Part 1: Propositional Calculus (b)

COM2107 Logic in Computer Science

Slides by Georg Struth
Adapted by Mike Stannett and Jonni Virtema

Ver. 1.1

School of Computer Science
Session code: XX-XX-XX

Main Themes

We'll be looking at

- Derived rules
- Classical rules
- Structural induction principle for proving things about Prop
- Semantics of Prop
- Soundness, completeness and compactness
- An example of logical modelling (Sudoku)

Lecture notes

Reading: 2.5–2.8, 2.10

Derived Rules

Proofs are rarely done from scratch. In practice we use derived rules as shortcuts and/or for information hiding.

Examples

$$\frac{}{\varphi \to \varphi} \qquad \frac{\varphi \to \psi \qquad \psi \to \gamma}{\varphi \to \gamma}$$

$$\frac{\varphi \to \psi}{(\varphi \land \gamma) \to (\psi \land \gamma)} \qquad \frac{\varphi \to \psi}{(\varphi \lor \gamma) \to (\psi \lor \gamma)}$$

see lecture notes for proofs

Derived Rules

some derived rules are important enough to have names

double-negation introduction

$$\frac{\varphi}{\neg\neg\varphi}\ (\neg\neg I)$$

transposition

$$\frac{\varphi \to \psi}{\neg\psi \to \neg\varphi}\ (\mathrm{tp})$$

modus tollens

$$\frac{\varphi \to \psi \qquad \neg\psi}{\neg\varphi}\ (\mathrm{mt})$$

see lecture notes for proofs

How is $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$ proved?

1.	$\varphi \to \psi$	hyp
2.		

How is $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$ proved?

1.	$\varphi \to \psi$	hyp
2.	$\neg\psi$	hyp
3.	D	
4.	$\neg\varphi$	
5.	$\neg\psi \to \neg\varphi$	$\to I$

How is $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$ proved?

$$\dfrac{\begin{array}{c}[\varphi]\\ \vdots \\ \bot\end{array}}{\neg\varphi}\ \neg I$$

1.	$\varphi \to \psi$	hyp
2.	$\neg\psi$	hyp
3.	D	
4.	$\neg\varphi$	
5.	$\neg\psi \to \neg\varphi$	$\to I$

How is $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$ proved?

$$\dfrac{\begin{array}{c}[\varphi]\\ \vdots \\ \bot\end{array}}{\neg\varphi}\ \neg I$$

1.	$\varphi \to \psi$	hyp
2.	$\neg\psi$	hyp
3.	φ	hyp
4.	ψ	$\to E, 1, 3$
5.	$\bot$	$\neg E, 2, 4$
6.	$\neg\varphi$	
7.	$\neg\psi \to \neg\varphi$	$\to I$

How is $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$ proved?

$$\dfrac{\begin{array}{c}[\varphi]\\ \vdots \\ \bot\end{array}}{\neg\varphi}\ \neg I$$

1.	$\varphi \to \psi$	hyp
2.	$\neg\psi$	hyp
3.	φ	hyp
4.	ψ	$\to E, 1, 3$
5.	$\bot$	$\neg E, 2, 4$
6.	$\neg\varphi$	$\neg I, 3-5$
7.	$\neg\psi \to \neg\varphi$	$\to I, 2-6$

Classical Rules

many computer scientists like constructive propositional logic

many mathematicians prefer classical logic

this requires an additional inference rule

Classical Rules

$$\frac{}{\varphi \vee \neg\varphi}\,(\mathit{lem}) \qquad \begin{array}{c}[\neg\varphi]\\ \vdots \\ \dfrac{\bot}{\varphi}\,(\mathit{pbc})\end{array} \qquad \frac{\neg\neg\varphi}{\varphi}\,(\neg\neg E)$$

These are

- (lem): law of the excluded middle;
- (pbc): proof by contradiction;
- ($\neg\neg E$): double-negation elimination.

Any one of them is enough — the others are then derivable.

Law of the excluded middle

Classical rules are problematic for constructivists. For example, $\vdash \varphi \vee \neg\varphi$

- says that every proposition is either true or false – there is no "middle" option.
- to prove it, you should either provide a proof of φ or a proof of $\neg\varphi$

Examples (This isn't always possible!)

Given an arbitrary program P, we can't necessarily determine whether P will or won't eventually halt. In this situation, if we put $\varphi :=$ "P halts", we can't prove φ and we can't prove $\neg\varphi$ either.

NOTE: We're only saying that *some* φ's are problematic, not *all* of them! For example, if $\varphi := \mathit{isEven}(7)$, then $\vdash \varphi \vee \neg\varphi$ is easily provable.

Example of a classical (non-constructive) proof

Claim: There are irrational numbers m and n for which m^n is rational.

Proof (by cases): We know that $\sqrt{2}$ is irrational. Let $x = \sqrt{2}^{\sqrt{2}}$. According to (lem), x is either rational or irrational.

- if x rational, the claim holds for $m = n = \sqrt{2}$ because then $m^n = x$.
- if x is irrational, the claim holds for $m = x$ and $n = \sqrt{2}$, because then $m^n = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\sqrt{2}} = \sqrt{2}^2 = 2$.

So, as long as you accept (lem), the claim must be true.

Nonetheless, the proof doesn't actually tell us which choice of m, n makes the claim hold. It doesn't construct definite witnesses to the claim.

Classical Connectives

The connectives of classical propositional logic are no longer independent. For example, we can prove

$$\varphi \vee \psi \dashv\vdash \neg(\neg\varphi \wedge \neg\psi) \qquad \varphi \to \psi \dashv\vdash \neg\varphi \vee \psi$$

One can start from small subset and define the rest explicitly

$$\{\neg, \wedge\} \qquad \{\bot, \to\} \qquad \{\neg, \wedge, \vee\}$$

and the inference rules for defined connectives become derivable.

We'll see later that the set $\{\neg, \wedge, \vee\}$ is important for automated proof search.

Classical Theorems

lecture notes contain long list of classical theorems

proving them using natural deduction is good practice

We'll also see later how to prove some of them using an interactive proof assistant.

Syntax (revisited) and Semantics

There are many versions of logic.

In general, a logic consists roughly of

- a syntax/language that tells us what we can say
- a deductive system that tells us what we can show
- a semantics that tells us what this all means

proofs are syntactic objects, too

deductive systems are often considered as syntax

Object Language and Metalanguage

We have already seen the underlying language of propositional logic. It includes propositional variables and connectives. We have also been talking about propositional calculus using English.

English is a metalanguage used for discussing the object language of Prop.

A metalanguage is a language in which the object language of a logic is explained.

Ours is English. There are many others we could have chosen, both natural and formal.

Object Language and Metalanguage

distinguishing object/metalanguage can be tedious

quotes can be used for lifting object language to meta-level, e.g. "'Bird' is a word"

we generally use metavariables (schematic variables) $\varphi, \psi, \gamma, \ldots$ to stand for arbitrary formulas in the object language (logical formulas), and we write $\Rightarrow$ and $\Leftrightarrow$ in metalanguage proofs for "implies".

Examples

$$\varphi \vdash \psi \implies \vdash \varphi \to \psi$$

Alphabet

Every language starts with an alphabet. The alphabet of Prop consists of the following symbols:

- propositional variables $p, q, r, s, \ldots$ from some (countably infinite) set P
- propositional connectives $\bot$, $\top$, $\neg$, $\vee$, $\wedge$, $\rightarrow$
- auxiliary symbols: (,), ...

propositional variables form basic building blocks for formulas

$\bot$ and $\top$ are nullary connectives; $\neg$ is unary; $\vee$, $\wedge$, $\rightarrow$ are binary

arity of connective indicates number of parameters it can take

brackets make formulas unambiguous

Formulas

set Φ of propositional formulas is defined recursively by grammar:

$$\Phi ::= \bot \mid \top \mid p \mid (\neg\Phi) \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi) \mid (\Phi \to \Phi), \qquad \text{where } p \in P$$

propositional variables $p \in P$, $\bot$, $\top$ are atomic formulas

all others are composite

Precedence

we use precedence to save brackets:

$\neg$ binds more strongly than $\wedge$, $\vee$, $\rightarrow$

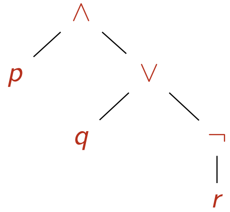
$\wedge$, $\vee$ have equal precedence and bind more strongly than $\rightarrow$

Abstract Syntax Trees

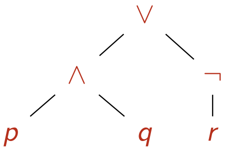
grammar for formulas is tree data type

for every formula we can construct an abstract syntax tree (ast)

asts of $(p \wedge (q \vee (\neg r)))$ and $((p \wedge q) \vee (\neg r)))$ are



and



string $p \wedge q \vee \neg r$ doesn't have an ast: it isn't a formula (without precedence!)

Subformulas

Recursive data types admit recursive definitions; many properties of propositional formulas can be defined that way

Examples

The recursive function $\mathsf{Sf} : \Phi \to \mathcal{P}(\Phi)$ defined, for all $\varphi, \psi \in \Phi$, by

$$\begin{aligned} \mathsf{Sf}(\varphi) &= \{\varphi\} && \text{if } \varphi \text{ atomic} \\ \mathsf{Sf}(\neg\varphi) &= \{\neg\varphi\} \cup \mathsf{Sf}(\varphi) && \\ \mathsf{Sf}(\varphi \diamond \psi) &= \{\varphi \diamond \psi\} \cup \mathsf{Sf}(\varphi) \cup \mathsf{Sf}(\psi) && \text{if } \diamond \in \{\wedge, \vee, \to\} \end{aligned}$$

computes the set of all subformulas of φ.

We say that ψ is a subformula of φ iff $\psi \in \mathsf{Sf}(\varphi)$

Subformulas

example:

$$\mathsf{Sf}(p \wedge (q \vee \neg r)) = \{p \wedge (q \vee \neg r)\} \cup \mathsf{Sf}(p) \cup \mathsf{Sf}(q \vee \neg r)$$

Subformulas

example:

$$\begin{aligned}
\mathsf{Sf}(p \wedge (q \vee \neg r)) &= \{p \wedge (q \vee \neg r)\} \cup \mathsf{Sf}(p) \cup \mathsf{Sf}(q \vee \neg r) \\
&= \{p \wedge (q \vee \neg r)\} \cup \{p\} \cup \{q \vee \neg r\} \cup \mathsf{Sf}(q) \cup \mathsf{Sf}(\neg r))
\end{aligned}$$

Subformulas

example:

$$
\begin{aligned}
\mathrm{Sf}(p \wedge (q \vee \neg r)) &= \{p \wedge (q \vee \neg r)\} \cup \mathrm{Sf}(p) \cup \mathrm{Sf}(q \vee \neg r) \\
&= \{p \wedge (q \vee \neg r)\} \cup \{p\} \cup \{q \vee \neg r\} \cup \mathrm{Sf}(q) \cup \mathrm{Sf}(\neg r)) \\
&= \{p \wedge (q \vee \neg r), p, q \vee \neg r\} \cup \{q\} \cup \{\neg r\} \cup \mathrm{Sf}(r)
\end{aligned}
$$

Subformulas

example:

$$\begin{aligned}
\mathrm{Sf}(p \wedge (q \vee \neg r)) &= \{p \wedge (q \vee \neg r)\} \cup \mathrm{Sf}(p) \cup \mathrm{Sf}(q \vee \neg r) \\
&= \{p \wedge (q \vee \neg r)\} \cup \{p\} \cup \{q \vee \neg r\} \cup \mathrm{Sf}(q) \cup \mathrm{Sf}(\neg r)) \\
&= \{p \wedge (q \vee \neg r), p, q \vee \neg r\} \cup \{q\} \cup \{\neg r\} \cup \mathrm{Sf}(r) \\
&= \{p \wedge (q \vee \neg r), p, q \vee \neg r, q, \neg r\} \cup \{r\}
\end{aligned}$$

Subformulas

example:

$$
\begin{aligned}
\mathrm{Sf}(p \wedge (q \vee \neg r)) &= \{p \wedge (q \vee \neg r)\} \cup \mathrm{Sf}(p) \cup \mathrm{Sf}(q \vee \neg r) \\
&= \{p \wedge (q \vee \neg r)\} \cup \{p\} \cup \{q \vee \neg r\} \cup \mathrm{Sf}(q) \cup \mathrm{Sf}(\neg r)) \\
&= \{p \wedge (q \vee \neg r), p, q \vee \neg r\} \cup \{q\} \cup \{\neg r\} \cup \mathrm{Sf}(r) \\
&= \{p \wedge (q \vee \neg r), p, q \vee \neg r, q, \neg r\} \cup \{r\} \\
&= \{p \wedge (q \vee \neg r), p, q \vee \neg r, q, \neg r, r\}
\end{aligned}
$$

Example: Variables of a Formula

The recursive function $V : \Phi \to \mathcal{P}(P)$ defined by

$$
\begin{aligned}
V(\varphi) &= \emptyset & &\text{if } \varphi \in \{\bot, \top\} \\
V(\varphi) &= \{\varphi\} & &\text{if } \varphi \in P \\
V(\neg\varphi) &= V(\varphi) & & \\
V(\varphi \diamond \psi) &= V(\varphi) \cup V(\psi) & &\text{if } \diamond \in \{\wedge, \vee, \to\}
\end{aligned}
$$

computes the set of propositional variables that occur in φ

We say that p occurs in φ iff $p \in V(\varphi)$

Proofs About Propositional Formulas

Proofs about recursive data types typically use structural induction.

Principle of Structural Induction for Prop:
To prove that a claim $Cl(\varphi)$ holds for all propositions φ, it is enough to do all of the following:

- base cases:
  Prove that Cl holds for $\top$, $\bot$ and for an arbitrary propositional variable p;
- step case (unary):
  Prove that, if Cl holds for an arbitrary proposition ψ, then it also holds for $\neg\psi$;
- step cases (binary):
  Prove that, if Cl holds for arbitrary propositions ψ and γ, then it also holds for $(\psi \wedge \gamma)$, $(\psi \vee \gamma)$ and $(\psi \rightarrow \gamma)$.

Example of a structural induction proof over Prop

Claim: $V(\varphi) = \mathrm{Sf}(\varphi) \cap P$ for all φ

base cases

$Cl(\bot)$ holds: if $\varphi = \bot$, then, because $\bot \notin P$,

$$V(\varphi) = \emptyset = \{\bot\} \cap P = \mathrm{Sf}(\varphi) \cap P$$

$Cl(\top)$ holds: if $\varphi = \top$, then there's a similar proof.

$Cl(\varphi)$ holds if φ is a propositional variable p:
In this case

$$V(\varphi) = V(p) = \{p\} = \{p\} \cap P = \mathrm{Sf}(\varphi) \cap P$$

Proofs About Propositional Formulas

induction step (unary):

if $\varphi = \neg\psi$, then $V(\psi) = \mathsf{Sf}(\psi) \cap P$ by induction hypothesis and

$$\begin{aligned} V(\neg\psi) &= V(\psi) \\ &= \mathsf{Sf}(\psi) \cap P \\ &= (\{\neg\psi\} \cap P) \cup (\mathsf{Sf}(\psi) \cap P) \\ &= (\{\neg\psi\} \cup \mathsf{Sf}(\psi)) \cap P \\ &= \mathsf{Sf}(\neg\psi) \cap P \end{aligned}$$

because $\{\neg\psi\} \cap P = \emptyset$

Proofs About Propositional Formulas

induction steps (binary): for any $\diamond \in \{\wedge, \vee, \rightarrow\}$

if $\varphi = \psi \diamond \gamma$, then

$$V(\psi) = \mathsf{Sf}(\psi) \cap P \qquad \text{and} \qquad V(\gamma) = \mathsf{Sf}(\gamma) \cap P$$

by induction hypothesis and

$$\begin{aligned} V(\psi \diamond \gamma) &= V(\psi) \cup V(\gamma) \\ &= (\mathsf{Sf}(\psi) \cap P) \cup (\mathsf{Sf}(\gamma) \cap P) \\ &= (\{\psi \diamond \gamma\} \cap P) \cup (\mathsf{Sf}(\psi) \cap P) \cup (\mathsf{Sf}(\gamma) \cap P) \\ &= (\{\psi \diamond \gamma\} \cup \mathsf{Sf}(\psi) \cup \mathsf{Sf}(\gamma)) \cap P \\ &= \mathsf{Sf}(\psi \diamond \gamma) \cap P \end{aligned}$$

because $\{\psi \diamond \gamma\} \cap P = \emptyset$

Semantics of Classical Propositional Logic

The classical meaning of a proposition φ is its truth value: true/false.

meaning of composite formulas depends only on that of their top propositional connective and immediate subformulas

meaning of propositional formula is thus computed recursively

Truth Tables

this recursive function is captured implicitly by truth tables

we write 1 for "true" and 0 for "false" and call $\mathbb{B} = \{0, 1\}$ the Booleans

φ	$\neg\varphi$
1	0
0	1

φ	ψ	$\varphi \wedge \psi$
1	1	1
1	0	0
0	1	0
0	0	0

φ	ψ	$\varphi \vee \psi$
1	1	1
1	0	1
0	1	1
0	0	0

φ	ψ	$\varphi \to \psi$
1	1	1
1	0	0
0	1	1
0	0	1

Assignments and Valuations

We define valuations using recursive functions over propositions:

An assignment is any function $v : P \to \mathbb{B}$ telling us which propositional variables should be assigned the value true and which should be assigned the value false.

For any v, the corresponding valuation $[\![-]\!]_v : \Phi \to \mathbb{B}$ is defined by

$$
\begin{aligned}
[\![\top]\!]_v &= 1 \\
[\![\bot]\!]_v &= 0 \\
[\![p]\!]_v &= v(p) && \text{for } p \in P \\
[\![\neg\varphi]\!]_v &= 1 - [\![\varphi]\!]_v \\
[\![\varphi \wedge \psi]\!]_v &= \mathtt{min}([\![\varphi]\!]_v, [\![\psi]\!]_v) \\
[\![\varphi \vee \psi]\!]_v &= \mathtt{max}([\![\varphi]\!]_v, [\![\psi]\!]_v) \\
[\![\varphi \to \psi]\!]_v &= \mathtt{max}(1 - [\![\varphi]\!]_v, [\![\psi]\!]_v)
\end{aligned}
$$

Assignments and Valuations

valuations are determined by the truth values of variables occurring in the formula

fact: if $\varphi \in \Phi$ and v, w are assignments, then

$$\big(v(p) = w(p),\ \text{for all } p \in V(\varphi)\big) \ \Rightarrow\ [\![\varphi]\!]_v = [\![\varphi]\!]_w$$

the proof uses structural induction

we write $[\![\varphi]\!]$ when assignments don't matter

Assignments and Valuations

The following facts are useful for reasoning with valuations

$$
\begin{aligned}
[\![\neg\varphi]\!] = 1 &\quad\Leftrightarrow\quad [\![\varphi]\!] = 0 \\
[\![\varphi \wedge \psi]\!] = 1 &\quad\Leftrightarrow\quad [\![\varphi]\!] = 1 \text{ and } [\![\psi]\!] = 1 \\
[\![\varphi \vee \psi]\!] = 1 &\quad\Leftrightarrow\quad [\![\varphi]\!] = 1 \text{ or } [\![\psi]\!] = 1 \\
[\![\varphi \rightarrow \psi]\!] = 1 &\quad\Leftrightarrow\quad [\![\varphi]\!] \leq [\![\psi]\!]
\end{aligned}
$$

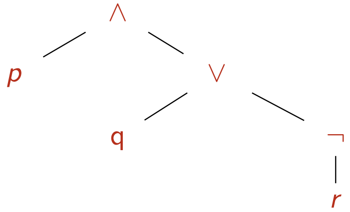
Assignments and Valuations

Example: for $v : p \mapsto 1, q \mapsto 0, r \mapsto 0$,

$$\begin{aligned}
[\![p \wedge (q \vee \neg r)]\!]_v &= \mathtt{min}([\![p]\!]_v, [\![q \vee \neg r]\!]_v) \\
&= \mathtt{min}(v(p), \mathtt{max}([\![q]\!]_v, [\![\neg r]\!]_v)) \\
&= \mathtt{min}(1, \mathtt{max}(v(q), 1 - [\![r]\!]_v)) \\
&= \mathtt{max}(0, 1 - v(r))) \\
&= 1 - 0 \\
&= 1
\end{aligned}$$

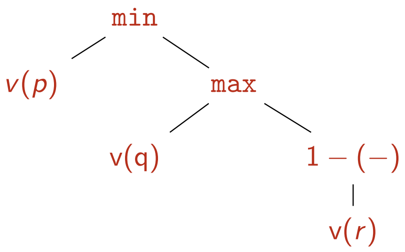
Assignments and Valuations

evaluation can be visualised using ast of $p \wedge (q \vee \neg r)$



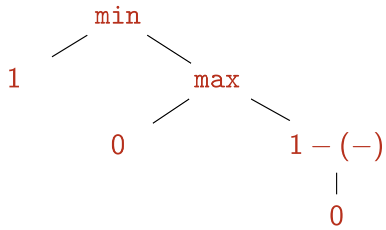
Assignments and Valuations

evaluation can be visualised using ast of $p \wedge (q \vee \neg r)$



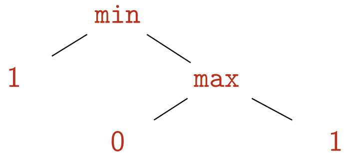
Assignments and Valuations

evaluation can be visualised using ast of $p \wedge (q \vee \neg r)$



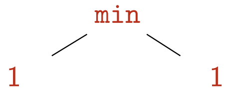
Assignments and Valuations

evaluation can be visualised using ast of $p \wedge (q \vee \neg r)$



Assignments and Valuations

evaluation can be visualised using ast of $p \wedge (q \vee \neg r)$



Assignments and Valuations

evaluation can be visualised using ast of $p \wedge (q \vee \neg r)$

$$1$$

Tautologies, Entailment, Equivalence

$\varphi \in \Phi$ is a tautology (or valid), written $\models \varphi$,
if $[\![\varphi]\!]_v = 1$ for all v

$\Gamma \subseteq \Phi$ (semantically) entails $\varphi \in \Phi$, written $\Gamma \models \varphi$,
if $\mathtt{min}\{[\![\psi]\!]_v \mid \psi \in \Gamma\} = 1$ implies $[\![\varphi]\!]_v = 1$ for all v

$\varphi, \psi \in \Phi$ are logically equivalent, written $\varphi \equiv \psi$,
if $[\![\varphi]\!]_v = [\![\psi]\!]_v$ for all v

Satisfiability

- A formula φ is satisfiable
  if $[\![\varphi]\!]_v = 1$ for some v
- φ is unsatisfiable (a contradiction)
  if it is not satisfiable
- $\Gamma \subseteq \Phi$ is satisfiable
  if $\mathtt{min}\{[\![\psi]\!]_v \mid \psi \in \Gamma\} = 1$ for some v
- $\Gamma \subseteq \Phi$ is unsatisfiable
  if Γ is not satisfiable

Fact: φ is valid if and only if $\neg\varphi$ is unsatisfiable

SAT Solvers

SAT problem: Given φ decide whether $[\![\varphi]\!]_v = 1$ for some v

solution is an algorithm that takes any propositional formula as an input, and returns “yes” if it is satisfiable and “no” otherwise

the problem is NP-complete (no polynomial time algorithm in $|V(\varphi)|$ is known)

yet SAT-solvers can handle problems with $>10k$ variables

Soundness and Completeness

two important properties of any logic are

soundness: $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$

completeness: $\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$

soundness guarantees that one can't deduce falsities from true hypotheses

completeness that one can deduce all consequences of true hypotheses

Prop is sound and complete!

You can read the proof of soundness and completeness from the lecture notes, and extra slides.

Going through the details in a lecture would take an hour!

Theorem:

For any set of propositional formulae Γ and any propositional formula φ

$$\Gamma \vdash \varphi \Leftrightarrow \Gamma \models \varphi$$

A few final facts about Prop

The compactness theorem

A set Γ is satisfiable if and only if every finite subset of Γ is satisfiable.

Proof:

- if Γ is satisfiable, so is every subset
- conversely,

$$
\begin{aligned}
\Gamma \text{ unsatisfiable } &\Rightarrow \Gamma \models \bot \\
&\Rightarrow \Gamma \vdash \bot \qquad \text{(by completeness)} \\
&\Rightarrow \Gamma_0 \vdash \bot \text{ for some finite } \Gamma_0 \subseteq \Gamma \qquad \text{(since every proof is finite)} \\
&\Rightarrow \Gamma_0 \models \bot \text{ for some finite } \Gamma_0 \subseteq \Gamma \qquad \text{(by soundness)} \\
&\Rightarrow \text{ some finite } \Gamma_0 \subseteq \Gamma \text{ is unsatisfiable}
\end{aligned}
$$

Alternatively: a set is unsatisfiable iff some finite subset is.

A Modelling Example

many computing systems can be modelled using propositional logic

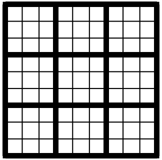
this may surprise given its limited expressivity

many real-world examples are too complex to be discussed here

so we look at somewhat artificial example

others can be found in lecture notes

Sudoku



Sudoku is based on 9×9 grid divided into nine disjoint 3×3 regions, with an initial labelling of some cells

Each remaining cell (i, j) needs to be labelled with an integer in $[1, 9]$ such that each row, column and region contains each label precisely once

Sudoku

notation:

- we write $N = \{1, \ldots, 9\}$
- variable λ_{ij}^{k} indicates that cell (i, j) is labelled with $k \in N$
- $\bigwedge_{i \in N} P_i$ means $P_1 \wedge \cdots \wedge P_9$ and $\bigvee_{i \in N} P_i$ means $P_1 \vee \cdots \vee P_9$

examples:

- $\bigwedge_{i \in N} \lambda_{ij}^{k}$ means every element in column j has label k
- $\bigvee_{j \in N} \lambda_{ij}^{k}$ means some element in row i has label k

Sudoku Constraints

no cell contains more than one number

$$C_1 = \bigwedge_{i,j,k_1,k_2 \in N, k_1 < k_2} \neg\left(\lambda_{ij}^{k_1} \wedge \lambda_{ij}^{k_2}\right)$$

every row/column contains every number

$$C_2 = \bigwedge_{i,k \in N} \bigvee_{j \in N} \lambda_{ij}^{k} \qquad C_3 = \bigwedge_{j,k \in N} \bigvee_{i \in N} \lambda_{ij}^{k}$$

Sudoku Constraints

every region contains every number

$$C_4 = \bigwedge_{k \in N} \bigwedge_{l,m \in \{0,1,2\}} \bigvee_{i,j \in \{1,2,3\}} \lambda^k_{(3l+i)(3m+j)}$$

initial configuration C_5 is encoded as conjunction of certain λ^k_{ij}'s

Solving Sudoku

puzzle has solution iff $C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5$ satisfiable

i.e. some assignment v for the $9^3 = 729$ variables λ_{ij}^k yields

$$[\![C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5]\!]_v = 1$$

each C_i must be translated into proper propositional formula which is too large and boring to be shown

resulting truth table has 2^{729} rows — but modern SAT-solvers solve Sudoku very quickly

Summary of Part 1: Propositional Calculus

- Propositional calculus (Prop) underpins the logics we use in everyday mathematics and science;
- Natural deduction rules tell us how to deduce things from axioms;
- Constructivists only accept argument supported by witnesses/evidence;
- Classical logicians accept things like (lem) and (pbc);
- Classical propositional logic is both sound and complete;
- SAT-solving can be use to solve problems.
  It is hard in theory but can be easy in practice.

Summary of Part 1: Propositional Calculus

- Propositional calculus (Prop) underpins the logics we use in everyday mathematics and science;
- Natural deduction rules tell us how to deduce things from axioms;
- Constructivists only accept argument supported by witnesses/evidence;
- Classical logicians accept things like (lem) and (pbc);
- Classical propositional logic is both sound and complete;
- SAT-solving can be use to solve problems.
  It is hard in theory but can be easy in practice.

Next Time

We shift to Predicate Calculus. It is both more powerful, and more surprising, than propositional logic.