CS 228 : Logic in Computer Science

Krishna. S

Some Real Life Stories

Therac-25(1987)

- The Therac-25 : radiation therapy machine produced by Atomic Energy of Canada Limited (AECL)

Therac-25(1987)

- The Therac-25 : radiation therapy machine produced by Atomic Energy of Canada Limited (AECL)
- Involved in at least six accidents, in which patients were given massive overdoses of radiation, approximately 100 times the intended dose.

Therac-25(1987)

- The Therac-25 : radiation therapy machine produced by Atomic Energy of Canada Limited (AECL)
- Involved in at least six accidents, in which patients were given massive overdoses of radiation, approximately 100 times the intended dose.
- Design error in the control software (race condition)

Intel Pentium Bug (1994)

- The Intel FDIV bug : Bug in the intel P5 floating point unit

Intel Pentium Bug (1994)

- The Intel FDIV bug : Bug in the intel P5 floating point unit
- Discovered by a professor working on Brun's constant
- $(\frac{1}{3}+\frac{1}{5})+(\frac{1}{5}+\frac{1}{7})+(\frac{1}{11}+\frac{1}{13})+(\frac{1}{17}+\frac{1}{19})+\ldots$ converges to $B \cong 1.90216054$

Intel Pentium Bug (1994)

- The Intel FDIV bug : Bug in the intel P5 floating point unit
- Discovered by a professor working on Brun's constant
- $(\frac{1}{3}+\frac{1}{5})+(\frac{1}{5}+\frac{1}{7})+(\frac{1}{11}+\frac{1}{13})+(\frac{1}{17}+\frac{1}{19})+\ldots$ converges to $B \cong 1.90216054$
- Intel offered to replace all flawed processors

Ariane 5 (1996)

- ESA (European Space Agency) Ariane 5 Launcher

Ariane 5 (1996)

- ESA (European Space Agency) Ariane 5 Launcher
  - Shown here in maiden flight on 4th June 1996

Ariane 5 (1996)

- ESA (European Space Agency) Ariane 5 Launcher
  - Shown here in maiden flight on 4th June 1996
- Self destructs 37 secs later

Ariane 5 (1996)

- ESA (European Space Agency) Ariane 5 Launcher
  - Shown here in maiden flight on 4th June 1996
- Self destructs 37 secs later
  - uncaught exception: data conversion from 64-bit float to 16-bit signed int

Toyota Prius (2010)

- First mass produced hybrid vehicle

Toyota Prius (2010)

- First mass produced hybrid vehicle
  - software “glitch” found in anti-lock braking system
  - Eventually fixed via software update in total 185,000 cars recalled, at huge cost

Nest Thermostat (2016)

- Nest Thermostat, the smart, learning thermostat from Nest Labs

Nest Thermostat (2016)

- Nest Thermostat, the smart, learning thermostat from Nest Labs
  - software "glitch" led several homes to a frozen state, reported in NY times, Jan 13, 2016. May be, old fashioned mechanical thermostats better!

What do these stories have in common?

- Programmable computing devices
    - conventional computers and networks
    - software embedded in devices

What do these stories have in common?

- Programmable computing devices
  - conventional computers and networks
  - software embedded in devices
- Programming error direct cause of failure
- Software critical
  - for safety
  - for business
  - for performance

What do these stories have in common?

- Programmable computing devices
  - conventional computers and networks
  - software embedded in devices
- Programming error direct cause of failure
- Software critical
  - for safety
  - for business
  - for performance
- High costs incurred: financial, loss of life
- Failures avoidable

Formal Methods

Intuitive Description

"Applied Mathematics for modelling and analysing ICT systems"

Formal methods offer a large potential for:

Formal Methods

Intuitive Description

"Applied Mathematics for modelling and analysing ICT systems"

Formal methods offer a large potential for:

- obtaining an early integration of verification in the design process

Formal Methods

Intuitive Description

"Applied Mathematics for modelling and analysing ICT systems"

Formal methods offer a large potential for:

- obtaining an early integration of verification in the design process
- providing more effective verification techniques (higher coverage)

Formal Methods

Intuitive Description

“Applied Mathematics for modelling and analysing ICT systems”

Formal methods offer a large potential for:

- obtaining an early integration of verification in the design process
- providing more effective verification techniques (higher coverage)
- reducing the verification time

Simulation and Testing

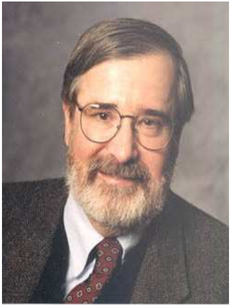
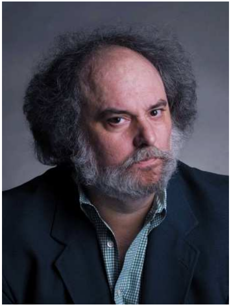
Basic procedure

- Take a model
- Simulate it with certain inputs
- Observe what happens, and if this is desired

Important Drawbacks

- possible behaviours very large/infinite
- unexplored behaviours may contain fatal bug
- can show presence of errors, not their absence

Model Checking

- Year 2008 : ACM confers the Turing Award to the pioneers of Model Checking: Ed Clarke, Allen Emerson, and Joseph Sifakis
- Why?

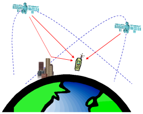
Model checking

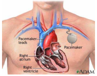
- Model checking has evolved in last 25 years into a widely used verification and debugging technique for software and hardware.
- Cost of *not* doing formal verification is high!
  - The France Telecom example
  - Ariane rocket: kaboom due to integer overflow!
  - Toyota/Ford recalls

Model checking

- Model checking has evolved in last 25 years into a widely used verification and debugging technique for software and hardware.
- Cost of *not* doing formal verification is high!
  - The France Telecom example
  - Ariane rocket: kaboom due to integer overflow!
  - Toyota/Ford recalls
- Model checking used (and further developed) by companies/institutes such as IBM, Intel, NASA, Cadence, Microsoft, and Siemens, and has culminated in many freely downloadable software tools that allow automated verification.

What is Model Checking?





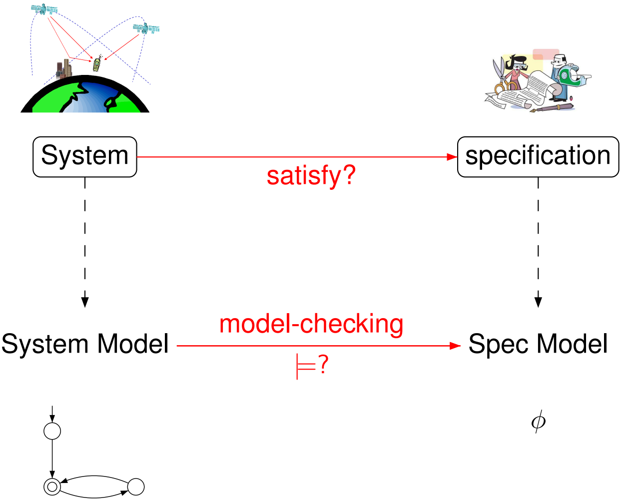
System

satisfy?

specification

good/bad properties

What is Model Checking?



Model Checker as a Black Box

- Inputs to Model checker : A finite state system M, and a property P to be checked.
- Question : Does M satisfy P?
- Possible Outputs
  - Yes, M satisfies P
  - No, here is a counter example!.

What are Models?

Transition Systems

- States labeled with propositions
- Transition relation between states
- Action-labeled transitions to facilitate composition

What are Models?

Transition Systems

- States labeled with propositions
- Transition relation between states
- Action-labeled transitions to facilitate composition

Expressivity

- Programs are transition systems
- Multi-threading programs are transition systems
- Communicating processes are transition systems
- Hardware circuits are transition systems
- What else?

What are Properties?

Example properties

- Can the system reach a deadlock?
- Can two processes ever be together in a critical section?
- On termination, does a program provide correct output?

What are Properties?

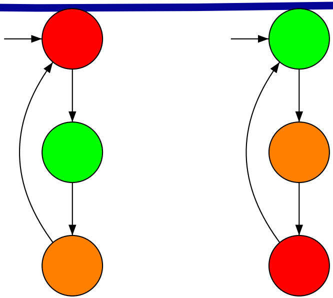
Example properties

- Can the system reach a deadlock?
- Can two processes ever be together in a critical section?
- On termination, does a program provide correct output?

Logics of Relevance

- Classical Logics
  - First Order Logic
  - Monadic Second Order Logic
- Temporal Logics
  - Propositional Logic, enriched with modal operators such as $\Box$ (always) and $\Diamond$ (eventually)
  - Interpreted over state sequences (linear)
  - Or over infinite trees (branching)

Two Traffic Lights



1. The traffic lights are never green simultaneously
   $\forall x(\neg(green_1(x) \land green_2(x)))$ or $\Box(\neg(green_1 \land green_2))$
2. The first traffic light is infinitely often green
   $\forall x \exists y(x < y \land green_1(y))$ or $\Box\Diamond green_1$
3. Between every two occurrences of traffic light 1 becoming red, traffic light 2 becomes red once.

The Model Checking Process

- Modeling Phase
  - model the system under consideration
  - as a first sanity check, perform some simulations
  - formalise property to be checked

The Model Checking Process

- Modeling Phase
  - model the system under consideration
  - as a first sanity check, perform some simulations
  - formalise property to be checked
- Running Phase
  - run the model checker to check the validity of the property in the model

The Model Checking Process

- Modeling Phase
  - model the system under consideration
  - as a first sanity check, perform some simulations
  - formalise property to be checked
- Running Phase
  - run the model checker to check the validity of the property in the model
- Analysis Phase
  - property satisfied? $\rightarrow$ check next property (if any)
  - property violated? $\rightarrow$
    - analyse generated counter example by simulation
    - refine the model, design, property, ... and repeat entire procedure
  - out of memory? $\rightarrow$ try to reduce the model and try again

The Pros of Model Checking

- widely applicable (hardware, software...)
- allows for partial verification (only relevant properties)
- potential “push-button” technology (tools)
- rapidly increasing industrial interest
- in case of property violation, a counter example is provided
- sound mathematical foundations
- not biased to the most possible scenarios (like testing)

The Cons of Model Checking

- model checking is only as “good” as the system model
- no guarantee about completeness of results (incomplete specifications)

Neverthless:
Model Checking is an effective technique to expose potential design errors

Striking Model-Checking Examples

- Security : Needham-Schroeder encryption protocol
  - error that remained undiscovered for 17 years revealed (model checker SAL)
- Transportation Systems
  - Train model containing 10^{47} states (model checker UPPAAL)
- Model Checkers for C, JAVA, C++
  - used (and developed) by Microsoft, Intel, NASA
  - successful application area: device drivers (model checker SLAM)
- Dutch storm surge barrier in Nieuwe Waterweg
- Software in current/next generation of space missiles
  - NASA's
    - Java Pathfinder, Deep Space Habitat, Lab for Reliable Software

Relevant Topics

- What are appropriate models?
  - from programs, circuits, communication protocols to transition systems

Relevant Topics

- What are appropriate models?
  - from programs, circuits, communication protocols to transition systems
- What are properties?
  - Safety, Liveness, fairness

Relevant Topics

- What are appropriate models?
  - from programs, circuits, communication protocols to transition systems
- What are properties?
  - Safety, Liveness, fairness
- How to check regular properties?
  - finite state automata and regular safety properties
  - Buchi automata and ω-regular properties

Relevant Topics

- How to express properties succintly?
  - First Order Logic (FO) : syntax, semantics
  - Monadic Second Order Logic (MSO) : syntax, semantics
  - Linear-Temporal-Logic (LTL) : syntax, semantics
  - What can be expressed in each logic?
  - Satisfiability and Model checking : algorithms, complexity

Relevant Topics

- How to express properties succintly?
  - First Order Logic (FO) : syntax, semantics
  - Monadic Second Order Logic (MSO) : syntax, semantics
  - Linear-Temporal-Logic (LTL) : syntax, semantics
  - What can be expressed in each logic?
  - Satisfiability and Model checking : algorithms, complexity
- How to make models succint?
  - Equivalences and partial-orders on transition systems
  - Which properties are preserved?
  - Minimization algorithms

CS 228 : Logic in Computer Science

Krishna. S

Welcome

What is this course about? A mini-zoo of logics.

Welcome

What is this course about? A mini-zoo of logics. Here are some typical questions you will learn to answer:

Welcome

What is this course about? A mini-zoo of logics. Here are some typical questions you will learn to answer:

- Q1: Given a formula φ in a logic L, is φ satisfiable?

Welcome

What is this course about? A mini-zoo of logics. Here are some typical questions you will learn to answer:

- Q1: Given a formula φ in a logic L, is φ satisfiable?
- Q2: Given a formula φ in a logic L, is φ valid?

Welcome

What is this course about? A mini-zoo of logics. Here are some typical questions you will learn to answer:

- Q1: Given a formula φ in a logic L, is φ satisfiable?
- Q2: Given a formula φ in a logic L, is φ valid?
- Q3: How easy is to answer Q1 and Q2?

Welcome

What is this course about? A mini-zoo of logics. Here are some typical questions you will learn to answer:

- Q1: Given a formula φ in a logic L, is φ satisfiable?
- Q2: Given a formula φ in a logic L, is φ valid?
- Q3: How easy is to answer Q1 and Q2?
- Q4: Can you write an algorithm to answer Q1 and Q2?

Welcome

What is this course about? A mini-zoo of logics. Here are some typical questions you will learn to answer:

- Q1: Given a formula φ in a logic L, is φ satisfiable?
- Q2: Given a formula φ in a logic L, is φ valid?
- Q3: How easy is to answer Q1 and Q2?
- Q4: Can you write an algorithm to answer Q1 and Q2?
- Q5: Can you "prove" any factually correct statement using the chosen logic L?

Welcome

What is this course about? A mini-zoo of logics. Here are some typical questions you will learn to answer:

- Q1: Given a formula φ in a logic L, is φ satisfiable?
- Q2: Given a formula φ in a logic L, is φ valid?
- Q3: How easy is to answer Q1 and Q2?
- Q4: Can you write an algorithm to answer Q1 and Q2?
- Q5: Can you "prove" any factually correct statement using the chosen logic L?
- Q6: How is logic L used in computer science?

Welcome

What is this course about? A mini-zoo of logics. Here are some typical questions you will learn to answer:

- Q1: Given a formula φ in a logic L, is φ satisfiable?
- Q2: Given a formula φ in a logic L, is φ valid?
- Q3: How easy is to answer Q1 and Q2?
- Q4: Can you write an algorithm to answer Q1 and Q2?
- Q5: Can you "prove" any factually correct statement using the chosen logic L?
- Q6: How is logic L used in computer science?
- Q7: What are the techniques needed to go about these questions?

Some Members of the mini-zoo

- Propositional Logic
- First Order Logic
- Monadic Second Order Logic
- Propositional Dynamic Logic
- Linear Temporal Logic
- Computational Tree Logic

More if time permits!

References

- To start with, the text book of Huth and Ryan : Logic for CS.
- As we go ahead, lecture notes/monographs/other text books.
- Classes : Slot 4. Tutorial: To discuss.

Propositional Logic

Syntax

- Finite set of propositional variables $p, q, \ldots$

Syntax

- Finite set of propositional variables $p, q, \dots$
- Each of these can be true/false

Syntax

- Finite set of propositional variables $p, q, \ldots$
- Each of these can be true/false
- Combine propositions using $\neg, \vee, \wedge, \rightarrow$

Syntax

- Finite set of propositional variables $p, q, \ldots$
- Each of these can be true/false
- Combine propositions using $\neg, \vee, \wedge, \rightarrow$
- Parantheses as required

Syntax

- Finite set of propositional variables $p, q, \ldots$
- Each of these can be true/false
- Combine propositions using $\neg, \vee, \wedge, \rightarrow$
- Parantheses as required
- Example : $[p \wedge (q \vee r)] \rightarrow [\neg r \wedge p]$
- $\neg$ binds tighter than $\vee, \wedge$, which bind tighter than $\rightarrow$. In the absence of parantheses, $p \rightarrow q \rightarrow r$ is read as $p \rightarrow (q \rightarrow r)$

Natural Deduction

- If it rains, Alice is outside and does not have any raingear with her, she will get wet. $\varphi = (R \wedge \mathit{AliceOut} \wedge \neg \mathit{RG}) \rightarrow \mathit{AliceWet}$

Natural Deduction

- If it rains, Alice is outside and does not have any raingear with her, she will get wet. $\varphi = (R \wedge \mathit{AliceOut} \wedge \neg \mathit{RG}) \rightarrow \mathit{AliceWet}$
- It is raining, and Alice is outside, and is not wet. $\psi = (R \wedge \mathit{AliceOut} \wedge \neg \mathit{AliceWet})$

Natural Deduction

- If it rains, Alice is outside and does not have any raingear with her, she will get wet. $\varphi = (R \wedge \mathit{AliceOut} \wedge \neg RG) \rightarrow \mathit{AliceWet}$
- It is raining, and Alice is outside, and is not wet. $\psi = (R \wedge \mathit{AliceOut} \wedge \neg \mathit{AliceWet})$
- So, Alice has her rain gear with her. *RG*
- Thus, $\chi = \varphi \wedge \psi \rightarrow RG$. You can deduce *RG* from $\varphi \wedge \psi$.
- Is χ valid? Is χ satisfiable?

Two Examples of Natural Deduction

Solve Sudoku

Consider the following kid's version of Sudoku.

	2	4	
1			3
4			2
	1	3	

Rules:

- Each row must contain all numbers 1-4
- Each column must contain all numbers 1-4
- Each 2×2 block must contain all numbers 1-4
- No cell contains 2 or more numbers

Encoding as Propositional Satisfiability

- Proposition $P(i, j, n)$ is true when cell (i, j) has number n

Encoding as Propositional Satisfiability

- Proposition $P(i, j, n)$ is true when cell (i, j) has number n
- $4 \times 4 \times 4$ propositions

Encoding as Propositional Satisfiability

- Proposition $P(i, j, n)$ is true when cell (i, j) has number n
- $4 \times 4 \times 4$ propositions
- Each row must contain all 4 numbers
  - Row 1: $[P(1,1,1) \vee P(1,2,1) \vee P(1,3,1) \vee P(1,4,1)] \wedge$
    $[P(1,1,2) \vee P(1,2,2) \vee P(1,3,2) \vee P(1,4,2)] \wedge$
    $[P(1,1,3) \vee P(1,2,3) \vee P(1,3,3) \vee P(1,4,3)] \wedge$
    $[P(1,1,4) \vee P(1,2,4) \vee P(1,3,4) \vee P(1,4,4)]$

Encoding as Propositional Satisfiability

- Proposition $P(i, j, n)$ is true when cell (i, j) has number n
- $4 \times 4 \times 4$ propositions
- Each row must contain all 4 numbers
  - Row 1: $[P(1,1,1) \vee P(1,2,1) \vee P(1,3,1) \vee P(1,4,1)] \wedge$
    $[P(1,1,2) \vee P(1,2,2) \vee P(1,3,2) \vee P(1,4,2)] \wedge$
    $[P(1,1,3) \vee P(1,2,3) \vee P(1,3,3) \vee P(1,4,3)] \wedge$
    $[P(1,1,4) \vee P(1,2,4) \vee P(1,3,4) \vee P(1,4,4)]$
  - Row 2: $[P(2,1,1) \vee \ldots$
  - Row 3: $[P(3,1,1) \vee \ldots$
  - Row 4: $[P(4,1,1) \vee \ldots$

Encoding as Propositional Satisfiability

Each column must contain all numbers 1-4

Encoding as Propositional Satisfiability

Each column must contain all numbers 1-4

- Column 1: $[P(1,1,1) \vee P(2,1,1) \vee P(3,1,1) \vee P(4,1,1)] \wedge$
  $[P(1,1,2) \vee P(2,1,2) \vee P(3,1,2) \vee P(4,1,2)] \wedge$
  $[P(1,1,3) \vee P(2,1,3) \vee P(3,1,3) \vee P(4,1,3)] \wedge$
  $[P(1,1,4) \vee P(2,1,4) \vee P(3,1,4) \vee P(4,1,4)]$

Encoding as Propositional Satisfiability

Each column must contain all numbers 1-4

- Column 1: $[P(1,1,1) \vee P(2,1,1) \vee P(3,1,1) \vee P(4,1,1)] \wedge$
  $[P(1,1,2) \vee P(2,1,2) \vee P(3,1,2) \vee P(4,1,2)] \wedge$
  $[P(1,1,3) \vee P(2,1,3) \vee P(3,1,3) \vee P(4,1,3)] \wedge$
  $[P(1,1,4) \vee P(2,1,4) \vee P(3,1,4) \vee P(4,1,4)]$
- Column 2: $[P(1,2,1) \vee \ldots$
- Column 3: $[P(1,3,1) \vee \ldots$
- Column 4: $[P(1,4,1) \vee \ldots$

Encoding as Propositional Satisfiability

Each 2×2 block must contain all numbers 1-4

Encoding as Propositional Satisfiability

Each 2×2 block must contain all numbers 1-4

- Upper left block contains all numbers 1-4:

$$[P(1,1,1) \vee P(1,2,1) \vee P(2,1,1) \vee P(2,2,1)] \wedge$$
$$[P(1,1,2) \vee P(1,2,2) \vee P(2,1,2) \vee P(2,2,2)] \wedge$$
$$[P(1,1,3) \vee P(1,2,3) \vee P(2,1,3) \vee P(2,2,3)] \wedge$$
$$[P(1,1,4) \vee P(1,2,4) \vee P(2,1,4) \vee P(2,2,4)]$$

Encoding as Propositional Satisfiability

Each 2×2 block must contain all numbers 1-4

- Upper left block contains all numbers 1-4:

$$[P(1,1,1) \vee P(1,2,1) \vee P(2,1,1) \vee P(2,2,1)] \wedge$$
$$[P(1,1,2) \vee P(1,2,2) \vee P(2,1,2) \vee P(2,2,2)] \wedge$$
$$[P(1,1,3) \vee P(1,2,3) \vee P(2,1,3) \vee P(2,2,3)] \wedge$$
$$[P(1,1,4) \vee P(1,2,4) \vee P(2,1,4) \vee P(2,2,4)]$$

- Upper right block contains all numbers 1-4:

$$[P(1,3,1) \vee P(1,4,1) \vee P(2,3,1) \vee P(2,4,1)] \wedge \ldots$$

- Lower left block contains all numbers 1-4:

$$[P(3,1,1) \vee P(3,2,1) \vee P(4,1,1) \vee P(4,2,1)] \wedge \ldots$$

- Lower right block contains all numbers 1-4:

$$[P(3,3,1) \vee P(3,4,1) \vee P(4,3,1) \vee P(4,4,1)] \wedge \ldots$$

Encoding as Propositional Satisfiability

No cell contains 2 or more numbers

- For cell(1,1):

$$P(1,1,1) \rightarrow [\neg P(1,1,2) \wedge \neg P(1,1,3) \wedge \neg P(1,1,4)] \wedge$$

$$P(1,1,2) \rightarrow [\neg P(1,1,1) \wedge \neg P(1,1,3) \wedge \neg P(1,1,4)] \wedge$$

$$P(1,1,3) \rightarrow [\neg P(1,1,1) \wedge \neg P(1,1,2) \wedge \neg P(1,1,4)] \wedge$$

$$P(1,1,4) \rightarrow [\neg P(1,1,1) \wedge \neg P(1,1,2) \wedge \neg P(1,1,3)] \wedge$$

- Similar for other cells

Encoding as Propositional Satisfiability

Encoding Initial Configuration:

$$P(1,2,2) \wedge P(1,3,4) \wedge P(2,1,1) \wedge P(2,4,3) \wedge$$

$$P(3,1,4) \wedge P(3,4,2) \wedge P(4,2,1) \wedge P(4,3,3)$$

Solving Sodoku

To solve the puzzle, just conjunct all the above formulae and find a satisfiable truth assignment!

Gold Rush

(**Box1**)	*The gold is not here*
(**Box2**)	*The gold is not here*
(**Box3**)	*The gold is in Box 2*

Only one message is true; the other two are false. Which box has the gold?

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$,

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$, $M2 \leftrightarrow \neg G2$,

Solve Gold Rush

- Propositions $M1$, $M2$, $M3$ representing messages in boxes 1,2,3
- Propositions $G1$, $G2$, $G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$, $M2 \leftrightarrow \neg G2$, $M3 \leftrightarrow G2$

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$, $M2 \leftrightarrow \neg G2$, $M3 \leftrightarrow G2$
  - $\neg(M1 \wedge M2 \wedge M3)$,

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$, $M2 \leftrightarrow \neg G2$, $M3 \leftrightarrow G2$
  - $\neg(M1 \land M2 \land M3)$, $M1 \lor M2 \lor M3$,

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$, $M2 \leftrightarrow \neg G2$, $M3 \leftrightarrow G2$
  - $\neg(M1 \land M2 \land M3)$, $M1 \lor M2 \lor M3$,
  - $(\neg M1 \land \neg M2) \lor (\neg M1 \land \neg M3) \lor (\neg M2 \land \neg M3)$

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$, $M2 \leftrightarrow \neg G2$, $M3 \leftrightarrow G2$
  - $\neg(M1 \wedge M2 \wedge M3)$, $M1 \vee M2 \vee M3$,
  - $(\neg M1 \wedge \neg M2) \vee (\neg M1 \wedge \neg M3) \vee (\neg M2 \wedge \neg M3)$
  - Conjunct all these, and call the formula φ.

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$, $M2 \leftrightarrow \neg G2$, $M3 \leftrightarrow G2$
  - $\neg(M1 \wedge M2 \wedge M3)$, $M1 \vee M2 \vee M3$,
  - $(\neg M1 \wedge \neg M2) \vee (\neg M1 \wedge \neg M3) \vee (\neg M2 \wedge \neg M3)$
  - Conjunct all these, and call the formula φ.
  - Is there a unique satisfiable assignment for φ?

Solve Gold Rush

- Propositions $M1, M2, M3$ representing messages in boxes 1,2,3
- Propositions $G1, G2, G3$ representing gold in boxes 1,2,3
- Formalize what is given to you
  - $M1 \leftrightarrow \neg G1$, $M2 \leftrightarrow \neg G2$, $M3 \leftrightarrow G2$
  - $\neg(M1 \wedge M2 \wedge M3)$, $M1 \vee M2 \vee M3$,
  - $(\neg M1 \wedge \neg M2) \vee (\neg M1 \wedge \neg M3) \vee (\neg M2 \wedge \neg M3)$
  - Conjunct all these, and call the formula φ.
  - Is there a unique satisfiable assignment for φ?
  - For example, is $M1 = \mathit{true}$ a part of the satisfiable assignment?

A Proof Engine for Natural Deduction

- If it rains, Alice is outside and does not have any raingear with her, she will get wet. $\varphi = (R \wedge \mathit{AliceOut} \wedge \neg \mathit{RG}) \rightarrow \mathit{AliceWet}$
- It is raining, and Alice is outside, and is not wet. $\psi = (R \wedge \mathit{AliceOut} \wedge \neg \mathit{AliceWet})$
- So, Alice has her rain gear with her. *RG*
- Thus, $\chi = \varphi \wedge \psi \rightarrow \mathit{RG}$.
- Given φ, ψ, can we "prove" *RG*?

A Proof Engine

- Given a formula φ in propositional logic, how to “prove” φ if φ is valid?
- What is a proof engine?
- Show that this proof engine is sound and complete
  - Completeness: Any fact that can be captured using propositional logic can be proved by the proof engine
  - Soundness: Any formula that is proved to be valid by the proof engine is indeed valid

Natural Deduction

- In natural deduction, we have a collection of proof rules

Natural Deduction

- In natural deduction, we have a collection of proof rules
- These proof rules allow us to infer formulae from some given formulae

Natural Deduction

- In natural deduction, we have a collection of proof rules
- These proof rules allow us to infer formulae from some given formulae
- Given a set of premises, we deduce a conclusion which is also a formula using proof rules.

Natural Deduction

- In natural deduction, we have a collection of proof rules
- These proof rules allow us to infer formulae from some given formulae
- Given a set of premises, we deduce a conclusion which is also a formula using proof rules.
- $\varphi_1, \ldots, \varphi_n \vdash \psi$: This is called a sequent. $\varphi_1, \ldots, \varphi_n$ are premises, and ψ, the conclusion.
- Given $\varphi_1, \ldots, \varphi_n$, we can deduce or prove ψ. What was the sequent in the Alice example?

Natural Deduction

- In natural deduction, we have a collection of proof rules
- These proof rules allow us to infer formulae from some given formulae
- Given a set of premises, we deduce a conclusion which is also a formula using proof rules.
- $\varphi_1, \ldots, \varphi_n \vdash \psi$: This is called a sequent. $\varphi_1, \ldots, \varphi_n$ are premises, and ψ, the conclusion.
- Given $\varphi_1, \ldots, \varphi_n$, we can deduce or prove ψ. What was the sequent in the Alice example?
- For example, $\neg p \rightarrow q, q \rightarrow r, \neg r \vdash p$ is a sequent. How do you prove this?

Natural Deduction

- In natural deduction, we have a collection of proof rules
- These proof rules allow us to infer formulae from some given formulae
- Given a set of premises, we deduce a conclusion which is also a formula using proof rules.
- $\varphi_1, \ldots, \varphi_n \vdash \psi$: This is called a sequent. $\varphi_1, \ldots, \varphi_n$ are premises, and ψ, the conclusion.
- Given $\varphi_1, \ldots, \varphi_n$, we can deduce or prove ψ. What was the sequent in the Alice example?
- For example, $\neg p \to q, q \to r, \neg r \vdash p$ is a sequent. How do you prove this?
- Proof rules to be carefully chosen, for instance you should not end up proving something like $p \wedge q \vdash \neg q$

The Rules of the Proof Engine

Rules for Natural Deduction

The and introduction rule denoted $\wedge i$

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi}$$

Rules for Natural Deduction

The and elimination rule denoted $\wedge e_1$

$$\frac{\varphi \wedge \psi}{\varphi}$$

The and elimination rule denoted $\wedge e_2$

$$\frac{\varphi \wedge \psi}{\psi}$$

A first proof using $\wedge i, \wedge e_1, \wedge e_2$

- Show that $p \wedge q, r \vdash q \wedge r$

1.	$p \wedge q$	premise
2.		

A first proof using $\wedge i, \wedge e_1, \wedge e_2$

- Show that $p \wedge q, r \vdash q \wedge r$

1.	$p \wedge q$	premise
2.	r	premise
3.		

A first proof using $\wedge i, \wedge e_1, \wedge e_2$

- Show that $p \wedge q, r \vdash q \wedge r$

1.	$p \wedge q$	premise
2.	r	premise
3.	q	$\wedge e_2$ 1
4.		

A first proof using $\wedge i, \wedge e_1, \wedge e_2$

- Show that $p \wedge q, r \vdash q \wedge r$

1.	$p \wedge q$	premise
2.	r	premise
3.	q	$\wedge e_2$ 1
4.	$q \wedge r$	$\wedge i$ 3,2

Rules for Natural Deduction

The rule of double negation elimination $\neg\neg e$

$$\frac{\neg\neg\varphi}{\varphi}$$

The rule of double negation introduction $\neg\neg i$

$$\frac{\varphi}{\neg\neg\varphi}$$

Rules for Natural Deduction

The implies elimination rule or Modus Ponens MP

$$\frac{\varphi \qquad \varphi \rightarrow \psi}{\psi}$$

Another Proof

- Show that $p, p \to q, p \to (q \to \neg\neg r) \vdash r$

1.	$p \to (q \to \neg\neg r)$	premise
2.		

Another Proof

- Show that $p, p \to q, p \to (q \to \neg\neg r) \vdash r$

1.	$p \to (q \to \neg\neg r)$	premise
2.	$p \to q$	premise
3.		

Another Proof

- Show that $p, p \to q, p \to (q \to \neg\neg r) \vdash r$

1.	$p \to (q \to \neg\neg r)$	premise
2.	$p \to q$	premise
3.	p	premise
4.		

Another Proof

- Show that $p, p \to q, p \to (q \to \neg\neg r) \vdash r$

1.	$p \to (q \to \neg\neg r)$	premise
2.	$p \to q$	premise
3.	p	premise
4.	$q \to \neg\neg r$	MP 1,3
5.		

Another Proof

- Show that $p, p \to q, p \to (q \to \neg\neg r) \vdash r$

1.	$p \to (q \to \neg\neg r)$	premise
2.	$p \to q$	premise
3.	p	premise
4.	$q \to \neg\neg r$	MP 1,3
5.	q	MP 2,3
6.		

Another Proof

- Show that $p, p \to q, p \to (q \to \neg\neg r) \vdash r$

1.	$p \to (q \to \neg\neg r)$	premise
2.	$p \to q$	premise
3.	p	premise
4.	$q \to \neg\neg r$	MP 1,3
5.	q	MP 2,3
6.	$\neg\neg r$	MP 4,5
7.		

Another Proof

- Show that $p, p \to q, p \to (q \to \neg\neg r) \vdash r$

1.	$p \to (q \to \neg\neg r)$	premise
2.	$p \to q$	premise
3.	p	premise
4.	$q \to \neg\neg r$	MP 1,3
5.	q	MP 2,3
6.	$\neg\neg r$	MP 4,5
7.	r	$\neg\neg e$ 6

CS 228 : Logic in Computer Science

Krishna. S

Rules for Natural Deduction

Another implies elimination rule or Modus Tollens MT

$$\frac{\varphi \rightarrow \psi \qquad \neg\psi}{\neg\varphi}$$

A Proof

- Show that $p \rightarrow \neg q, q \vdash \neg p$

1.	$p \rightarrow \neg q$	premise
2.		

A Proof

- Show that $p \rightarrow \neg q, q \vdash \neg p$

1.	$p \rightarrow \neg q$	premise
2.	q	premise
3.		

A Proof

- Show that $p \to \neg q, q \vdash \neg p$

1.	$p \to \neg q$	premise
2.	q	premise
3.	$\neg\neg q$	$\neg\neg i$ 2
4.		

A Proof

- Show that $p \to \neg q, q \vdash \neg p$

1.	$p \to \neg q$	premise
2.	q	premise
3.	$\neg\neg q$	$\neg\neg i$ 2
4.	$\neg p$	MT 1,3

More Rules

- Thanks to MT, we have $p \to q, \neg q \vdash \neg p$.

More Rules

- Thanks to MT, we have $p \to q, \neg q \vdash \neg p$.
- Can we prove $p \to q \vdash \neg q \to \neg p$?

More Rules

- Thanks to MT, we have $p \to q, \neg q \vdash \neg p$.
- Can we prove $p \to q \vdash \neg q \to \neg p$?
- So far, no proof rule that can do this.

More Rules

- Thanks to MT, we have $p \to q, \neg q \vdash \neg p$.
- Can we prove $p \to q \vdash \neg q \to \neg p$?
- So far, no proof rule that can do this.
- Given $p \to q$, let us assume $\neg q$. Can we then prove $\neg p$?

More Rules

- Thanks to MT, we have $p \to q, \neg q \vdash \neg p$.
- Can we prove $p \to q \vdash \neg q \to \neg p$?
- So far, no proof rule that can do this.
- Given $p \to q$, let us assume $\neg q$. Can we then prove $\neg p$?
- Yes, using MT.

The implies introduction rule $\rightarrow i$

- $p \rightarrow q \vdash \neg q \rightarrow \neg p$

1.	$p \rightarrow q$	premise
2.	$\neg q$	assumption
3.	$\neg p$	MT 1,2
4.	$\neg q \rightarrow \neg p$	$\rightarrow i$ 2-3

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	*true*	premise
2.		

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	*true*	premise
2.	$q \rightarrow r$	assumption
3.		

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	*true*	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.		

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	$true$	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.	p	assumption
5.		

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	$true$	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.	p	assumption
5.	$\neg\neg p$	$\neg\neg i$ 4
6.		

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	$true$	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.	p	assumption
5.	$\neg\neg p$	$\neg\neg i$ 4
6.	$\neg\neg q$	MT 3,5
7.		

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	$true$	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.	p	assumption
5.	$\neg\neg p$	$\neg\neg i$ 4
6.	$\neg\neg q$	MT 3,5
7.	q	$\neg\neg e$ 6
8.		

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	*true*	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.	p	assumption
5.	$\neg\neg p$	$\neg\neg i$ 4
6.	$\neg\neg q$	MT 3,5
7.	q	$\neg\neg e$ 6
8.	r	MP 2,7

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	$true$	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.	p	assumption
5.	$\neg\neg p$	$\neg\neg i$ 4
6.	$\neg\neg q$	MT 3,5
7.	q	$\neg\neg e$ 6
8.	r	MP 2,7
9.	$p \rightarrow r$	$\rightarrow i$ 4-8

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	$true$	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.	p	assumption
5.	$\neg\neg p$	$\neg\neg i$ 4
6.	$\neg\neg q$	MT 3,5
7.	q	$\neg\neg e$ 6
8.	r	MP 2,7
9.	$p \rightarrow r$	$\rightarrow i$ 4-8
10.	$(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)$	$\rightarrow i$ 3-9
11.		

More on $\rightarrow i$

- $\vdash (q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$

1.	*true*	premise
2.	$q \rightarrow r$	assumption
3.	$\neg q \rightarrow \neg p$	assumption
4.	p	assumption
5.	$\neg\neg p$	$\neg\neg i$ 4
6.	$\neg\neg q$	MT 3,5
7.	q	$\neg\neg e$ 6
8.	r	MP 2,7
9.	$p \rightarrow r$	$\rightarrow i$ 4-8
10.	$(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)$	$\rightarrow i$ 3-9
11.	$(q \rightarrow r) \rightarrow [(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)]$	$\rightarrow i$ 2-10

Transforming Proofs

- $(q \to r), (\neg q \to \neg p), p \vdash r$
- Transform any proof $\varphi_1, \ldots, \varphi_n \vdash \psi$ to $\vdash \varphi_1 \to (\varphi_2 \to \ldots (\varphi_n \to \psi) \ldots))$ by adding n lines of the rule $\to i$

More Examples

- $p \to (q \to r) \vdash (p \land q) \to r$

1.	$p \to (q \to r)$	premise
2.		

More Examples

- $p \to (q \to r) \vdash (p \wedge q) \to r$

1.	$p \to (q \to r)$	premise
2.	$p \wedge q$	assumption
3.		

More Examples

- $p \to (q \to r) \vdash (p \land q) \to r$

1.	$p \to (q \to r)$	premise
2.	$p \land q$	assumption
3.	p	$\land e_1$ 2
4.		

More Examples

- $p \to (q \to r) \vdash (p \land q) \to r$

1.	$p \to (q \to r)$	premise
2.	$p \land q$	assumption
3.	p	$\land e_1$ 2
4.	q	$\land e_2$ 2
5.		

More Examples

- $p \to (q \to r) \vdash (p \wedge q) \to r$

1.	$p \to (q \to r)$	premise
2.	$p \wedge q$	assumption
3.	p	$\wedge e_1$ 2
4.	q	$\wedge e_2$ 2
5.	$q \to r$	MP 1,3
6.		

More Examples

- $p \to (q \to r) \vdash (p \wedge q) \to r$

1.	$p \to (q \to r)$	premise
2.	$p \wedge q$	assumption
3.	p	$\wedge e_1$ 2
4.	q	$\wedge e_2$ 2
5.	$q \to r$	MP 1,3
6.	r	MP 4,5
7.		

More Examples

- $p \to (q \to r) \vdash (p \wedge q) \to r$

1.	$p \to (q \to r)$	premise
2.	$p \wedge q$	assumption
3.	p	$\wedge e_1$ 2
4.	q	$\wedge e_2$ 2
5.	$q \to r$	MP 1,3
6.	r	MP 4,5
7.	$p \wedge q \to r$	$\to i$ 2-6

More Rules

The or introduction rule $\vee i_1$

$$\frac{\varphi}{\varphi \vee \psi}$$

The or introduction rule $\vee i_2$

$$\frac{\psi}{\varphi \vee \psi}$$

More Rules

The or elimination rule $\vee e$

$$\frac{\varphi \vee \psi \qquad \varphi \vdash \chi \qquad \psi \vdash \chi}{\chi}$$

Or Elimination Example

- $q \to r \vdash (p \vee q) \to (p \vee r)$

1.	$q \to r$	premise
2.		

Or Elimination Example

- $q \to r \vdash (p \vee q) \to (p \vee r)$

1.	$q \to r$	premise
2.	$p \vee q$	assumption
3.		

Or Elimination Example

- $q \rightarrow r \vdash (p \vee q) \rightarrow (p \vee r)$

1.	$q \rightarrow r$	premise
2.	$p \vee q$	assumption
3.	p	$\vee e$ (1)
4.		

Or Elimination Example

- $q \to r \vdash (p \vee q) \to (p \vee r)$

1.	$q \to r$	premise
2.	$p \vee q$	assumption
3.	p	$\vee e$ (1)
4.	$p \vee r$	$\vee i_1$ 3
5.		

Or Elimination Example

- $q \to r \vdash (p \vee q) \to (p \vee r)$

1.	$q \to r$	premise
2.	$p \vee q$	assumption
3.	p	$\vee e$ (1)
4.	$p \vee r$	$\vee i_1$ 3
5.	q	$\vee e$ (2)
6.		

Or Elimination Example

- $q \to r \vdash (p \vee q) \to (p \vee r)$

1.	$q \to r$	premise
2.	$p \vee q$	assumption
3.	p	$\vee\ e\ (1)$
4.	$p \vee r$	$\vee i_1$ 3
5.	q	$\vee\ e\ (2)$
6.	r	MP 1,5
7.		

Or Elimination Example

- $q \rightarrow r \vdash (p \vee q) \rightarrow (p \vee r)$

1.	$q \rightarrow r$	premise
2.	$p \vee q$	assumption
3.	p	$\vee\ e\ (1)$
4.	$p \vee r$	$\vee i_1$ 3
5.	q	$\vee\ e\ (2)$
6.	r	MP 1,5
7.	$p \vee r$	$\vee i_2$ 6

Or Elimination Example

- $q \to r \vdash (p \vee q) \to (p \vee r)$

1.	$q \to r$	premise
2.	$p \vee q$	assumption
3.	p	$\vee\ e\ (1)$
4.	$p \vee r$	$\vee i_1$ 3
5.	q	$\vee\ e\ (2)$
6.	r	MP 1,5
7.	$p \vee r$	$\vee i_2$ 6
8.	$p \vee r$	$\vee e$ 2, 3-4, 5-7

Or Elimination Example

- $q \rightarrow r \vdash (p \vee q) \rightarrow (p \vee r)$

1.	$q \rightarrow r$	premise
2.	$p \vee q$	assumption
3.	p	$\vee\ e$ (1)
4.	$p \vee r$	$\vee i_1$ 3
5.	q	$\vee\ e$ (2)
6.	r	MP 1,5
7.	$p \vee r$	$\vee i_2$ 6
8.	$p \vee r$	$\vee e$ 2, 3-4, 5-7
9.		

Or Elimination Example

- $q \rightarrow r \vdash (p \vee q) \rightarrow (p \vee r)$

1.	$q \rightarrow r$	premise
2.	$p \vee q$	assumption
3.	p	$\vee\ e\ (1)$
4.	$p \vee r$	$\vee i_1$ 3
5.	q	$\vee\ e\ (2)$
6.	r	MP 1,5
7.	$p \vee r$	$\vee i_2$ 6
8.	$p \vee r$	$\vee e$ 2, 3-4, 5-7
9.	$(p \vee q) \rightarrow (p \vee r)$	$\rightarrow i$ 2-8

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee e$ (1)
3.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee e$ (1)
3.	p	$\vee e$ (1.1)
4.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee e\,(1)$
3.	p	$\vee e\,(1.1)$
4.	$p \vee (q \vee r)$	$\vee i_1$ 3
5.		

Associativity Using Or Elimination

- $(p \lor q) \lor r \vdash p \lor (q \lor r)$

1.	$(p \lor q) \lor r$	premise
2.	$p \lor q$	$\lor e$ (1)
3.	p	$\lor e$ (1.1)
4.	$p \lor (q \lor r)$	$\lor i_1$ 3
5.	q	$\lor e$ (1.2)
6.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee e\,(1)$
3.	p	$\vee e\,(1.1)$
4.	$p \vee (q \vee r)$	$\vee i_1\ 3$
5.	q	$\vee e\,(1.2)$
6.	$q \vee r$	$\vee i_1\ 5$
7.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee\, e\,(1)$
3.	p	$\vee\, e\,(1.1)$
4.	$p \vee (q \vee r)$	$\vee i_1\; 3$
5.	q	$\vee\, e\,(1.2)$
6.	$q \vee r$	$\vee i_1\; 5$
7.	$p \vee (q \vee r)$	$\vee i_2\; 6$
8.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee e$ (1)
3.	p	$\vee e$ (1.1)
4.	$p \vee (q \vee r)$	$\vee i_1$ 3
5.	q	$\vee e$ (1.2)
6.	$q \vee r$	$\vee i_1$ 5
7.	$p \vee (q \vee r)$	$\vee i_2$ 6
8.	$p \vee (q \vee r)$	$\vee e$ 2, 3-4, 5-7
9.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee\ e\ (1)$
3.	p	$\vee\ e\ (1.1)$
4.	$p \vee (q \vee r)$	$\vee i_1\ 3$
5.	q	$\vee\ e\ (1.2)$
6.	$q \vee r$	$\vee i_1\ 5$
7.	$p \vee (q \vee r)$	$\vee i_2\ 6$
8.	$p \vee (q \vee r)$	$\vee e$ 2, 3-4, 5-7
9.	r	$\vee\ e\ (2)$
10.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee e$ (1)
3.	p	$\vee e$ (1.1)
4.	$p \vee (q \vee r)$	$\vee i_1$ 3
5.	q	$\vee e$ (1.2)
6.	$q \vee r$	$\vee i_1$ 5
7.	$p \vee (q \vee r)$	$\vee i_2$ 6
8.	$p \vee (q \vee r)$	$\vee e$ 2, 3-4, 5-7
9.	r	$\vee e$ (2)
10.	$q \vee r$	$\vee i_2$ 9
11.		

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee e$ (1)
3.	p	$\vee e$ (1.1)
4.	$p \vee (q \vee r)$	$\vee i_1$ 3
5.	q	$\vee e$ (1.2)
6.	$q \vee r$	$\vee i_1$ 5
7.	$p \vee (q \vee r)$	$\vee i_2$ 6
8.	$p \vee (q \vee r)$	$\vee e$ 2, 3-4, 5-7
9.	r	$\vee e$ (2)
10.	$q \vee r$	$\vee i_2$ 9
11.	$p \vee (q \vee r)$	$\vee i_2$ 10

Associativity Using Or Elimination

- $(p \vee q) \vee r \vdash p \vee (q \vee r)$

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	$\vee\ e\ (1)$
3.	p	$\vee\ e\ (1.1)$
4.	$p \vee (q \vee r)$	$\vee i_1$ 3
5.	q	$\vee\ e\ (1.2)$
6.	$q \vee r$	$\vee i_1$ 5
7.	$p \vee (q \vee r)$	$\vee i_2$ 6
8.	$p \vee (q \vee r)$	$\vee e$ 2, 3-4, 5-7
9.	r	$\vee\ e\ (2)$
10.	$q \vee r$	$\vee i_2$ 9
11.	$p \vee (q \vee r)$	$\vee i_2$ 10
12.	$p \vee (q \vee r)$	$\vee e$ 1, 2-8, 9-11

Basic Rules So Far

- $\wedge$i, $\wedge e_1$, $\wedge e_2$ (and introduction and elimination)
- $\neg\neg e$, $\neg\neg i$ (double negation elimination and introduction)
- MP (Modus Ponens)
- $\rightarrow i$ (Implies Introduction : remember opening boxes)
- $\vee i_1, \vee i_2, \vee e$ (Or introduction and elimination)

The Copy Rule

- $\vdash p \to (q \to p)$

1.	*true*	premise
2.		

The Copy Rule

- $\vdash p \rightarrow (q \rightarrow p)$

1.	*true*	premise
2.	p	assumption
3.		

The Copy Rule

- $\vdash p \to (q \to p)$

1.	*true*	premise
2.	p	assumption
3.	q	assumption
4.		

The Copy Rule

- $\vdash p \to (q \to p)$

1.	*true*	premise
2.	p	assumption
3.	q	assumption
4.	p	copy 2
5.		

The Copy Rule

- $\vdash p \rightarrow (q \rightarrow p)$

1.	*true*	premise
2.	p	assumption
3.	q	assumption
4.	p	copy 2
5.	$q \rightarrow p$	$\rightarrow i$ 3-4
6.		

The Copy Rule

- $\vdash p \rightarrow (q \rightarrow p)$

1.	*true*	premise
2.	p	assumption
3.	q	assumption
4.	p	copy 2
5.	$q \rightarrow p$	$\rightarrow i$ 3-4
6.	$p \rightarrow (q \rightarrow p)$	$\rightarrow i$ 2-5

The Rules of Single Negation

- We have seen $\neg\neg e$ and $\neg\neg i$, the elimination and introduction of double negation.

The Rules of Single Negation

- We have seen $\neg\neg e$ and $\neg\neg i$, the elimination and introduction of double negation.
- How about introducing and eliminating single negations?

The Rules of Single Negation

- We have seen $\neg\neg e$ and $\neg\neg i$, the elimination and introduction of double negation.
- How about introducing and eliminating single negations?
- We use the notion of contradictions, an expression of the form $\varphi \wedge \neg\varphi$, where φ is any propositional logic formula.

The Rules of Single Negation

- We have seen $\neg\neg e$ and $\neg\neg i$, the elimination and introduction of double negation.
- How about introducing and eliminating single negations?
- We use the notion of contradictions, an expression of the form $\varphi \wedge \neg\varphi$, where φ is any propositional logic formula.
- Any two contradictions are equivalent : $p \wedge \neg p$ is equivalent to $\neg r \wedge r$. Contradictions denoted by $\perp$.

The Rules of Single Negation

- We have seen $\neg\neg e$ and $\neg\neg i$, the elimination and introduction of double negation.
- How about introducing and eliminating single negations?
- We use the notion of contradictions, an expression of the form $\varphi \wedge \neg\varphi$, where φ is any propositional logic formula.
- Any two contradictions are equivalent : $p \wedge \neg p$ is equivalent to $\neg r \wedge r$. Contradictions denoted by $\perp$.
- $\perp \rightarrow \varphi$ for any formula φ.

Rules with $\perp$

The $\perp$ elimination rule $\perp e$

$$\frac{\perp}{\psi}$$

The $\perp$ introduction rule $\perp i$

$$\frac{\varphi \qquad \neg\varphi}{\perp}$$

An Example

- $\neg p \vee q \vdash p \rightarrow q$

1.	$\neg p \vee q$	premise
2.		

An Example

- $\neg p \lor q \vdash p \to q$

1.	$\neg p \lor q$	premise
2.	$\neg p$	$\lor e$ (1)
3.		

An Example

- $\neg p \vee q \vdash p \rightarrow q$

1.	$\neg p \vee q$	premise
2.	$\neg p$	$\vee e$ (1)
3.	p	assumption
4.		

An Example

- $\neg p \vee q \vdash p \to q$

1.	$\neg p \vee q$	premise
2.	$\neg p$	$\vee e$ (1)
3.	p	assumption
4.	$\perp$	$\perp i$ 2,3
5.		

An Example

- $\neg p \vee q \vdash p \rightarrow q$

1.	$\neg p \vee q$	premise
2.	$\neg p$	$\vee e$ (1)
3.	p	assumption
4.	$\perp$	$\perp i$ 2,3
5.	q	$\perp e$ 4
6.		

An Example

- $\neg p \vee q \vdash p \rightarrow q$

1.	$\neg p \vee q$	premise
2.	$\neg p$	$\vee e$ (1)
3.	p	assumption
4.	$\bot$	$\bot i$ 2,3
5.	q	$\bot e$ 4
6.	$p \rightarrow q$	$\rightarrow i$ 3-5
7.	q	$\vee e$ (2)
8.		

An Example

- $\neg p \vee q \vdash p \rightarrow q$

1.	$\neg p \vee q$	premise
2.	$\neg p$	$\vee e$ (1)
3.	p	assumption
4.	$\perp$	$\perp i$ 2,3
5.	q	$\perp e$ 4
6.	$p \rightarrow q$	$\rightarrow i$ 3-5
7.	q	$\vee e$ (2)
8.	p	assumption
9.		

An Example

- $\neg p \vee q \vdash p \rightarrow q$

1.	$\neg p \vee q$	premise
2.	$\neg p$	$\vee e$ (1)
3.	p	assumption
4.	$\perp$	$\perp i$ 2,3
5.	q	$\perp e$ 4
6.	$p \rightarrow q$	$\rightarrow i$ 3-5
7.	q	$\vee e$ (2)
8.	p	assumption
9.	q	copy 7
10.	$p \rightarrow q$	$\rightarrow i$ 8-9
11.	$p \rightarrow q$	$\vee e$ 1, 2-6, 7-10

Introducing Negations (PBC)

- In the course of a proof, if you assume φ (by opening a box) and obtain $\perp$ in the box, then we conclude $\neg\varphi$
- This rule is denoted $\neg i$ and is read as $\neg$ introduction.
- Also known as Proof By Contradiction

An Example

- $p \to \neg p \vdash \neg p$

1.	$p \to \neg p$	premise
2.		

An Example

- $p \to \neg p \vdash \neg p$

1.	$p \to \neg p$	premise
2.	p	assumption
3.		

An Example

- $p \to \neg p \vdash \neg p$

1.	$p \to \neg p$	premise
2.	p	assumption
3.	$\neg p$	MP 1,2
4.		

An Example

- $p \to \neg p \vdash \neg p$

1.	$p \to \neg p$	premise
2.	p	assumption
3.	$\neg p$	MP 1,2
4.	$\bot$	$\bot i$ 2,3
5.	$\neg p$	$\neg i$ 2-4

The Last One

Law of the Excluded Middle (LEM)

$$\frac{}{\varphi \vee \neg\varphi}$$

Summary of Basic Rules

- $\wedge i$, $\wedge e_1$, $\wedge e_2$,
- $\neg\neg e$
- MP
- $\rightarrow i$
- $\vee i_1, \vee i_2, \vee e$
- Copy, $\neg i$ or PBC
- $\bot e, \bot i$

Derived Rules

- MT (derive using MP, $\bot i$ and $\neg i$)
- $\neg\neg i$ (derive using $\bot i$ and $\neg i$)
- LEM (derive using $\vee i_1, \bot i, \neg i, \vee i_2, \neg\neg e$)

The Proofs So Far

- So far, the “proof” we have seen is purely syntactic, no true/false meanings were attached

The Proofs So Far

- So far, the "proof" we have seen is purely syntactic, no true/false meanings were attached
- Intuitively, $p \to q \vdash \neg p \vee q$ makes sense because you think semantically. However, we never used any semantics so far.

The Proofs So Far

- So far, the “proof” we have seen is purely syntactic, no true/false meanings were attached
- Intuitively, $p \to q \vdash \neg p \vee q$ makes sense because you think semantically. However, we never used any semantics so far.
- Now we show that whatever can be proved makes sense semantically too.

Semantics

- Each propositional variable is assigned values true/false. Truth tables for each of the operators $\vee, \wedge, \neg, \rightarrow$ to determine truth values of complex formulae.

Semantics

- Each propositional variable is assigned values true/false. Truth tables for each of the operators $\vee, \wedge, \neg, \rightarrow$ to determine truth values of complex formulae.
- $\varphi_1, \ldots, \varphi_n \models \psi$ iff whenever $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ. $\models$ is read as semantically entails
  - Recall $\vdash$, and compare with $\models$

Semantics

- Each propositional variable is assigned values true/false. Truth tables for each of the operators $\vee, \wedge, \neg, \rightarrow$ to determine truth values of complex formulae.
- $\varphi_1, \ldots, \varphi_n \models \psi$ iff whenever $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ. $\models$ is read as semantically entails
  - Recall $\vdash$, and compare with $\models$
- Formulae φ and ψ are provably equivalent iff $\varphi \vdash \psi$ and $\psi \vdash \varphi$

Semantics

- Each propositional variable is assigned values true/false. Truth tables for each of the operators $\vee, \wedge, \neg, \rightarrow$ to determine truth values of complex formulae.
- $\varphi_1, \ldots, \varphi_n \models \psi$ iff whenever $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ. $\models$ is read as semantically entails
  - Recall $\vdash$, and compare with $\models$
- Formulae φ and ψ are provably equivalent iff $\varphi \vdash \psi$ and $\psi \vdash \varphi$
- Formulae φ and ψ are semantically equivalent iff $\varphi \models \psi$ and $\psi \models \varphi$

CS 228 : Logic in Computer Science

Krishna. S

The Proofs So Far

- So far, the "proof" we have seen is purely syntactic, no true/false meanings were attached

The Proofs So Far

- So far, the “proof” we have seen is purely syntactic, no true/false meanings were attached
- Intuitively, $p \rightarrow q \vdash \neg p \vee q$ makes sense because you think semantically. However, we never used any semantics so far.

The Proofs So Far

- So far, the “proof” we have seen is purely syntactic, no true/false meanings were attached
- Intuitively, $p \rightarrow q \vdash \neg p \vee q$ makes sense because you think semantically. However, we never used any semantics so far.
- Now we show that whatever can be proved makes sense semantically too.

Semantics

- Each propositional variable is assigned values true/false. Truth tables for each of the operators $\vee, \wedge, \neg, \rightarrow$ to determine truth values of complex formulae.

Semantics

- Each propositional variable is assigned values true/false. Truth tables for each of the operators $\vee, \wedge, \neg, \rightarrow$ to determine truth values of complex formulae.
- $\varphi_1, \ldots, \varphi_n \models \psi$ iff whenever $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ. $\models$ is read as semantically entails

Semantics

- Each propositional variable is assigned values true/false. Truth tables for each of the operators $\vee, \wedge, \neg, \rightarrow$ to determine truth values of complex formulae.
- $\varphi_1, \ldots, \varphi_n \models \psi$ iff whenever $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ. $\models$ is read as semantically entails
- Two formulae φ and ψ are provably equivalent iff $\varphi \vdash \psi$ and $\psi \vdash \varphi$

Semantics

- Each propositional variable is assigned values true/false. Truth tables for each of the operators $\vee, \wedge, \neg, \rightarrow$ to determine truth values of complex formulae.
- $\varphi_1, \ldots, \varphi_n \models \psi$ iff whenever $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ. $\models$ is read as semantically entails
- Two formulae φ and ψ are provably equivalent iff $\varphi \vdash \psi$ and $\psi \vdash \varphi$
- Two formulae φ and ψ are semantically equivalent iff $\varphi \models \psi$ and $\psi \models \varphi$

Soundness of Propositional Logic

$\varphi_1, \ldots, \varphi_n \vdash \psi \Rightarrow \varphi_1, \ldots, \varphi_n \models \psi$

Whenever ψ can be proved from $\varphi_1, \ldots, \varphi_n$, then ψ evaluates to true whenever $\varphi_1, \ldots, \varphi_n$ evaluate to true

Soundness

- Assume $\varphi_1, \ldots, \varphi_n \vdash \psi$.

Soundness

- Assume $\varphi_1, \ldots, \varphi_n \vdash \psi$.
- There is some proof (of length k lines) that yields ψ. Induct on k.

Soundness

- Assume $\varphi_1, \ldots, \varphi_n \vdash \psi$.
- There is some proof (of length k lines) that yields ψ. Induct on k.
- When $k = 1$, there is only one line in the proof, say φ, which is the premise. Then we have $\varphi \vdash \varphi$, since φ is given. But then we also have $\varphi \models \varphi$.

Soundness

- Assume $\varphi_1, \ldots, \varphi_n \vdash \psi$.
- There is some proof (of length k lines) that yields ψ. Induct on k.
- When $k = 1$, there is only one line in the proof, say φ, which is the premise. Then we have $\varphi \vdash \varphi$, since φ is given. But then we also have $\varphi \models \varphi$.
- Assume that whenever $\varphi_1, \ldots, \varphi_n \vdash \psi$ using a proof of length $\leqslant k - 1$, we have $\varphi_1, \ldots, \varphi_n \models \psi$.

Soundness

- Assume $\varphi_1, \ldots, \varphi_n \vdash \psi$.
- There is some proof (of length k lines) that yields ψ. Induct on k.
- When $k = 1$, there is only one line in the proof, say φ, which is the premise. Then we have $\varphi \vdash \varphi$, since φ is given. But then we also have $\varphi \models \varphi$.
- Assume that whenever $\varphi_1, \ldots, \varphi_n \vdash \psi$ using a proof of length $\leqslant k - 1$, we have $\varphi_1, \ldots, \varphi_n \models \psi$.
- Consider now a proof with k lines.

Soundness : Case $\wedge i$

- How did we arrive at ψ? Which proof rule gave ψ as the last line?

Soundness : Case $\wedge i$

- How did we arrive at ψ? Which proof rule gave ψ as the last line?
- Assume ψ was obtained using $\wedge i$. Then ψ is of the form $\psi_1 \wedge \psi_2$.

Soundness : Case $\wedge i$

- How did we arrive at ψ? Which proof rule gave ψ as the last line?
- Assume ψ was obtained using $\wedge i$. Then ψ is of the form $\psi_1 \wedge \psi_2$.
- ψ_1 and ψ_2 were proved earlier, say in lines $k_1, k_2 < k$.

Soundness : Case $\wedge i$

- How did we arrive at ψ? Which proof rule gave ψ as the last line?
- Assume ψ was obtained using $\wedge i$. Then ψ is of the form $\psi_1 \wedge \psi_2$.
- ψ_1 and ψ_2 were proved earlier, say in lines $k_1, k_2 < k$.
- We have the shorter proofs $\varphi_1, \ldots, \varphi_n \vdash \psi_1$ and $\varphi_1, \ldots, \varphi_n \vdash \psi_2$

Soundness : Case $\wedge i$

- How did we arrive at ψ? Which proof rule gave ψ as the last line?
- Assume ψ was obtained using $\wedge i$. Then ψ is of the form $\psi_1 \wedge \psi_2$.
- ψ_1 and ψ_2 were proved earlier, say in lines $k_1, k_2 < k$.
- We have the shorter proofs $\varphi_1, \ldots, \varphi_n \vdash \psi_1$ and $\varphi_1, \ldots, \varphi_n \vdash \psi_2$
- By inductive hypothesis, we have $\varphi_1, \ldots, \varphi_n \models \psi_1$ and $\varphi_1, \ldots, \varphi_n \models \psi_2$. By semantics, we have $\varphi_1, \ldots, \varphi_n \models \psi_1 \wedge \psi_2$.

Soundness : Case $\rightarrow i$

- Assume ψ was obtained using $\rightarrow i$. Then ψ is of the form $\psi_1 \rightarrow \psi_2$.

Soundness : Case $\rightarrow i$

- Assume ψ was obtained using $\rightarrow i$. Then ψ is of the form $\psi_1 \rightarrow \psi_2$.
- A box starting with ψ_1 was opened at some line $k_1 < k$.

Soundness : Case $\rightarrow i$

- Assume ψ was obtained using $\rightarrow i$. Then ψ is of the form $\psi_1 \rightarrow \psi_2$.
- A box starting with ψ_1 was opened at some line $k_1 < k$.
- The last line in the box was ψ_2.

Soundness : Case $\rightarrow i$

- Assume ψ was obtained using $\rightarrow i$. Then ψ is of the form $\psi_1 \rightarrow \psi_2$.
- A box starting with ψ_1 was opened at some line $k_1 < k$.
- The last line in the box was ψ_2.
- The line just after the box was ψ.

Soundness : Case $\rightarrow i$

- Assume ψ was obtained using $\rightarrow i$. Then ψ is of the form $\psi_1 \rightarrow \psi_2$.
- A box starting with ψ_1 was opened at some line $k_1 < k$.
- The last line in the box was ψ_2.
- The line just after the box was ψ.
- Consider adding ψ_1 in the premises along with $\varphi_1, \ldots, \varphi_n$. Then we will get a proof $\varphi_1, \ldots, \varphi_n, \psi_1 \vdash \psi_2$, of length $k - 1$. By inductive hypothesis, $\varphi_1, \ldots, \varphi_n, \psi_1 \models \psi_2$. By semantics, this is same as $\varphi_1, \ldots, \varphi_n \models \psi_1 \rightarrow \psi_2$

Soundness : Case $\rightarrow i$

- Assume ψ was obtained using $\rightarrow i$. Then ψ is of the form $\psi_1 \rightarrow \psi_2$.
- A box starting with ψ_1 was opened at some line $k_1 < k$.
- The last line in the box was ψ_2.
- The line just after the box was ψ.
- Consider adding ψ_1 in the premises along with $\varphi_1, \ldots, \varphi_n$. Then we will get a proof $\varphi_1, \ldots, \varphi_n, \psi_1 \vdash \psi_2$, of length $k - 1$. By inductive hypothesis, $\varphi_1, \ldots, \varphi_n, \psi_1 \models \psi_2$. By semantics, this is same as $\varphi_1, \ldots, \varphi_n \models \psi_1 \rightarrow \psi_2$
- The equivalence of $\varphi_1, \ldots, \varphi_n \vdash \psi_1 \rightarrow \psi_2$ and $\varphi_1, \ldots, \varphi_n, \psi_1 \vdash \psi_2$ gives the proof.

Soundness : Other cases

Completeness

$$\varphi_1, \ldots, \varphi_n \models \psi \Rightarrow \varphi_1, \ldots, \varphi_n \vdash \psi$$

Whenever $\varphi_1, \ldots, \varphi_n$ semantically entail ψ, then ψ can be proved from $\varphi_1, \ldots, \varphi_n$. The proof rules are complete

Completeness : 3 steps

- Given $\varphi_1, \ldots, \varphi_n \models \psi$

Completeness : 3 steps

- Given $\varphi_1, \ldots, \varphi_n \models \psi$
- Step 1: Show that $\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$

Completeness : 3 steps

- Given $\varphi_1, \ldots, \varphi_n \models \psi$
- Step 1: Show that $\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$
- Step 2: Show that $\vdash \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$

Completeness : 3 steps

- Given $\varphi_1, \ldots, \varphi_n \models \psi$
- Step 1: Show that $\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$
- Step 2: Show that $\vdash \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$
- Step 3: Show that $\varphi_1, \ldots, \varphi_n \vdash \psi$

Completeness : Step 1

- Assume $\varphi_1, \ldots, \varphi_n \models \psi$. Whenever all of $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ.

Completeness : Step 1

- Assume $\varphi_1, \ldots, \varphi_n \models \psi$. Whenever all of $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ.
- If $\not\models \varphi_1 \to (\varphi_2 \to (\ldots (\varphi_n \to \psi) \ldots))$, then ψ evaluates to false when all of $\varphi_1, \ldots, \varphi_n$ evaluate to true, a contradiction.

Completeness : Step 1

- Assume $\varphi_1, \ldots, \varphi_n \models \psi$. Whenever all of $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ.
- If $\not\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$, then ψ evaluates to false when all of $\varphi_1, \ldots, \varphi_n$ evaluate to true, a contradiction.
- Hence, $\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$.

Completeness : Step 2

- Given $\models \psi$, show that $\vdash \psi$

Completeness : Step 2

- Given $\models \psi$, show that $\vdash \psi$
- Assume $p_1, \ldots, p_n$ are the propositional variables in ψ. We know that all the 2^n assignments of values to $p_1, \ldots, p_n$ make ψ true.

Completeness : Step 2

- Given $\models \psi$, show that $\vdash \psi$
- Assume $p_1, \ldots, p_n$ are the propositional variables in ψ. We know that all the 2^n assignments of values to $p_1, \ldots, p_n$ make ψ true.
- Using this insight, we have to give a proof of ψ.

Completeness : Step 2

Truth Table to Proof

Let φ be a formula with variables $p_1, \ldots, p_n$. Let $\mathcal{T}$ be the truth table of φ, and let l be a line number in $\mathcal{T}$. Let $\hat{p}_i$ represent p_i if p_i is assigned true in line l, and let it denote $\neg p_i$ if p_i is assigned false in line l. Then

1. $\hat{p}_1, \ldots, \hat{p}_n \vdash \varphi$ if φ evaluates to true in line l
2. $\hat{p}_1, \ldots, \hat{p}_n \vdash \neg\varphi$ if φ evaluates to false in line l

CS 228 : Logic in Computer Science

Krishna. S

Completeness

$\varphi_1, \ldots, \varphi_n \models \psi \Rightarrow \varphi_1, \ldots, \varphi_n \vdash \psi$

Whenever $\varphi_1, \ldots, \varphi_n$ semantically entail ψ, then ψ can be proved from $\varphi_1, \ldots, \varphi_n$. The proof rules are complete

Completeness : 3 steps

- Given $\varphi_1, \ldots, \varphi_n \models \psi$
- Step 1: Show that $\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$
- Step 2: Show that $\vdash \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$
- Step 3: Show that $\varphi_1, \ldots, \varphi_n \vdash \psi$

Completeness : Step 1

- Assume $\varphi_1, \ldots, \varphi_n \models \psi$. Whenever all of $\varphi_1, \ldots, \varphi_n$ evaluate to true, so does ψ.
- If $\not\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$, then ψ evaluates to false when all of $\varphi_1, \ldots, \varphi_n$ evaluate to true, a contradiction.
- Hence, $\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$.

Completeness : Step 2

- Given $\models \psi$, show that $\vdash \psi$
- Assume $p_1, \ldots, p_n$ are the propositional variables in ψ. We know that all the 2^n assignments of values to $p_1, \ldots, p_n$ make ψ true.
- Using this insight, we have to give a proof of ψ.

Completeness : Step 2

Truth Table to Proof

Let φ be a formula with variables $p_1, \ldots, p_n$. Let $\mathcal{T}$ be the truth table of φ, and let l be a line number in $\mathcal{T}$. Let $\hat{p}_i$ represent p_i if p_i is assigned true in line l, and let it denote $\neg p_i$ if p_i is assigned false in line l. Then

1. $\hat{p}_1, \ldots, \hat{p}_n \vdash \varphi$ if φ evaluates to true in line l
2. $\hat{p}_1, \ldots, \hat{p}_n \vdash \neg\varphi$ if φ evaluates to false in line l

Truth Table to Proof

- Structural Induction on φ.

Truth Table to Proof

- Structural Induction on φ.
- Base : If $\varphi = p$, a proposition, then we have $p \vdash p$ and $\neg p \vdash \neg p$.

Truth Table to Proof

- Structural Induction on φ.
- Base : If $\varphi = p$, a proposition, then we have $p \vdash p$ and $\neg p \vdash \neg p$.
- Assume for formulae of size $\leqslant k - 1$ (size=height of the parse tree). What is a parse tree?

Truth Table to Proof

- Structural Induction on φ.
- Base : If $\varphi = p$, a proposition, then we have $p \vdash p$ and $\neg p \vdash \neg p$.
- Assume for formulae of size $\leqslant k - 1$ (size=height of the parse tree). What is a parse tree?
- Case Negation : $\varphi = \neg\varphi_1$

Truth Table to Proof

- Structural Induction on φ.
- Base : If $\varphi = p$, a proposition, then we have $p \vdash p$ and $\neg p \vdash \neg p$.
- Assume for formulae of size $\leqslant k - 1$ (size=height of the parse tree). What is a parse tree?
- Case Negation : $\varphi = \neg\varphi_1$
  - Assume φ evaluates to true in line l of $\mathcal{T}$. Then φ_1 evaluates to false in line l. By inductive hypothesis, $\hat{p}_1, \ldots, \hat{p}_n \vdash \neg\varphi_1$.

Truth Table to Proof

- Structural Induction on φ.
- Base : If $\varphi = p$, a proposition, then we have $p \vdash p$ and $\neg p \vdash \neg p$.
- Assume for formulae of size $\leqslant k - 1$ (size=height of the parse tree). What is a parse tree?
- Case Negation : $\varphi = \neg\varphi_1$
  - Assume φ evaluates to true in line l of $\mathcal{T}$. Then φ_1 evaluates to false in line l. By inductive hypothesis, $\hat{p}_1, \dots, \hat{p}_n \vdash \neg\varphi_1$.
  - Assume φ evaluates to false in line l of $\mathcal{T}$. Then φ_1 evaluates to true in line l. By inductive hypothesis, $\hat{p}_1, \dots, \hat{p}_n \vdash \varphi_1$. Use the $\neg\neg i$ rule to obtain a proof of $\hat{p}_1, \dots, \hat{p}_n \vdash \neg\neg\varphi_1$.

Truth Table to Proof

- Case $\rightarrow$: $\varphi = \varphi_1 \rightarrow \varphi_2$.

Truth Table to Proof

- Case $\rightarrow$: $\varphi = \varphi_1 \rightarrow \varphi_2$.
  - If φ evaluates to false in line l, then φ_1 evaluates to true and φ_2 to false. Let $\{q_1, \ldots, q_k\}$ be the variables of φ_1 and let $\{r_1, \ldots, r_j\}$ be the variables in φ_2. $\{q_1, \ldots, q_k\} \cup \{r_1, \ldots, r_j\} = \{p_1, \ldots, p_n\}$.

Truth Table to Proof

- Case $\rightarrow$: $\varphi = \varphi_1 \rightarrow \varphi_2$.
  - If φ evaluates to false in line l, then φ_1 evaluates to true and φ_2 to false. Let $\{q_1, \ldots, q_k\}$ be the variables of φ_1 and let $\{r_1, \ldots, r_j\}$ be the variables in φ_2. $\{q_1, \ldots, q_k\} \cup \{r_1, \ldots, r_j\} = \{p_1, \ldots, p_n\}$.
  - By inductive hypothesis, $\hat{q}_1, \ldots, \hat{q}_k \models \varphi_1$ and $\hat{r}_1, \ldots, \hat{r}_j \models \neg\varphi_2$. Then, $\hat{p}_1, \ldots, \hat{p}_n \models \varphi_1 \wedge \neg\varphi_2$.

Truth Table to Proof

- Case $\to$: $\varphi = \varphi_1 \to \varphi_2$.
  - If φ evaluates to false in line l, then φ_1 evaluates to true and φ_2 to false. Let $\{q_1, \ldots, q_k\}$ be the variables of φ_1 and let $\{r_1, \ldots, r_j\}$ be the variables in φ_2. $\{q_1, \ldots, q_k\} \cup \{r_1, \ldots, r_j\} = \{p_1, \ldots, p_n\}$.
  - By inductive hypothesis, $\hat{q}_1, \ldots, \hat{q}_k \models \varphi_1$ and $\hat{r}_1, \ldots, \hat{r}_j \models \neg\varphi_2$. Then, $\hat{p}_1, \ldots, \hat{p}_n \models \varphi_1 \wedge \neg\varphi_2$.
  - Prove that $\varphi_1 \wedge \neg\varphi_2 \vdash \neg(\varphi_1 \to \varphi_2)$.

Truth Table to Proof

- Case $\to$: $\varphi = \varphi_1 \to \varphi_2$.

Truth Table to Proof

- Case $\rightarrow$: $\varphi = \varphi_1 \rightarrow \varphi_2$.
  - If φ evaluates to true in line l, then there are 3 possibilities. If both φ_1, φ_2 evaluate to true, then we have $\hat{p}_1, \ldots, \hat{p}_n \models \varphi_1 \wedge \varphi_2$. Proving $\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$, we are done.

Truth Table to Proof

- Case $\rightarrow$: $\varphi = \varphi_1 \rightarrow \varphi_2$.
  - If φ evaluates to true in line l, then there are 3 possibilities. If both φ_1, φ_2 evaluate to true, then we have $\hat{p}_1, \ldots, \hat{p}_n \models \varphi_1 \wedge \varphi_2$. Proving $\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$, we are done.
  - If both φ_1, φ_2 evaluate to false, then we have $\hat{p}_1, \ldots, \hat{p}_n \models \neg\varphi_1 \wedge \neg\varphi_2$. Proving $\neg\varphi_1 \wedge \neg\varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$, we are done.

Truth Table to Proof

- Case $\rightarrow$: $\varphi = \varphi_1 \rightarrow \varphi_2$.
  - If φ evaluates to true in line l, then there are 3 possibilities. If both φ_1, φ_2 evaluate to true, then we have $\hat{p}_1, \ldots, \hat{p}_n \models \varphi_1 \wedge \varphi_2$. Proving $\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$, we are done.
  - If both φ_1, φ_2 evaluate to false, then we have $\hat{p}_1, \ldots, \hat{p}_n \models \neg\varphi_1 \wedge \neg\varphi_2$. Proving $\neg\varphi_1 \wedge \neg\varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$, we are done.
  - Last, if φ_1 evaluates to false and φ_2 evaluates to true, then we have $\hat{p}_1, \ldots, \hat{p}_n \models \neg\varphi_1 \wedge \varphi_2$. Proving $\neg\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$, we are done.

Truth Table to Proof

- Prove the cases $\wedge, \vee$.

On An Example

We know $\models (p \wedge q) \rightarrow p$. Using this fact, show that $\vdash (p \wedge q) \rightarrow p$.

- $p, q \vdash (p \wedge q) \rightarrow p$
- $\neg p, q \vdash (p \wedge q) \rightarrow p$
- $p, \neg q \vdash (p \wedge q) \rightarrow p$
- $\neg p, \neg q \vdash (p \wedge q) \rightarrow p$

Now, combine the 4 proofs above to give a single proof for
$\vdash (p \wedge q) \rightarrow p$.

Completeness : Steps 2, 3

- Step 2: From $\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$, use LEM on all the propositional variables of $\varphi_1, \ldots, \varphi_n, \psi$ to obtain $\vdash \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$.

Completeness : Steps 2, 3

- Step 2: From $\models \varphi_1 \to (\varphi_2 \to (\dots(\varphi_n \to \psi)\dots))$, use LEM on all the propositional variables of $\varphi_1, \dots, \varphi_n, \psi$ to obtain $\vdash \varphi_1 \to (\varphi_2 \to (\dots(\varphi_n \to \psi)\dots))$.
- Step 3: Take the proof $\vdash \varphi_1 \to (\varphi_2 \to (\dots(\varphi_n \to \psi)\dots))$. This proof has *n* nested boxes, the *i*th box opening with the assumption φ_i. The last box closes with the last line ψ. Hence, the line immediately after the last box is $\varphi_n \to \psi$.

Completeness : Steps 2, 3

- Step 2: From $\models \varphi_1 \to (\varphi_2 \to (\dots(\varphi_n \to \psi)\dots))$, use LEM on all the propositional variables of $\varphi_1, \dots, \varphi_n, \psi$ to obtain $\vdash \varphi_1 \to (\varphi_2 \to (\dots(\varphi_n \to \psi)\dots))$.
- Step 3: Take the proof $\vdash \varphi_1 \to (\varphi_2 \to (\dots(\varphi_n \to \psi)\dots))$. This proof has n nested boxes, the ith box opening with the assumption φ_i. The last box closes with the last line ψ. Hence, the line immediately after the last box is $\varphi_n \to \psi$.
- In a similar way, the $(n-1)$th box has as its last line $\varphi_n \to \psi$, and hence, the line immediately after this box is $\varphi_{n-1} \to (\varphi_n \to \psi)$ and so on.

Completeness : Steps 2, 3

- Step 2: From $\models \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$, use LEM on all the propositional variables of $\varphi_1, \ldots, \varphi_n, \psi$ to obtain $\vdash \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$.
- Step 3: Take the proof $\vdash \varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots))$. This proof has n nested boxes, the ith box opening with the assumption φ_i. The last box closes with the last line ψ. Hence, the line immediately after the last box is $\varphi_n \to \psi$.
- In a similar way, the $(n-1)$th box has as its last line $\varphi_n \to \psi$, and hence, the line immediately after this box is $\varphi_{n-1} \to (\varphi_n \to \psi)$ and so on.
- Add premises $\varphi_1, \ldots, \varphi_n$ on the top. Use MP on the premises, and the lines after boxes 1 to n in order to obtain ψ.

Summary

Propositional Logic is sound and complete.

Normal Forms

- A literal is a propositional variable p or its negation $\neg p$. These are referred to as positive and negative literals respectively.

Normal Forms

- A literal is a propositional variable p or its negation $\neg p$. These are referred to as positive and negative literals respectively.
- A formula F is in CNF if it is a conjunction of a disjunction of literals.

$$F = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} L_{i,j}$$

each $L_{i,j}$ is a literal.

Normal Forms

- A literal is a propositional variable p or its negation $\neg p$. These are referred to as positive and negative literals respectively.
- A formula F is in CNF if it is a conjunction of a disjunction of literals.
$$F = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} L_{i,j}$$
each $L_{i,j}$ is a literal.
- A formula F is in DNF if it is a disjunction of a conjunction of literals.
$$F = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} L_{i,j}$$
each $L_{i,j}$ is a literal.

Normal Forms

In the following, equivalent stands for semantically equivalent

Let F be a formula in CNF and let G be a formula in DNF. Then $\neg F$ is equivalent to a formula in DNF and $\neg G$ is equivalent to a formula in CNF.

Normal Forms

In the following, equivalent stands for semantically equivalent

Let F be a formula in CNF and let G be a formula in DNF. Then $\neg F$ is equivalent to a formula in DNF and $\neg G$ is equivalent to a formula in CNF.

Every formula F is equivalent to some formula F_1 in CNF and some formula F_2 in DNF.

CNF Algorithm

Given a formula F, $(x \to [\neg(y \vee z) \wedge \neg(y \to x)])$

- Replace all subformulae of the form $F \to G$ with $\neg F \vee G$, and all subformulae of the form $F \leftrightarrow G$ with $(\neg F \vee G) \wedge (\neg G \vee F)$. When there are no more occurrences of $\to, \leftrightarrow$, proceed to the next step.

CNF Algorithm

Given a formula F, $(x \rightarrow [\neg(y \vee z) \wedge \neg(y \rightarrow x)])$

- Replace all subformulae of the form $F \rightarrow G$ with $\neg F \vee G$, and all subformulae of the form $F \leftrightarrow G$ with $(\neg F \vee G) \wedge (\neg G \vee F)$. When there are no more occurrences of $\rightarrow, \leftrightarrow$, proceed to the next step.
- Get rid of all double negations : Replace all subformulae
  - $\neg\neg G$ with G,
  - $\neg(G \wedge H)$ with $\neg G \vee \neg H$
  - $\neg(G \vee H)$ with $\neg G \wedge \neg H$

  When there are no more such subformulae, proceed to the next step.

CNF Algorithm

Given a formula F, $(x \to [\neg(y \lor z) \land \neg(y \to x)])$

- Replace all subformulae of the form $F \to G$ with $\neg F \lor G$, and all subformulae of the form $F \leftrightarrow G$ with $(\neg F \lor G) \land (\neg G \lor F)$. When there are no more occurrences of $\to, \leftrightarrow$, proceed to the next step.
- Get rid of all double negations : Replace all subformulae
  - $\neg\neg G$ with G,
  - $\neg(G \land H)$ with $\neg G \lor \neg H$
  - $\neg(G \lor H)$ with $\neg G \land \neg H$

  When there are no more such subformulae, proceed to the next step.
- Distribute $\lor$ wherever possible.

The resultant formula F_1 is in CNF and is provably equivalent to F.
$[(\neg x \lor \neg y) \land (\neg x \lor \neg z)] \land [(\neg x \lor y) \land (\neg x \lor \neg x)]$

The Hardness of SAT

- Given a formula φ how to check if φ is satisfiable?
- Given a formula φ how to check if φ is unsatisfiable?
- SAT is NP-complete

Polynomial Time Formula Classes

Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- Programming languages Prolog and Datalog are based on Horn clauses in first order logic

Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- Programming languages Prolog and Datalog are based on Horn clauses in first order logic
- A formula F is a Horn formula if it is in CNF and every disjunction contains atmost one positive literal.

Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- Programming languages Prolog and Datalog are based on Horn clauses in first order logic
- A formula F is a Horn formula if it is in CNF and every disjunction contains atmost one positive literal.
- $p \wedge (\neg p \vee \neg q \vee r) \wedge (\neg a \vee \neg b)$ is Horn, but $a \vee b$ is not Horn.

Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- Programming languages Prolog and Datalog are based on Horn clauses in first order logic
- A formula F is a Horn formula if it is in CNF and every disjunction contains atmost one positive literal.
- $p \wedge (\neg p \vee \neg q \vee r) \wedge (\neg a \vee \neg b)$ is Horn, but $a \vee b$ is not Horn.
- A basic Horn formula is one which has no $\wedge$. Every Horn formula is a conjunction of basic Horn formulae.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.
- Basic Horn with no negative literals are of the form p and are written as $\top \rightarrow p$.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.
- Basic Horn with no negative literals are of the form p and are written as $\top \rightarrow p$.
- Basic Horn with no positive literals are written as $p \wedge q \wedge \cdots \wedge r \rightarrow \perp$.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.
- Basic Horn with no negative literals are of the form p and are written as $\top \rightarrow p$.
- Basic Horn with no positive literals are written as $p \wedge q \wedge \cdots \wedge r \rightarrow \bot$.
- Thus, a Horn formula is written as a conjunction of implications.

CS 228 : Logic in Computer Science

Krishna. S

Polynomial Time Formula Classes

Horn Formulae

- A formula F is a Horn formula if it is in CNF and every disjunction contains at most one positive literal.
- $p \wedge (\neg p \vee \neg q \vee r) \wedge (\neg a \vee \neg b)$ is Horn, but $a \vee b$ is not Horn.
- A basic Horn formula is one which has no $\wedge$. Every Horn formula is a conjunction of basic Horn formulae.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.
- Basic Horn with no negative literals are of the form p and are written as $\top \rightarrow p$.
- Basic Horn with no positive literals are written as $p \wedge q \wedge \cdots \wedge r \rightarrow \perp$.
- Thus, a Horn formula is written as a conjunction of implications.

The Horn Algorithm

Given a Horn formula H,

- Mark all occurrences of p, whenever $\top \to p$ is a subformula.

The Horn Algorithm

Given a Horn formula H,

- Mark all occurrences of p, whenever $\top \to p$ is a subformula.
- If there is a subformula of the form $(p_1 \wedge \cdots \wedge p_m) \to q$, where each p_i is marked, and q is not marked, mark q. Repeat this until there are no subformulae of this form and proceed to the next step.

The Horn Algorithm

Given a Horn formula H,

- Mark all occurrences of p, whenever $\top \to p$ is a subformula.
- If there is a subformula of the form $(p_1 \wedge \cdots \wedge p_m) \to q$, where each p_i is marked, and q is not marked, mark q. Repeat this until there are no subformulae of this form and proceed to the next step.
- Consider subformulae of the form $(p_1 \wedge \cdots \wedge p_m) \to \bot$. If there is one such subformula with all p_i marked, then say Unsat, otherwise say Sat.

An Example

$$(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$$

An Example

$(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$

- $(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$

An Example

$$(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$$

- $(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B)$.
- $(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B)$.

An Example

$(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B)$.

- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B)$.
- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B)$.
- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B)$.

An Example

$(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$

- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$
- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$
- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$
- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$

The Horn Algorithm

The Horn algorithm concludes Sat iff H is satisfiable.

Complexity of Horn

- Given a Horn formula ψ with *n* propositions, how many times do you have to read ψ?
- Step 1: Read once
- Step 2: Read atmost *n* times
- Step 3: Read once

2-CNF

- 2-CNF : CNF where each clause has at most 2 literals.

Resolution

- Resolution is a technique used to check if a formula in CNF is unsatisfiable.

Resolution

- Resolution is a technique used to check if a formula in CNF is unsatisfiable.
- CNF notation as set of sets : $(p \vee q) \wedge (\neg p \vee q) \wedge p$ represented as $\{\{p, q\}, \{\neg p, q\}, \{p\}\}$

Resolution

- Resolution is a technique used to check if a formula in CNF is unsatisfiable.
- CNF notation as set of sets : $(p \vee q) \wedge (\neg p \vee q) \wedge p$ represented as $\{\{p, q\}, \{\neg p, q\}, \{p\}\}$
- Let C_1, C_2 be two clauses. Assume $p \in C_1$ and $\neg p \in C_2$ for some literal p. Then the clause $R = (C_1 - \{p\}) \cup (C_2 - \{\neg p\})$ is a resolvent of C_1 and C_2.

Resolution

- Resolution is a technique used to check if a formula in CNF is unsatisfiable.
- CNF notation as set of sets : $(p \vee q) \wedge (\neg p \vee q) \wedge p$ represented as $\{\{p, q\}, \{\neg p, q\}, \{p\}\}$
- Let C_1, C_2 be two clauses. Assume $p \in C_1$ and $\neg p \in C_2$ for some literal p. Then the clause $R = (C_1 - \{p\}) \cup (C_2 - \{\neg p\})$ is a resolvent of C_1 and C_2.
- Let $C_1 = \{p_1, \neg p_2, p_3\}$ and $C_2 = \{p_2, \neg p_3, p_4\}$. As $p_3 \in C_1$ and $\neg p_3 \in C_2$, we can find the resolvent. The resolvent is $\{p_1, p_2, \neg p_2, p_4\}$.

Resolution

- Resolution is a technique used to check if a formula in CNF is unsatisfiable.
- CNF notation as set of sets : $(p \vee q) \wedge (\neg p \vee q) \wedge p$ represented as $\{\{p, q\}, \{\neg p, q\}, \{p\}\}$
- Let C_1, C_2 be two clauses. Assume $p \in C_1$ and $\neg p \in C_2$ for some literal p. Then the clause $R = (C_1 - \{p\}) \cup (C_2 - \{\neg p\})$ is a resolvent of C_1 and C_2.
- Let $C_1 = \{p_1, \neg p_2, p_3\}$ and $C_2 = \{p_2, \neg p_3, p_4\}$. As $p_3 \in C_1$ and $\neg p_3 \in C_2$, we can find the resolvent. The resolvent is $\{p_1, p_2, \neg p_2, p_4\}$.
- Resolvent not unique : $\{p_1, p_3, \neg p_3, p_4\}$ is also a resolvent.

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)
- Let F be a formula in CNF, and let C be a clause in F. Then $F \vdash C$ (Prove!)

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)
- Let F be a formula in CNF, and let C be a clause in F. Then $F \vdash C$ (Prove!)
- Let F be a formula in CNF. Let R be a resolvent of two clauses of F. Then $F \vdash R$ (Prove!)

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is unsatisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is unsatisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is unsatisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$
- $Res^0(F) = F$, there are finitely many clauses that can be derived from F.

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is unsatisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$
- $Res^0(F) = F$, there are finitely many clauses that can be derived from F.
- There is some $m \geqslant 0$ such that $Res^m(F) = Res^{m+1}(F)$. Denote it by $Res^*(F)$.

Example

Let $F = \{\{p_1, p_2, \neg p_3\}, \{\neg p_2, p_3\}\}$.

- $Res^0(F) = F$

Example

Let $F = \{\{p_1, p_2, \neg p_3\}, \{\neg p_2, p_3\}\}$.

- $Res^0(F) = F$
- $Res^1(F) = F \cup \{p_1, p_2, \neg p_2\} \cup \{p_1, \neg p_3, p_3\}$.

Example

Let $F = \{\{p_1, p_2, \neg p_3\}, \{\neg p_2, p_3\}\}$.

- $Res^0(F) = F$
- $Res^1(F) = F \cup \{p_1, p_2, \neg p_2\} \cup \{p_1, \neg p_3, p_3\}$.
- $Res^2(F) = Res^1(F) \cup \{p_1, p_2, \neg p_3\} \cup \{p_1, p_3, \neg p_2\}$

CS 228 : Logic in Computer Science

Krishna. S

Resolution

- Given a propositional logic formula φ, is it unsatisfiable?

Resolution

- Given a propositional logic formula φ, is it unsatisfiable?
- How does a solver do it?
- Assume it is in CNF

Resolution

- Let C_1, C_2 be two clauses. Assume $p \in C_1$ and $\neg p \in C_2$ for some literal p.

Resolution

- Let C_1, C_2 be two clauses. Assume $p \in C_1$ and $\neg p \in C_2$ for some literal p. Then the clause $R = (C_1 - \{p\}) \cup (C_2 - \{\neg p\})$ is a resolvent of C_1 and C_2.
- Let $C_1 = \{p_1, \neg p_2, p_3\}$ and $C_2 = \{p_2, \neg p_3, p_4\}$. As $p_3 \in C_1$ and $\neg p_3 \in C_2$, we can find the resolvent. The resolvent is $\{p_1, p_2, \neg p_2, p_4\}$.
- Resolvent not unique : $\{p_1, p_3, \neg p_3, p_4\}$ is also a resolvent.

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)
- Let F be a formula in CNF, and let C be a clause in F. Then $F \vdash C$ (Prove!)

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)
- Let F be a formula in CNF, and let C be a clause in F. Then $F \vdash C$ (Prove!)
- Let F be a formula in CNF. Let R be a resolvent of two clauses of F. Then $F \vdash R$ (Prove!)

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is unsatisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is unsatisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is unsatisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$
- $Res^0(F) = F$, there are finitely many clauses that can be derived from F.

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is unsatisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$
- $Res^0(F) = F$, there are finitely many clauses that can be derived from F.
- There is some $m \geqslant 0$ such that $Res^m(F) = Res^{m+1}(F)$. Denote it by $Res^*(F)$.

Example

Let $F = \{\{p_1, p_2, \neg p_3\}, \{\neg p_2, p_3\}\}$.

- $Res^0(F) = F$

Example

Let $F = \{\{p_1, p_2, \neg p_3\}, \{\neg p_2, p_3\}\}$.

- $Res^0(F) = F$
- $Res^1(F) = F \cup \{p_1, p_2, \neg p_2\} \cup \{p_1, \neg p_3, p_3\}$.

Example

Let $F = \{\{p_1, p_2, \neg p_3\}, \{\neg p_2, p_3\}\}$.

- $Res^0(F) = F$
- $Res^1(F) = F \cup \{p_1, p_2, \neg p_2\} \cup \{p_1, \neg p_3, p_3\}$.
- $Res^2(F) = Res^1(F) \cup \{p_1, p_2, \neg p_3\} \cup \{p_1, p_3, \neg p_2\}$

Resolution

Let F be a formula in CNF. If $\emptyset \in \mathit{Res}^*(F)$, then F is unsatisfiable.

- If $\emptyset \in \mathit{Res}^*(F)$. Then $\emptyset \in \mathit{Res}^n(F)$ for some n.

Resolution

Let F be a formula in CNF. If $\emptyset \in Res^*(F)$, then F is unsatisfiable.

- If $\emptyset \in Res^*(F)$. Then $\emptyset \in Res^n(F)$ for some n.
- Since $\emptyset \notin Res^0(F)$ ($\emptyset$ is not a clause), there is an $m > 0$ such that $\emptyset \notin Res^m(F)$ and $\emptyset \in Res^{m+1}(F)$.

Resolution

Let F be a formula in CNF. If $\emptyset \in Res^*(F)$, then F is unsatisfiable.

- If $\emptyset \in Res^*(F)$. Then $\emptyset \in Res^n(F)$ for some n.
- Since $\emptyset \notin Res^0(F)$ ($\emptyset$ is not a clause), there is an $m > 0$ such that $\emptyset \notin Res^m(F)$ and $\emptyset \in Res^{m+1}(F)$.
- Then $\{p\}, \{\neg p\} \in Res^m(F)$. By the rules of resolution, we have $F \vdash p, \neg p$, and thus $F \vdash \bot$. Hence, F is unsatisfiable.

Resolution

Prove the converse: F is unsatisfiable implies $\emptyset \in \mathit{Res}^*(F)$.

(Discuss substitution before the proof)

Resolution

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.

Resolution

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$

Resolution

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$
- Induct on the number n of propositional variables that occur in F.

Resolution

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$
- Induct on the number n of propositional variables that occur in F.
- If $n = 1$, then the possible clauses are p, $\neg p$ and $p \vee \neg p$. The third one is ruled out, by assumption.

Resolution

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$
- Induct on the number n of propositional variables that occur in F.
- If $n = 1$, then the possible clauses are p, $\neg p$ and $p \vee \neg p$. The third one is ruled out, by assumption.
- If $F = \{\{p\}\}$ or $F = \{\{\neg p\}\}$, F is satisfiable.

Resolution

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$
- Induct on the number n of propositional variables that occur in F.
- If $n = 1$, then the possible clauses are p, $\neg p$ and $p \vee \neg p$. The third one is ruled out, by assumption.
- If $F = \{\{p\}\}$ or $F = \{\{\neg p\}\}$, F is satisfiable.
- Hence, $F = \{\{p\}, \{\neg p\}\}$. Clearly, $\emptyset \in Res(F)$.

Resolution

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.

Resolution

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.
- Let F have $n+1$ variables $p_1, \ldots, p_{n+1}$.

Resolution

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.
- Let F have $n+1$ variables $p_1, \ldots, p_{n+1}$.

  - Let G_0 be the conjunction of all C_i in F such that $\neg p_{n+1} \notin C_i$.
  - Let G_1 be the conjunction of all C_i in F such that $p_{n+1} \notin C_i$.

Resolution

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.
- Let F have $n+1$ variables $p_1, \ldots, p_{n+1}$.

  - Let G_0 be the conjunction of all C_i in F such that $\neg p_{n+1} \notin C_i$.
  - Let G_1 be the conjunction of all C_i in F such that $p_{n+1} \notin C_i$.

- Clauses in F= Clauses in G_0 $\cup$ Clauses in G_1

Resolution

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.
- Let F have $n+1$ variables $p_1, \ldots, p_{n+1}$.

- Let G_0 be the conjunction of all C_i in F such that $\neg p_{n+1} \notin C_i$.
- Let G_1 be the conjunction of all C_i in F such that $p_{n+1} \notin C_i$.

- Clauses in F= Clauses in $G_0 \cup$ Clauses in G_1

- Let $F_0 = \{C_i - \{p_{n+1}\} \mid C_i \in G_0\}$
- Let $F_1 = \{C_i - \{\neg p_{n+1}\} \mid C_i \in G_1\}$

Resolution

Let $F = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_2, \neg p_3\}\}$ and $n = 2$.

- $G_0 = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}\}$, $G_1 = \{\{p_2\}, \{\neg p_2, \neg p_3\}\}$.
- $F_0 = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}$ and $F_1 = \{\{p_2\}, \{\neg p_2\}\}$
- If $p_{n+1} = \textit{false}$ in F, then F is equisatisfiable with F_0

Resolution

Let $F = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_2, \neg p_3\}\}$ and $n = 2$.

- $G_0 = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}\}$, $G_1 = \{\{p_2\}, \{\neg p_2, \neg p_3\}\}$.
- $F_0 = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}$ and $F_1 = \{\{p_2\}, \{\neg p_2\}\}$

- If $p_{n+1} = \mathit{false}$ in F, then F is equisatisfiable with F_0
- If $p_{n+1} = \mathit{true}$ in F, then F is equisatisfiable with F_1

Resolution

Let $F = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_2, \neg p_3\}\}$ and $n = 2$.

- $G_0 = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}\}$, $G_1 = \{\{p_2\}, \{\neg p_2, \neg p_3\}\}$.
- $F_0 = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}$ and $F_1 = \{\{p_2\}, \{\neg p_2\}\}$

- If $p_{n+1} = \mathit{false}$ in F, then F is equisatisfiable with F_0
- If $p_{n+1} = \mathit{true}$ in F, then F is equisatisfiable with F_1
- Hence F is satisfiable iff $F_0 \vee F_1$ is.

Resolution

Let $F = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_2, \neg p_3\}\}$ and $n = 2$.

- $G_0 = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}\}$, $G_1 = \{\{p_2\}, \{\neg p_2, \neg p_3\}\}$.
- $F_0 = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}$ and $F_1 = \{\{p_2\}, \{\neg p_2\}\}$

- If $p_{n+1} = \mathit{false}$ in F, then F is equisatisfiable with F_0
- If $p_{n+1} = \mathit{true}$ in F, then F is equisatisfiable with F_1
- Hence F is satisfiable iff $F_0 \vee F_1$ is.
- As F is unsatisfiable, F_0 and F_1 are both unsatisfiable.

Resolution

- By induction hypothesis, $\emptyset \in \mathit{Res}^*(F_0)$ and $\emptyset \in \mathit{Res}^*(F_1)$.

Resolution

- By induction hypothesis, $\emptyset \in Res^*(F_0)$ and $\emptyset \in Res^*(F_1)$.
- Hence, $\emptyset \in Res^*(G_0)$ or $\{p_{n+1}\} \in Res^*(G_0)$, and $\emptyset \in Res^*(G_1)$ or $\{\neg p_{n+1}\} \in Res^*(G_1)$.

Resolution

- By induction hypothesis, $\emptyset \in \mathit{Res}^*(F_0)$ and $\emptyset \in \mathit{Res}^*(F_1)$.
- Hence, $\emptyset \in \mathit{Res}^*(G_0)$ or $\{p_{n+1}\} \in \mathit{Res}^*(G_0)$, and $\emptyset \in \mathit{Res}^*(G_1)$ or $\{\neg p_{n+1}\} \in \mathit{Res}^*(G_1)$.
- If $\emptyset \in \mathit{Res}^*(G_0)$ or $\emptyset \in \mathit{Res}^*(G_1)$, then $\emptyset \in \mathit{Res}^*(F)$.

Resolution

- By induction hypothesis, $\emptyset \in Res^*(F_0)$ and $\emptyset \in Res^*(F_1)$.
- Hence, $\emptyset \in Res^*(G_0)$ or $\{p_{n+1}\} \in Res^*(G_0)$, and $\emptyset \in Res^*(G_1)$ or $\{\neg p_{n+1}\} \in Res^*(G_1)$.
- If $\emptyset \in Res^*(G_0)$ or $\emptyset \in Res^*(G_1)$, then $\emptyset \in Res^*(F)$.
- Else, $\{p_{n+1}\} \in Res^*(G_0)$ and $\{\neg p_{n+1}\} \in Res^*(G_1)$.

Resolution

- By induction hypothesis, $\emptyset \in \mathit{Res}^*(F_0)$ and $\emptyset \in \mathit{Res}^*(F_1)$.
- Hence, $\emptyset \in \mathit{Res}^*(G_0)$ or $\{p_{n+1}\} \in \mathit{Res}^*(G_0)$, and $\emptyset \in \mathit{Res}^*(G_1)$ or $\{\neg p_{n+1}\} \in \mathit{Res}^*(G_1)$.
- If $\emptyset \in \mathit{Res}^*(G_0)$ or $\emptyset \in \mathit{Res}^*(G_1)$, then $\emptyset \in \mathit{Res}^*(F)$.
- Else, $\{p_{n+1}\} \in \mathit{Res}^*(G_0)$ and $\{\neg p_{n+1}\} \in \mathit{Res}^*(G_1)$.
- Hence $\emptyset \in \mathit{Res}^*(F)$.

Resolution Summary

Given a formula ψ, convert it into CNF, say ζ. ψ is satisfiable iff $\emptyset \notin \mathit{Res}^*(\zeta)$.

- If ψ is unsat, we might get $\emptyset$ before reaching $\mathit{Res}^*(\zeta)$.
- If ψ is sat, then truth tables are faster : stop when some row evaluates to 1.

CS 228 : Logic in Computer Science

Krishna. S

Recap of Basics

- A formula φ is satisfiable when ...

Recap of Basics

- A formula φ is satisfiable when ...
- A formula φ is valid when ...

Recap of Basics

- A formula φ is satisfiable when ...
- A formula φ is valid when ...
- A formula φ is satisfiable iff $\neg\varphi$ is not valid.

Recap of Basics

- A formula φ is satisfiable when ...
- A formula φ is valid when ...
- A formula φ is satisfiable iff $\neg\varphi$ is not valid.
- Two formulae φ_1 and φ_2 are equivalent iff ...

Recap of Basics

- A formula φ is satisfiable when ...
- A formula φ is valid when ...
- A formula φ is satisfiable iff $\neg\varphi$ is not valid.
- Two formulae φ_1 and φ_2 are equivalent iff ...
- Two formulae φ_1 and φ_2 are equisatisfiable iff ...

Recap of Basics

- A formula φ is satisfiable when ...
- A formula φ is valid when ...
- A formula φ is satisfiable iff $\neg\varphi$ is not valid.
- Two formulae φ_1 and φ_2 are equivalent iff ...
- Two formulae φ_1 and φ_2 are equisatisfiable iff ...
- A disjunction of literals $L_1 \vee L_2 \vee \ldots L_n$ is valid iff ...

Recap of Basics

- A formula φ is satisfiable when ...
- A formula φ is valid when ...
- A formula φ is satisfiable iff $\neg\varphi$ is not valid.
- Two formulae φ_1 and φ_2 are equivalent iff ...
- Two formulae φ_1 and φ_2 are equisatisfiable iff ...
- A disjunction of literals $L_1 \vee L_2 \vee \ldots L_n$ is valid iff ...
- A conjunction of literals $L_1 \wedge L_2 \wedge \ldots L_n$ is satisfiable iff ...

Normal Forms : CNF Validity

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ be in CNF.

- Checking if φ is satisfiable is NP-complete.
- Checking if φ is valid is polynomial time. Why?
- Question raised in class : If validity is polytime, so should be satisfiability. Is this true?

CS 228 : Logic in Computer Science

Krishna. S

Normal Forms : CNF Validity

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ be in CNF.

- Checking if φ is satisfiable is NP-complete.
- Checking if φ is valid is polynomial time. Why?
- Question raised in class : If validity check is polynomial time, so should be satisfiability. Is this true?
- If φ is valid, it is indeed satisfiable
- If φ is not valid, then...?

Normal Forms : DNF Satisfiability

Let $\varphi = D_1 \vee D_2 \vee \cdots \vee D_n$ be in DNF.

- Checking if φ is valid is NP-complete. Why?
- Checking if φ is satisfiable is polynomial time. Why?

Normal Forms from Truth Tables

Assume you are given the truth table of a formula φ. Then it is very easy to obtain the equivalent CNF/DNF of φ.

Normal Forms from Truth Tables

Assume you are given the truth table of a formula φ. Then it is very easy to obtain the equivalent CNF/DNF of φ.

- Consider for example $\varphi = p \leftrightarrow q$.
- Truth table of φ : φ is false when $p = T, q = F$ and $p = F, q = T$.

Normal Forms from Truth Tables

Assume you are given the truth table of a formula φ. Then it is very easy to obtain the equivalent CNF/DNF of φ.

- Consider for example $\varphi = p \leftrightarrow q$.
- Truth table of φ : φ is false when $p = T, q = F$ and $p = F, q = T$.
- CNF equivalent is $(\neg p \vee q) \wedge (p \vee \neg q)$.

CNF to DNF Sizes

- $\varphi = (p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \ldots (p_n \vee q_n)$
- What is the equivalent DNF formula?

CNF to DNF Sizes

- $\varphi = (p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \ldots (p_n \vee q_n)$
- What is the equivalent DNF formula?
- $$\varphi' = \bigvee_{S \subseteq \{1,\ldots,n\}} (\bigwedge_{i \in S} p_i \wedge \bigwedge_{i \notin S} q_i)$$

CNF to DNF Sizes

- $\varphi = (p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \ldots (p_n \vee q_n)$
- What is the equivalent DNF formula?
- $$\varphi' = \bigvee_{S \subseteq \{1,\ldots,n\}} (\bigwedge_{i \in S} p_i \wedge \bigwedge_{i \notin S} q_i)$$
- Prove that any equivalent DNF formula has 2^n clauses

CNF to DNF Sizes

- $\varphi = (p_1 \lor q_1) \land (p_2 \lor q_2) \land \ldots (p_n \lor q_n)$
- What is the equivalent DNF formula?
- $$\varphi' = \bigvee_{S \subseteq \{1,\ldots,n\}} (\bigwedge_{i \in S} p_i \land \bigwedge_{i \notin S} q_i)$$
- Prove that any equivalent DNF formula has 2^n clauses
- Call an assignment *minimal* if it maps exactly one of p_i, q_i to 1

CNF to DNF Sizes

- $\varphi = (p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \ldots (p_n \vee q_n)$
- What is the equivalent DNF formula?
- $$\varphi' = \bigvee_{S \subseteq \{1,\ldots,n\}} (\bigwedge_{i \in S} p_i \wedge \bigwedge_{i \notin S} q_i)$$
- Prove that any equivalent DNF formula has 2^n clauses
- Call an assignment *minimal* if it maps exactly one of p_i, q_i to 1
- There are 2^n *minimal* assignments, satisfying clauses in φ'
- Show that no two *minimal* assignments satisfy the same clause of φ' (hence there must be 2^n clauses in φ')

CNF to DNF Sizes

- Let α and β be two minimal assignments such that $\alpha(p_i) \neq \beta(p_i)$ for $i \in \{1, \ldots, n\}$

CNF to DNF Sizes

- Let α and β be two minimal assignments such that $\alpha(p_i) \neq \beta(p_i)$ for $i \in \{1, \ldots, n\}$
- Define a new assignment $\mathsf{min}(\alpha, \beta)$ as a pointwise min of α, β
- $\mathsf{min}(\alpha, \beta)(p)$=$\mathsf{min}(\alpha(p), \beta(p))$ for each variable p with the assumption that $0 < 1$, 0 represents false and 1 represents true

CNF to DNF Sizes

- Let α and β be two minimal assignments such that $\alpha(p_i) \neq \beta(p_i)$ for $i \in \{1, \ldots, n\}$
- Define a new assignment $\mathsf{min}(\alpha, \beta)$ as a pointwise min of α, β
- $\mathsf{min}(\alpha, \beta)(p)$=$\mathsf{min}(\alpha(p), \beta(p))$ for each variable p with the assumption that $0 < 1$, 0 represents false and 1 represents true
- $\mathsf{min}(\alpha, \beta) \not\models p_i \vee q_i$, $\mathsf{min}(\alpha, \beta) \not\models \varphi'$

CNF to DNF Sizes

- Let α and β be two minimal assignments such that $\alpha(p_i) \neq \beta(p_i)$ for $i \in \{1, \ldots, n\}$
- Define a new assignment $\mathsf{min}(\alpha, \beta)$ as a pointwise min of α, β
- $\mathsf{min}(\alpha, \beta)(p)$=$\mathsf{min}(\alpha(p), \beta(p))$ for each variable p with the assumption that $0 < 1$, 0 represents false and 1 represents true
- $\mathsf{min}(\alpha, \beta) \not\models p_i \vee q_i$, $\mathsf{min}(\alpha, \beta) \not\models \varphi'$
- However, if $\alpha \models D_j$ and $\beta \models D_j$ for some clause D_j of φ', then $\mathsf{min}(\alpha, \beta) \models D_j$ and hence $\mathsf{min}(\alpha, \beta) \models \varphi'$, a contradiction.

Think of an example where DNF to CNF explodes.