Boolean Logic

Specifications section, Logic topic, Lecture 2

Pavel Panchekha
CS 6110, U of Utah
9 January 2020
Software Verification

No Bugs

Bugs are bad
Challenges

Writing a specification for quicksort

Reasoning about predicates like sorted

Combining facts about lists and predicates

Propagating facts through the program

Expanding the specification to make it provable

Verification is hard!
Recent Successes

Quark  COMPCCERT  Memory Model  IronClad

ROSETTE  Z3  SLAM

Dafny  seL4  frenetic

alloy
Class Progress

Logical reasoning

Program logics

Static analysis

First-order Logic

Decision Procedures

Boolean logic

Syntax

Proof

Algorithm
Boolean logic

Language for stating facts about booleans
And/or/not, conjunctive form, universality

Proofs of boolean logic facts by refutation
Compact evidence of truth/falsity

Algorithm to automatically find proofs
Efficient in simple cases, core of modern solvers
Boolean Specifications
Syntax, Semantics, and Transformations
Specifications

Generally describe **facts** about the program

Which may or may not, but **should**, be true

Phrased in terms of **values** that flow through program

Arguments, returns, variables, loop indices, system calls, …

Today, **simplest** kind of value: booleans
Facts about Booleans

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
<th>Both</th>
<th>Neither</th>
<th>One+</th>
<th>One</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$\neg x$</td>
<td>$x \land y$</td>
<td>$\neg(x \lor y)$</td>
<td>$x \lor y$</td>
<td>$x \lor y$</td>
</tr>
</tbody>
</table>

If, Iff, Two of 3, Always, Never, ... 

$x \Rightarrow y$  
$x \Leftrightarrow y$  
$2\{x, y, z\}_2$  
$\top$  
$\bot$
Facts about Booleans

$x, y := x \mid \neg x \mid x \land y \mid x \lor y$

That's what you can say; what does it mean?

Depends on the values of the variables

\[
\begin{align*}
[v] &= "v is True" \\
[x \land y] &= \text{both } [x] \text{ and } [y] \\
[\neg x] &= \text{not } [x] \\
[x \lor y] &= \text{either } [x] \text{ or } [y] \text{ or both}
\end{align*}
\]
Conjunctive form

\[ x \land \neg x = \bot \quad \neg \neg x = x \quad x \lor y = \neg (\neg x \land \neg y) \]

**Conjunctive form**: and, then or, then not, then variables

\[ (x \land y) \lor (a \land b) = (x \lor (a \land b)) \land (y \lor (a \land b)) \]
\[ = (x \lor a) \land (x \lor b) \land (y \lor a) \land (y \lor b) \]

Any boolean logic fact can be **put into conjunctive form**

But can it be **done quickly**?
Conjunctive form

\((x \land y) \lor (a \land b)\)
Conjunctive form

\[( A ) \lor ( B )\]

\[A = x \land y\]

\[B = a \land b\]
Conjunctive form

\[
\begin{align*}
( A ) \lor ( B ) \\
(A \land x \land y) \lor (\neg A \land \neg (x \land y)) \\
(B \land a \land b) \lor (\neg B \land \neg (a \land b))
\end{align*}
\]

One per term

Each is small

“Tseytin transformation” to conjunctive form

Constant factor increase in expression size
What use is a Spec?

That’s what you can **say**; what does it **mean**?

- Depends on the **values of the variables** 😞

We want to use the spec **before** running a program

- Variables **don’t have values** yet!

- “**Could it** be true?” “**Must it** be true?” “**If** this, **then** that?”

- **Satisfiability**
- **Validity**
- **Implication**
Deduction and Proof

Satisfiability, Validity, and Resolution
What use is a Spec?

“Could it be true?” “Must it be true?” “If this, then that?”

Satisfiability  Validity  Implication

What evidence could one provide?

“Could \((x \land y) \lor (a \land b)\) be true?”  \textcolor{green}{Yes.}

“Try True for \(x\) and \(y\), and False for \(a\) and \(b\).”

“Must \((x \land y) \lor (a \land b)\) be true?”  \textcolor{red}{No.}

“Try False for \(x\) and \(y\), and False for \(a\) and \(b\).”
What use is a Spec?

“Could it be true?” “Must it be true?” “If this, then that?”

Satisfiability  Validity  Implication

What evidence could one provide?

“If \((x \land y) \lor (a \land b)\) is true, then must \(x \lor \neg a\) be true?”

“If \((x \land y) \lor (a \land b)\) is true, then could \(x \lor \neg a\) be false?”

“Could \(( (x \land y) \lor (a \land b) ) \land \neg (x \lor \neg a)\) be true?” Yes.

“Try True for \(x\) and \(y\), and False for \(a\) and \(b\).”
Evidence

"Must \((x \lor y) \land (\neg x \lor z) \land (\neg y \lor z) \land \neg z\) be false?" Yes.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Evidence

"Must \((x \lor y) \land (\neg x \lor z) \land (\neg y \lor z) \land \neg z\) be false?" Yes.

If \((x \lor y)\) and \((\neg x \lor z)\) then \((y \lor z)\)

\[
\frac{x \lor y \quad \neg x \lor z}{y \lor z}
\]

Logical Resolution
Evidence

Logical resolution can prove anything must be false.

Or disprove anything could be true, naturally...

Compact: resolution proof hard to find, easy to check
Proof by Resolution

Logical resolution can prove anything must be false.

1. Put into conjunctive form
2. Group “or” terms: with $x$ and with $\neg x$
3. Resolve every pair of terms across groups
4. New, equivalent formula without $x$

We’ll see “variable elimination” again in this class.
Course Updates

Recitation, Textbook, Assignment
Textbook

The Calculus of Computation
Aaron R. Bradley
Zohar Manna
Springer, 2007

Topic 1 readings from this book (today: Chapter 1)

Available electronically through SpringerLink

I prefer to read day after lecture, before assignment

That way you can skim if you already understand it
Assignment 1

Solving n-Queens using the miniSat solver

Given a board size, find all valid n-Queens solutions

Encode the problem to boolean logic

Transform the encoding into conjunctive form

Invoke the solver and parse the output

Repeat multiple times to find all solutions

Due 16 January, in a week (start now!)

Demo, install help tomorrow at 13:00 in MEB 3485
Proof Search

Davis-Putnam-Logemann-Loveland
Proof by Resolution

Logical resolution can **prove anything must be false**.

1. Put into conjunctive form

2. Group “or” terms: with $x$ and with $\neg x$

3. Resolve every pair of terms across groups

4. New, equivalent formula without $x$

**Rewrites one fact into another form**
Proof by Resolution

Algorithm with one variable: fact being rewritten

Like working in assembly language…

Let’s re-imagine this algorithm

Result is known as DPLL “Davis-Putnam-Logemann-Loveland”

Let’s re-imagine this algorithm
Special Cases

Logical resolution can prove anything must be false.

1. Put into conjunctive form

2. Group “or” terms: with $x$ and with $\neg x$

Then there’s only $x$ and no $\neg x$

So $x$ should be True

So terms with $x$ don’t matter
Logical resolution can prove anything must be false.

1. Put into conjunctive form

2. What if one term is a singleton?

3. Resolve every pair of terms across groups

4. Then $x$ must be True
Logical resolution can **prove anything must be false.**

1. Put into conjunctive form

2. **Otherwise, both terms are “or”s**

3. Resolve every pair of terms across groups

4. **Then result is also an “or”**
DPLL algorithm

Proof by resolution, presented as a program

1. Put into conjunctive form
2. Check for variables all alone
3. Check for variables with one polarity
4. Pick a variable and try setting to True
5. If it doesn’t work, it must be False
Proof by Resolution

DPLL algorithm is still **core of modern SAT solvers**

Two **additional improvements** possible
- Non-chronological backtracking
- Conflict-driven clausal learning

One **knob to tune**: which variable to pick
- Most common variable? Least common? Least controversial?
Next class:
First-order Logic

To do:
- Course feedback
- Reading in textbook
- Assignment 1
Boolean logic

Language for stating facts about booleans
  And/or/not, conjunctive form, universality

Proofs of boolean logic facts by refutation
  Compact evidence of truth/falsity

Algorithm to automatically find proofs
  Efficient in simple cases, core of modern solvers
TO INFINITY...

2 BOOLEANs

∞ INTEGERs
NEW SYNTAX

FOR ALL

THERE EXISTS
Theories

Only the limits of your mind make it impossible.
Next class:
First-order Logic

To do:
☐ Course feedback
☐ Reading in textbook
☐ Assignment 1