveR

Figure: Flowchart of the proveR algorithm
proveR Algorithm

Initialization

- **R, X, g**
- **s ⇐ newSolver(R)** (S0)
- **W ⇐ ∅** (S1)
- **satProve(s, ∅, g)** (S2)
  - Yes(∅)
  - No(M)
- **Valid**
proveR Algorithm

**Inner Loop**

1. **(S5) satProve(s, w ∪ {a}, b)**
2. **No(M) → W ← W ∪ {M}**
3. **No such ⟨w, λ⟩**
4. **select ⟨w, λ⟩ s.t.\ w ∈ W, λ ∈ X, w ⊭_{W, λ}**
5. **Valid**
6. **CountSat**
**proveR Algorithm**

**Restart Outer Loop**

\[ \varphi \leftarrow \bigwedge\{A \setminus \{a\}\} \rightarrow c \]

\[ \text{addClause}(s, \varphi) \]

(S6)

Yes(A)

(S5)

\[ \text{satProve}(s, w \cup \{a\}, b) \]

(S1)

\[ W \leftarrow \emptyset \]

(S2)

\[ \text{satProve}(s, \emptyset, g) \]

Yes(\emptyset)

No(M)

(S3)

\[ W \leftarrow W \cup \{M\} \]

Valid
Outline

Introduction

Background
  Kripke Semantics and Models
  SAT Solvers

intuitR
  Clausification Procedure
  Logic Rules
  proveR Algorithm

Experiment and Results
  Experiment
  Results

Conclusion
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

Introduction

Background
  Kripke Semantics and Models
  SAT Solvers

intuitR
  Clausification Procedure
  Logic Rules
  proveR Algorithm

Experiment and Results
  Experiment
  Results

Conclusion
## Significant Results

<table>
<thead>
<tr>
<th>Problem Set (Number of Problems)</th>
<th>intuitR</th>
<th>intuit</th>
<th>fCube</th>
<th>intHistGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYJ201 (50)</td>
<td>50 (2.259)</td>
<td>50 (11.494)</td>
<td>50 (259.776)</td>
<td>50 (39.466)</td>
</tr>
<tr>
<td>SYJ207 (50)</td>
<td>50 (2.291)</td>
<td>50 (109.919)</td>
<td>50 (138.546)</td>
<td>50 (1014.476)</td>
</tr>
<tr>
<td>SYJ211 (50)</td>
<td>50 (0.462)</td>
<td>50 (1.251)</td>
<td>50 (1.073)</td>
<td>50 (63.686)</td>
</tr>
<tr>
<td>SYJ212 (50)</td>
<td>50 (0.669)</td>
<td>42 (587.794)</td>
<td>50 (2.698)</td>
<td>50 (1.624)</td>
</tr>
<tr>
<td>EC (100)</td>
<td>100 (2.738)</td>
<td>100 (0.821)</td>
<td>100 (6.183)</td>
<td>100 (0.651)</td>
</tr>
<tr>
<td>negEC (100)</td>
<td>100 (3.614)</td>
<td><strong>100 (1.116)</strong></td>
<td>100 (13.733)</td>
<td>100 (5.807)</td>
</tr>
<tr>
<td>portia (100)</td>
<td>100 (32.878)</td>
<td><strong>100 (22.596)</strong></td>
<td>100 (3255.818)</td>
<td>100 (3200.135)</td>
</tr>
<tr>
<td>Total Unsolved</td>
<td>28</td>
<td>36</td>
<td>43</td>
<td>38</td>
</tr>
<tr>
<td>Total Time (For These Problems)</td>
<td>44.911</td>
<td>734.991</td>
<td>3677.827</td>
<td>4325.845</td>
</tr>
</tbody>
</table>

**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

Introduction

Background
  Kripke Semantics and Models
  SAT Solvers

intuitR
  Clausification Procedure
  Logic Rules
  proveR Algorithm

Experiment and Results
  Experiment
  Results

Conclusion
Questions?
References

