

# Logic

November 7, 2024

# TOC

- 1 Lecture 00 Motivation.pdf
- 2 Lecture 01 Natural Deduction 1.pdf
- 3 Lecture 02 Natural Deduction 2.pdf
- 4 Lecture 03 Soundness and Completeness Of PL.pdf
- 5 Lecture 04 Normal Forms and Horn Clauses.pdf
- 6 Lecture 05 Resolution 1.pdf
- 7 Lecture 06 Resolution 2.pdf
- 8 Lecture 07 Recap.pdf

# TOC

- 9 Lecture 08 CNF to DNF.pdf
- 10 Lecture 09 DPLL.pdf
- 11 Lecture 10 FOL Syntax.pdf
- 12 Lecture 11 FOL Semantics.pdf
- 13 Lecture 12 FOL SAT.pdf
- 14 Lecture 13 FOL SAT Translation.pdf
- 15 Lecture 14 FOL RPF Skolemisation.pdf
- 16 Lecture 15 FOL Words.pdf

# TOC

- 17 Lecture 16 FOL FSM 01.pdf
- 18 Lecture 17 FOL FSM 02.pdf
- 19 Lecture 18 FOL NFA.pdf
- 20 Lecture 19 FOL NFA to DFA 1.pdf
- 21 Lecture 20 FOL NFA to DFA 2.pdf
- 22 Lecture 21 FOL NFA to DFA 3.pdf
- 23 Lecture 22 MSO.pdf
- 24 Lecture 23 MSO Regular Languages.pdf

# TOC

- 25 Lecture 24 Herbrand Theory.pdf
- 26 Lecture 25 Herbrand Theorem.pdf
- 27 Lecture 26 Herbrand Method.pdf
- 28 Lecture 27-29 LTL Transitions DBA and NBA.pdf
- 29 Lecture 27 Model Checking LTL.pdf
- 30 Lecture 28-29 LTL.pdf
- 31 Lecture 30.pdf
- 32 Lecture 31.pdf

# TOC

33 Lecture 32-34.pdf

34 Lecture 36.pdf

# Lecture 00 Motivation.pdf

# 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
- ▶  $(\frac{1}{3} + \frac{1}{5}) + (\frac{1}{5} + \frac{1}{7}) + (\frac{1}{11} + \frac{1}{13}) + (\frac{1}{17} + \frac{1}{19}) + \dots$  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
- ▶  $(\frac{1}{3} + \frac{1}{5}) + (\frac{1}{5} + \frac{1}{7}) + (\frac{1}{11} + \frac{1}{13}) + (\frac{1}{17} + \frac{1}{19}) + \dots$  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

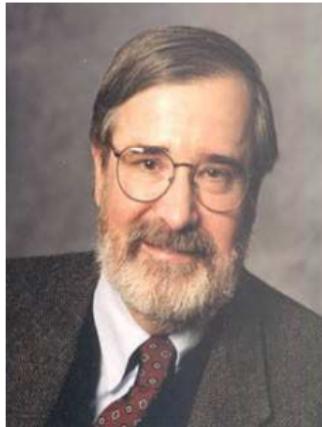
- ▶ Take a model
- ▶ Simulate it with certain inputs
- ▶ Observe what happens, and if this is desired

## 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



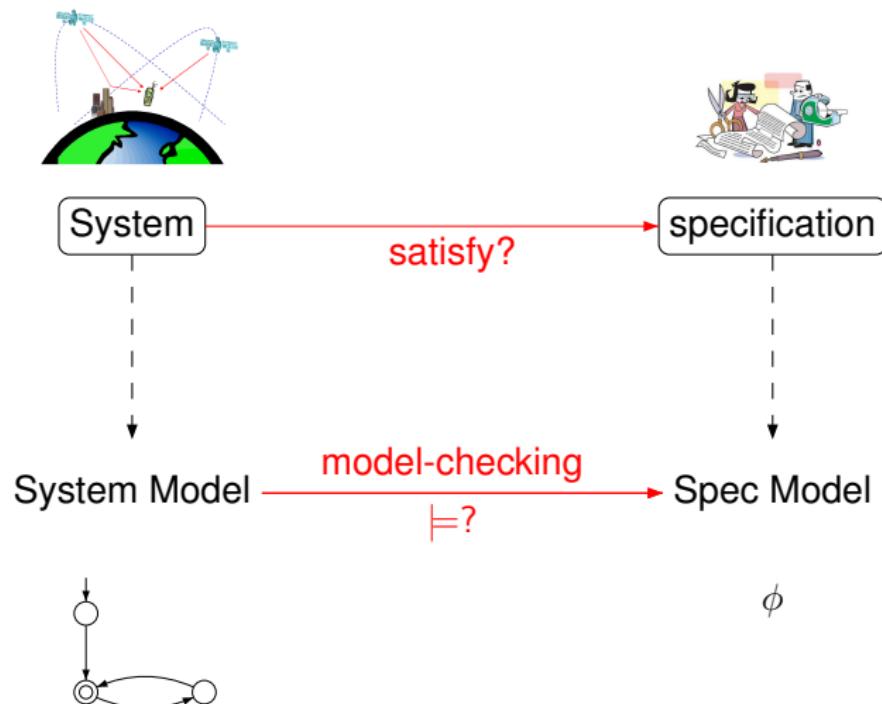
satisfy?



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

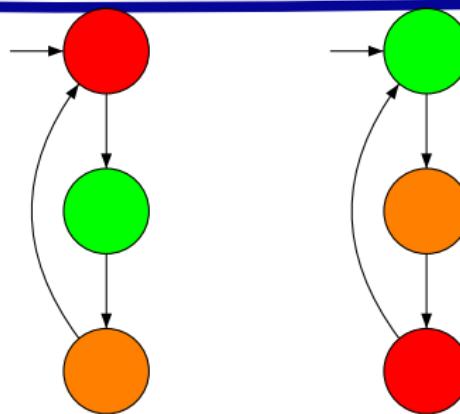
- ▶ Can the system reach a deadlock?
- ▶ Can two processes ever be together in a critical section?
- ▶ On termination, does a program provide correct output?

## 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  
 $\forall x (\neg(\text{green}_1(x) \wedge \text{green}_2(x)))$  or  $\square(\neg(\text{green}_1 \wedge \text{green}_2))$
2. The first traffic light is infinitely often green  
 $\forall x \exists y (x < y \wedge \text{green}_1(y))$  or  $\square\Diamond\text{green}_1$
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? → check next property (if any)
  - ▶ property violated?
    - ▶ analyse generated counter example by simulation
    - ▶ refine the model, design, property, ... and repeat entire procedure
  - ▶ out of memory? → 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)

Nevertheless:

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
  - ▶ Büchi automata and  $\omega$ -regular properties

# Relevant Topics

---

- ▶ How to express properties **succinctly**?
  - ▶ 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 **succinctly**?
  - ▶ 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

# Lecture 01 Natural Deduction 1.pdf

# 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  $\varphi$  in a logic  $L$ , is  $\varphi$  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  $\varphi$  in a logic  $L$ , is  $\varphi$  satisfiable?
- ▶ Q2: Given a formula  $\varphi$  in a logic  $L$ , is  $\varphi$  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  $\varphi$  in a logic  $L$ , is  $\varphi$  satisfiable?
- ▶ Q2: Given a formula  $\varphi$  in a logic  $L$ , is  $\varphi$  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  $\varphi$  in a logic  $L$ , is  $\varphi$  satisfiable?
- ▶ Q2: Given a formula  $\varphi$  in a logic  $L$ , is  $\varphi$  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  $\varphi$  in a logic  $L$ , is  $\varphi$  satisfiable?
- ▶ Q2: Given a formula  $\varphi$  in a logic  $L$ , is  $\varphi$  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  $\varphi$  in a logic  $L$ , is  $\varphi$  satisfiable?
- ▶ Q2: Given a formula  $\varphi$  in a logic  $L$ , is  $\varphi$  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  $\varphi$  in a logic  $L$ , is  $\varphi$  satisfiable?
- ▶ Q2: Given a formula  $\varphi$  in a logic  $L$ , is  $\varphi$  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  $p, q, \dots$

# Syntax

---

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

# Syntax

---

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

# Syntax

---

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

# Syntax

---

- ▶ Finite set of propositional variables  $p, q, \dots$
- ▶ Each of these can be true/false
- ▶ Combine propositions using  $\neg, \vee, \wedge, \rightarrow$
- ▶ Parentheses 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 parentheses,  $p \rightarrow q \rightarrow r$  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.  $\varphi = (R \wedge AliceOut \wedge \neg RG) \rightarrow AliceWet$

# Natural Deduction

---

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

# Natural Deduction

---

- ▶ If it rains, Alice is outside and does not have any raingear with her, she will get wet.  $\varphi = (R \wedge \text{AliceOut} \wedge \neg RG) \rightarrow \text{AliceWet}$
- ▶ It is raining, and Alice is outside, and is not wet.  
 $\psi = (R \wedge \text{AliceOut} \wedge \neg \text{AliceWet})$
- ▶ 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  $\chi$  valid? Is  $\chi$  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  $2 \times 2$  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:  $[P(2, 1, 1) \vee \dots]$
  - ▶ Row 3:  $[P(3, 1, 1) \vee \dots]$
  - ▶ Row 4:  $[P(4, 1, 1) \vee \dots]$

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

# Encoding as Propositional Satisfiability

---

Each  $2 \times 2$  block must contain all numbers 1-4

# Encoding as Propositional Satisfiability

---

Each  $2 \times 2$  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  $2 \times 2$  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:

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

- ▶ Lower left block contains all numbers 1-4:

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

- ▶ Lower right block contains all numbers 1-4:

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

# 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 Sudoku

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  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1$ ,

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1, M_2 \leftrightarrow \neg G_2,$

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1, M_2 \leftrightarrow \neg G_2, M_3 \leftrightarrow G_2$

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1, M_2 \leftrightarrow \neg G_2, M_3 \leftrightarrow G_2$
  - ▶  $\neg(M_1 \wedge M_2 \wedge M_3),$

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1, M_2 \leftrightarrow \neg G_2, M_3 \leftrightarrow G_2$
  - ▶  $\neg(M_1 \wedge M_2 \wedge M_3), M_1 \vee M_2 \vee M_3,$

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1, M_2 \leftrightarrow \neg G_2, M_3 \leftrightarrow G_2$
  - ▶  $\neg(M_1 \wedge M_2 \wedge M_3), M_1 \vee M_2 \vee M_3,$
  - ▶  $(\neg M_1 \wedge \neg M_2) \vee (\neg M_1 \wedge \neg M_3) \vee (\neg M_2 \wedge \neg M_3)$

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1, M_2 \leftrightarrow \neg G_2, M_3 \leftrightarrow G_2$
  - ▶  $\neg(M_1 \wedge M_2 \wedge M_3), M_1 \vee M_2 \vee M_3,$
  - ▶  $(\neg M_1 \wedge \neg M_2) \vee (\neg M_1 \wedge \neg M_3) \vee (\neg M_2 \wedge \neg M_3)$
  - ▶ Conjunct all these, and call the formula  $\varphi$ .

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1, M_2 \leftrightarrow \neg G_2, M_3 \leftrightarrow G_2$
  - ▶  $\neg(M_1 \wedge M_2 \wedge M_3), M_1 \vee M_2 \vee M_3,$
  - ▶  $(\neg M_1 \wedge \neg M_2) \vee (\neg M_1 \wedge \neg M_3) \vee (\neg M_2 \wedge \neg M_3)$
  - ▶ Conjunct all these, and call the formula  $\varphi$ .
  - ▶ Is there a unique satisfiable assignment for  $\varphi$ ?

# Solve Gold Rush

---

- ▶ Propositions  $M_1, M_2, M_3$  representing messages in boxes 1,2,3
- ▶ Propositions  $G_1, G_2, G_3$  representing gold in boxes 1,2,3
- ▶ Formalize what is given to you
  - ▶  $M_1 \leftrightarrow \neg G_1, M_2 \leftrightarrow \neg G_2, M_3 \leftrightarrow G_2$
  - ▶  $\neg(M_1 \wedge M_2 \wedge M_3), M_1 \vee M_2 \vee M_3,$
  - ▶  $(\neg M_1 \wedge \neg M_2) \vee (\neg M_1 \wedge \neg M_3) \vee (\neg M_2 \wedge \neg M_3)$
  - ▶ Conjunct all these, and call the formula  $\varphi$ .
  - ▶ Is there a unique satisfiable assignment for  $\varphi$ ?
  - ▶ For example, is  $M_1 = \text{true}$  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.  $\varphi = (R \wedge \text{AliceOut} \wedge \neg RG) \rightarrow \text{AliceWet}$
- ▶ It is raining, and Alice is outside, and is not wet.  
 $\psi = (R \wedge \text{AliceOut} \wedge \neg \text{AliceWet})$
- ▶ So, Alice has her rain gear with her.  $RG$
- ▶ Thus,  $\chi = \varphi \wedge \psi \rightarrow RG$ .
- ▶ Given  $\varphi, \psi$ , can we “prove”  $RG$ ?

# A Proof Engine

---

- ▶ Given a formula  $\varphi$  in propositional logic, how to “prove”  $\varphi$  if  $\varphi$  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.
- ▶  $\varphi_1, \dots, \varphi_n \vdash \psi$  : This is called a **sequent**.  $\varphi_1, \dots, \varphi_n$  are **premises**, and  $\psi$ , the **conclusion**.
- ▶ Given  $\varphi_1, \dots, \varphi_n$ , we can deduce or prove  $\psi$ . **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.
- ▶  $\varphi_1, \dots, \varphi_n \vdash \psi$  : This is called a **sequent**.  $\varphi_1, \dots, \varphi_n$  are **premises**, and  $\psi$ , the **conclusion**.
- ▶ Given  $\varphi_1, \dots, \varphi_n$ , we can deduce or prove  $\psi$ . **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.
- ▶  $\varphi_1, \dots, \varphi_n \vdash \psi$  : This is called a **sequent**.  $\varphi_1, \dots, \varphi_n$  are **premises**, and  $\psi$ , the **conclusion**.
- ▶ Given  $\varphi_1, \dots, \varphi_n$ , we can deduce or prove  $\psi$ . **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?
- ▶ 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  $\wedge i$

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

# 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

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

# Another Proof

---

- ▶ Show that  $p, p \rightarrow q, p \rightarrow (q \rightarrow \neg\neg r) \vdash r$ 
  1.  $p \rightarrow (q \rightarrow \neg\neg r)$  premise
  - 2.

# Another Proof

---

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

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

# Another Proof

---

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

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

# Another Proof

---

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

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

# Another Proof

---

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

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

# Another Proof

---

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

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

# Another Proof

---

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

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

# Lecture 02 Natural Deduction 2.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Rules for Natural Deduction

---

Another implies elimination rule or Modus Tollens MT

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

# 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  $p \rightarrow \neg q, q \vdash \neg p$

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

# A Proof

---

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

1.  $p \rightarrow \neg q$  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  $p \rightarrow q, \neg q \vdash \neg p$ .

# More Rules

---

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

# More Rules

---

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

# More Rules

---

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

# More Rules

---

- ▶ Thanks to MT, we have  $p \rightarrow q, \neg q \vdash \neg p$ .
- ▶ Can we prove  $p \rightarrow q \vdash \neg q \rightarrow \neg p$ ?
- ▶ So far, no proof rule that can do this.
- ▶ Given  $p \rightarrow q$ , 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*
- 2.

premise

## More on → *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.	<i>true</i>	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.	<i>true</i>	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.	<i>true</i>	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.	<i>true</i>	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.	<i>true</i>	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.	<i>true</i>	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.	<i>true</i>	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.	<i>true</i>	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.	<i>true</i>	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

---

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

# More Examples

---

►  $p \rightarrow (q \rightarrow r) \vdash (p \wedge q) \rightarrow r$

1.  $p \rightarrow (q \rightarrow r)$  premise
- 2.

# More Examples

---

►  $p \rightarrow (q \rightarrow r) \vdash (p \wedge q) \rightarrow r$

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

# More Examples

---

►  $p \rightarrow (q \rightarrow r) \vdash (p \wedge q) \rightarrow r$

1.  $p \rightarrow (q \rightarrow r)$  premise
2.  $p \wedge q$  assumption
3.  $p$   $\wedge e_1 2$
- 4.

# More Examples

---

►  $p \rightarrow (q \rightarrow r) \vdash (p \wedge q) \rightarrow r$

1.  $p \rightarrow (q \rightarrow r)$  premise
2.  $p \wedge q$  assumption
3.  $p$   $\wedge e_1$  2
4.  $q$   $\wedge e_2$  2
- 5.

# More Examples

---

►  $p \rightarrow (q \rightarrow r) \vdash (p \wedge q) \rightarrow r$

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

# More Examples

---

►  $p \rightarrow (q \rightarrow r) \vdash (p \wedge q) \rightarrow r$

1.  $p \rightarrow (q \rightarrow r)$  premise

2.  $p \wedge q$  assumption

3.  $p$   $\wedge e_1$  2

4.  $q$   $\wedge e_2$  2

5.  $q \rightarrow r$  MP 1,3

6.  $r$  MP 4,5

7.

# More Examples

---

- ▶  $p \rightarrow (q \rightarrow r) \vdash (p \wedge q) \rightarrow r$

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

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

# Or Elimination Example

---

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

1.  $q \rightarrow r$  premise
- 2.

# 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.

# 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

---

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

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

# 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.		

# 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.		

# 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

---

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

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.	$p \vee r$	$\vee e\ 2, 3-4, 5-7$

# 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.  $\left| \begin{array}{c} p \vee q \\ \hline \end{array} \right| \vee e (1)$
3.  $\left| \begin{array}{c} p \vee q \\ \hline \end{array} \right| \vee e (1)$

# 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

---

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

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.

# 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)  
[ $p$ ]  
 $p \vee (q \vee r)$   $\vee i_1$  3
4.  $p \vee (q \vee r)$   $\vee i_1$  3
5.  $q$   $\vee e$  (1.2)  
[ $q$ ]  
 $q \vee r$   $\vee i_1$  5
6.  $q \vee r$   $\vee i_1$  5
7.  $p \vee (q \vee r)$   $\vee i_2$  6  
[ $p \vee (q \vee r)$ ]  
 $p \vee (q \vee r)$   $\vee e$  2, 3-4, 5-7
8.  $p \vee (q \vee r)$   $\vee e$  2, 3-4, 5-7
9.  $r$   $\vee e$  (2)  
[ $r$ ]  
 $p \vee (q \vee r)$
- 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

---

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

1.     *true*                      premise
- 2.

# The Copy Rule

---

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

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

# The Copy Rule

---

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

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

# The Copy Rule

---

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

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  $\varphi$  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  $\varphi$  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  $\varphi$  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  $\varphi$ .

# Rules with $\perp$

---

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

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

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

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

# An Example

---

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

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

# An Example

---

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

1.  $\neg p \vee q$  premise

2.  $\boxed{\neg p \quad \vee e(1)}$

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

---

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

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

5.  $q$   $\perp e 4$

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  $\varphi$  (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

---

►  $p \rightarrow \neg p \vdash \neg p$

1.  $p \rightarrow \neg p$  premise

2.

# An Example

---

►  $p \rightarrow \neg p \vdash \neg p$

1.  $p \rightarrow \neg p$  premise
2.  $p$  assumption
- 3.

# An Example

---

►  $p \rightarrow \neg p \vdash \neg p$

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

# An Example

---

►  $p \rightarrow \neg p \vdash \neg p$

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

# The Last One

---

Law of the Excluded Middle (LEM)

$$\overline{\varphi \vee \neg\varphi}$$

# 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
- ▶  $\perp e, \perp i$

# Derived Rules

---

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

# 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,  $p \rightarrow q \vdash \neg p \vee q$  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.
- ▶  $\varphi_1, \dots, \varphi_n \models \psi$  iff whenever  $\varphi_1, \dots, \varphi_n$  evaluate to true, so does  $\psi$ .  $\models$  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.
- ▶  $\varphi_1, \dots, \varphi_n \models \psi$  iff whenever  $\varphi_1, \dots, \varphi_n$  evaluate to true, so does  $\psi$ .  $\models$  is read as **semantically entails**
  - ▶ Recall  $\vdash$ , and compare with  $\models$
- ▶ Formulae  $\varphi$  and  $\psi$  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.
- ▶  $\varphi_1, \dots, \varphi_n \models \psi$  iff whenever  $\varphi_1, \dots, \varphi_n$  evaluate to true, so does  $\psi$ .  $\models$  is read as **semantically entails**
  - ▶ Recall  $\vdash$ , and compare with  $\models$
- ▶ Formulae  $\varphi$  and  $\psi$  are **provably equivalent** iff  $\varphi \vdash \psi$  and  $\psi \vdash \varphi$
- ▶ Formulae  $\varphi$  and  $\psi$  are **semantically equivalent** iff  $\varphi \models \psi$  and  $\psi \models \varphi$

# Lecture 03 Soundness and Completeness Of PL.pdf

# 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.
- ▶  $\varphi_1, \dots, \varphi_n \models \psi$  iff whenever  $\varphi_1, \dots, \varphi_n$  evaluate to true, so does  $\psi$ .  $\models$  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.
- ▶  $\varphi_1, \dots, \varphi_n \models \psi$  iff whenever  $\varphi_1, \dots, \varphi_n$  evaluate to true, so does  $\psi$ .  $\models$  is read as **semantically entails**
- ▶ Two formulae  $\varphi$  and  $\psi$  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.
- ▶  $\varphi_1, \dots, \varphi_n \models \psi$  iff whenever  $\varphi_1, \dots, \varphi_n$  evaluate to true, so does  $\psi$ .  $\models$  is read as **semantically entails**
- ▶ Two formulae  $\varphi$  and  $\psi$  are **provably equivalent** iff  $\varphi \vdash \psi$  and  $\psi \vdash \varphi$
- ▶ Two formulae  $\varphi$  and  $\psi$  are **semantically equivalent** iff  $\varphi \models \psi$  and  $\psi \models \varphi$

# Soundness of Propositional Logic

---

$$\varphi_1, \dots, \varphi_n \vdash \psi \Rightarrow \varphi_1, \dots, \varphi_n \vDash \psi$$

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

# Soundness

---

- ▶ Assume  $\varphi_1, \dots, \varphi_n \vdash \psi$ .

# Soundness

---

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

# Soundness

---

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

# Soundness

---

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

# Soundness

---

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

## Soundness : Case $\wedge i$

---

- ▶ How did we arrive at  $\psi$ ? Which proof rule gave  $\psi$  as the last line?

## Soundness : Case $\wedge i$

---

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

## Soundness : Case $\wedge i$

---

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

# Soundness : Case $\wedge i$

---

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

# Soundness : Case $\wedge i$

---

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

## Soundness : Case $\rightarrow i$

---

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

## Soundness : Case $\rightarrow i$

---

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

## Soundness : Case $\rightarrow i$

---

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

## Soundness : Case $\rightarrow i$

---

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

# Soundness : Case $\rightarrow i$

---

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

# Soundness : Case $\rightarrow i$

---

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

# Soundness : Other cases

---

# Completeness

---

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

Whenever  $\varphi_1, \dots, \varphi_n$  semantically entail  $\psi$ , then  $\psi$  can be proved from  $\varphi_1, \dots, \varphi_n$ . The proof rules are **complete**

# Completeness : 3 steps

---

- ▶ Given  $\varphi_1, \dots, \varphi_n \models \psi$

# Completeness : 3 steps

---

- ▶ Given  $\varphi_1, \dots, \varphi_n \models \psi$
- ▶ Step 1: Show that  $\models \varphi_1 \rightarrow (\varphi_2 \rightarrow (\dots (\varphi_n \rightarrow \psi) \dots))$

# Completeness : 3 steps

---

- ▶ Given  $\varphi_1, \dots, \varphi_n \models \psi$
- ▶ Step 1: Show that  $\models \varphi_1 \rightarrow (\varphi_2 \rightarrow (\dots (\varphi_n \rightarrow \psi) \dots))$
- ▶ Step 2: Show that  $\vdash \varphi_1 \rightarrow (\varphi_2 \rightarrow (\dots (\varphi_n \rightarrow \psi) \dots))$

# Completeness : 3 steps

---

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

# Completeness : Step 1

---

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

# Completeness : Step 1

---

- ▶ Assume  $\varphi_1, \dots, \varphi_n \models \psi$ . Whenever all of  $\varphi_1, \dots, \varphi_n$  evaluate to true, so does  $\psi$ .
- ▶ If  $\nvdash \varphi_1 \rightarrow (\varphi_2 \rightarrow (\dots (\varphi_n \rightarrow \psi) \dots))$ , then  $\psi$  evaluates to false when all of  $\varphi_1, \dots, \varphi_n$  evaluate to true, a contradiction.

# Completeness : Step 1

---

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

# Completeness : Step 2

---

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

# Completeness : Step 2

---

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

# Completeness : Step 2

---

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

# Completeness : Step 2

---

## Truth Table to Proof

Let  $\varphi$  be a formula with variables  $p_1, \dots, p_n$ . Let  $\mathcal{T}$  be the truth table of  $\varphi$ , 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.  $\hat{p}_1, \dots, \hat{p}_n \vdash \varphi$  if  $\varphi$  evaluates to true in line  $l$
2.  $\hat{p}_1, \dots, \hat{p}_n \vdash \neg\varphi$  if  $\varphi$  evaluates to false in line  $l$

# Lecture 04 Normal Forms and Horn Clauses.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Completeness

---

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

Whenever  $\varphi_1, \dots, \varphi_n$  semantically entail  $\psi$ , then  $\psi$  can be proved from  $\varphi_1, \dots, \varphi_n$ . The proof rules are **complete**

# Completeness : 3 steps

---

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

# Completeness : Step 1

---

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

# Completeness : Step 2

---

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

# Completeness : Step 2

---

## Truth Table to Proof

Let  $\varphi$  be a formula with variables  $p_1, \dots, p_n$ . Let  $\mathcal{T}$  be the truth table of  $\varphi$ , 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.  $\hat{p}_1, \dots, \hat{p}_n \vdash \varphi$  if  $\varphi$  evaluates to true in line  $l$
2.  $\hat{p}_1, \dots, \hat{p}_n \vdash \neg\varphi$  if  $\varphi$  evaluates to false in line  $l$

# Truth Table to Proof

---

- ▶ Structural Induction on  $\varphi$ .

# Truth Table to Proof

---

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

# Truth Table to Proof

---

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

# Truth Table to Proof

---

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

# Truth Table to Proof

---

- ▶ Structural Induction on  $\varphi$ .
- ▶ Base : If  $\varphi = p$ , a proposition, then we have  $p \vdash p$  and  $\neg p \vdash \neg p$ .
- ▶ Assume for formulae of size  $\leq k - 1$  (size=height of the parse tree). **What is a parse tree?**
- ▶ Case Negation :  $\varphi = \neg\varphi_1$ 
  - ▶ Assume  $\varphi$  evaluates to true in line  $l$  of  $\mathcal{T}$ . Then  $\varphi_1$  evaluates to false in line  $l$ . By inductive hypothesis,  $\hat{p}_1, \dots, \hat{p}_n \vdash \neg\varphi_1$ .
  - ▶ Assume  $\varphi$  evaluates to false in line  $l$  of  $\mathcal{T}$ . Then  $\varphi_1$  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  $\varphi$  evaluates to false in line  $l$ , then  $\varphi_1$  evaluates to true and  $\varphi_2$  to false. Let  $\{q_1, \dots, q_k\}$  be the variables of  $\varphi_1$  and let  $\{r_1, \dots, r_j\}$  be the variables in  $\varphi_2$ .  $\{q_1, \dots, q_k\} \cup \{r_1, \dots, r_j\} = \{p_1, \dots, p_n\}$ .

# Truth Table to Proof

---

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

# Truth Table to Proof

---

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

# 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  $\varphi$  evaluates to true in line  $l$ , then there are 3 possibilities. If both  $\varphi_1, \varphi_2$  evaluate to true, then we have  $\hat{p}_1, \dots, \hat{p}_n \models \varphi_1 \wedge \varphi_2$ . 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  $\varphi$  evaluates to true in line  $l$ , then there are 3 possibilities. If both  $\varphi_1, \varphi_2$  evaluate to true, then we have  $\hat{p}_1, \dots, \hat{p}_n \models \varphi_1 \wedge \varphi_2$ . Proving  $\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$ , we are done.
  - ▶ If both  $\varphi_1, \varphi_2$  evaluate to false, then we have  $\hat{p}_1, \dots, \hat{p}_n \models \neg\varphi_1 \wedge \neg\varphi_2$ . 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  $\varphi$  evaluates to true in line  $l$ , then there are 3 possibilities. If both  $\varphi_1, \varphi_2$  evaluate to true, then we have  $\hat{p}_1, \dots, \hat{p}_n \models \varphi_1 \wedge \varphi_2$ . Proving  $\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$ , we are done.
  - ▶ If both  $\varphi_1, \varphi_2$  evaluate to false, then we have  $\hat{p}_1, \dots, \hat{p}_n \models \neg\varphi_1 \wedge \neg\varphi_2$ . Proving  $\neg\varphi_1 \wedge \neg\varphi_2 \vdash \varphi_1 \rightarrow \varphi_2$ , we are done.
  - ▶ Last, if  $\varphi_1$  evaluates to false and  $\varphi_2$  evaluates to true, then we have  $\hat{p}_1, \dots, \hat{p}_n \models \neg\varphi_1 \wedge \varphi_2$ . 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  $\models \varphi_1 \rightarrow (\varphi_2 \rightarrow (\dots (\varphi_n \rightarrow \psi) \dots))$ , use LEM on all the propositional variables of  $\varphi_1, \dots, \varphi_n, \psi$  to obtain  
 $\vdash \varphi_1 \rightarrow (\varphi_2 \rightarrow (\dots (\varphi_n \rightarrow \psi) \dots))$ .

# Completeness : Steps 2, 3

---

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

# Completeness : Steps 2, 3

---

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

# Completeness : Steps 2, 3

---

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

# 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  $F$ ,  $(x \rightarrow [\neg(y \vee z) \wedge \neg(y \rightarrow x)])$

- ▶ 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.

# 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  $F$ ,  $(x \rightarrow [\neg(y \vee z) \wedge \neg(y \rightarrow x)])$

- ▶ 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.

- ▶ Distribute  $\vee$  wherever possible.

The resultant formula  $F_1$  is in CNF and is provably equivalent to  $F$ .

$$[(\neg x \vee \neg y) \wedge (\neg x \vee \neg z)] \wedge [(\neg x \vee y) \wedge (\neg x \vee \neg x)]$$

# The Hardness of SAT

---

- ▶ Given a formula  $\varphi$  how to check if  $\varphi$  is satisfiable?
- ▶ Given a formula  $\varphi$  how to check if  $\varphi$  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 at most 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  $p \wedge q \wedge \dots \wedge r \rightarrow s$  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  $p \wedge q \wedge \cdots \wedge r \rightarrow \perp$ .
- ▶ Thus, a Horn formula is written as a conjunction of implications.

# Lecture 05 Resolution 1.pdf

# 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  $\top \rightarrow p$  is a subformula.

# The Horn Algorithm

---

Given a Horn formula  $H$ ,

- ▶ Mark all occurrences of  $p$ , whenever  $\top \rightarrow p$  is a subformula.
- ▶ If there is a subformula of the form  $(p_1 \wedge \cdots \wedge p_m) \rightarrow q$ , 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  $\top \rightarrow p$  is a subformula.
- ▶ If there is a subformula of the form  $(p_1 \wedge \cdots \wedge p_m) \rightarrow q$ , 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  $(p_1 \wedge \cdots \wedge p_m) \rightarrow \perp$ . If there is one such subformula with all  $p_i$  marked, then say **Unsat**, otherwise say **Sat**.

# An Example

---

$$(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$$

# An Example

---

$(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$

- $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$

# An Example

---

$(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$

- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$
- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$

# An Example

---

$(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$

- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$
- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$
- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$

# An Example

---

$(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$

- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$
- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$
- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$
- ▶  $(\top \rightarrow A) \wedge (C \rightarrow D) \wedge ((A \wedge B) \rightarrow C) \wedge ((C \wedge D) \rightarrow \perp) \wedge (\top \rightarrow B).$

# The Horn Algorithm

---

The Horn algorithm concludes Sat iff  $H$  is satisfiable.

# Complexity of Horn

---

- ▶ Given a Horn formula  $\psi$  with  $n$  propositions, how many times do you have to read  $\psi$ ?
- ▶ Step 1: Read once
- ▶ Step 2: Read almost  $n$  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  $\text{Res}^0(F) = \{C \mid C \text{ is a clause in } F\}$ .

# Completeness of Resolution

---

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

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

# Completeness of Resolution

---

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

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

# Example

---

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

- ▶  $\text{Res}^0(F) = F$

# Example

---

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

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

# Example

---

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

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

# Lecture 06 Resolution 2.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Resolution

---

- ▶ Given a propositional logic formula  $\varphi$ , is it unsatisfiable?

# Resolution

---

- ▶ Given a propositional logic formula  $\varphi$ , 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  $\text{Res}^0(F) = \{C \mid C \text{ is a clause in } F\}$ .

# Completeness of Resolution

---

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

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

# Completeness of Resolution

---

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

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

# Example

---

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

- ▶  $\text{Res}^0(F) = F$

# Example

---

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

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

# Example

---

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

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

# Resolution

---

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

- If  $\emptyset \in Res^*(F)$ . Then  $\emptyset \in Res^n(F)$  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  $F \vdash \perp$ . Hence,  $F$  is unsatisfiable.

# Resolution

---

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

(Discuss substitution before the proof)

# Resolution

---

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

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

# Resolution

---

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

- ▶ Let  $F$  have  $k$  clauses  $C_1, \dots, C_k$ .
- ▶ 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  $C_1, \dots, C_k$ .
- ▶ 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  $C_1, \dots, C_k$ .
- ▶ 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  $C_1, \dots, C_k$ .
- ▶ 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  $C_1, \dots, C_k$ .
- ▶ 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  $\leq n$  variables and is unsat, then  $\emptyset \in \text{Res}^*(F)$ .

# Resolution

---

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

# Resolution

---

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

- ▶ 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  $\leq n$  variables and is unsat, then  $\emptyset \in \text{Res}^*(F)$ .
- ▶ Let  $F$  have  $n + 1$  variables  $p_1, \dots, p_{n+1}$ .

- ▶ 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  $F = \text{Clauses in } G_0 \cup \text{Clauses in } G_1$

# Resolution

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

- ▶ 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  $F = \text{Clauses in } G_0 \cup \text{Clauses in } G_1$

- ▶ 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  $p_{n+1} = \text{false}$  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  $p_{n+1} = \text{false}$  in  $F$ , then  $F$  is equisatisfiable with  $F_0$
- ▶ If  $p_{n+1} = \text{true}$  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  $p_{n+1} = \text{false}$  in  $F$ , then  $F$  is equisatisfiable with  $F_0$
- ▶ If  $p_{n+1} = \text{true}$  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  $p_{n+1} = \text{false}$  in  $F$ , then  $F$  is equisatisfiable with  $F_0$
- ▶ If  $p_{n+1} = \text{true}$  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,  $\emptyset \in Res^*(F_0)$  and  $\emptyset \in Res^*(F_1)$ .

# 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,  $\emptyset \in Res^*(F_0)$  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  $\emptyset \in Res^*(F)$ .

# Resolution

---

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

# Resolution

---

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

# Resolution Summary

---

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

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

# Lecture 07 Recap.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Recap of Basics

---

- ▶ A formula  $\varphi$  is satisfiable when ...

# Recap of Basics

---

- ▶ A formula  $\varphi$  is satisfiable when ...
- ▶ A formula  $\varphi$  is valid when ...

# Recap of Basics

---

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

# Recap of Basics

---

- ▶ A formula  $\varphi$  is satisfiable when ...
- ▶ A formula  $\varphi$  is valid when ...
- ▶ A formula  $\varphi$  is satisfiable iff  $\neg\varphi$  is not valid.
- ▶ Two formulae  $\varphi_1$  and  $\varphi_2$  are equivalent iff ...

# Recap of Basics

---

- ▶ A formula  $\varphi$  is satisfiable when ...
- ▶ A formula  $\varphi$  is valid when ...
- ▶ A formula  $\varphi$  is satisfiable iff  $\neg\varphi$  is not valid.
- ▶ Two formulae  $\varphi_1$  and  $\varphi_2$  are equivalent iff ...
- ▶ Two formulae  $\varphi_1$  and  $\varphi_2$  are equisatisfiable iff ...

# Recap of Basics

---

- ▶ A formula  $\varphi$  is satisfiable when ...
- ▶ A formula  $\varphi$  is valid when ...
- ▶ A formula  $\varphi$  is satisfiable iff  $\neg\varphi$  is not valid.
- ▶ Two formulae  $\varphi_1$  and  $\varphi_2$  are equivalent iff ...
- ▶ Two formulae  $\varphi_1$  and  $\varphi_2$  are equisatisfiable iff ...
- ▶ A disjunction of literals  $L_1 \vee L_2 \vee \dots \vee L_n$  is valid iff ...

# Recap of Basics

---

- ▶ A formula  $\varphi$  is satisfiable when ...
- ▶ A formula  $\varphi$  is valid when ...
- ▶ A formula  $\varphi$  is satisfiable iff  $\neg\varphi$  is not valid.
- ▶ Two formulae  $\varphi_1$  and  $\varphi_2$  are equivalent iff ...
- ▶ Two formulae  $\varphi_1$  and  $\varphi_2$  are equisatisfiable iff ...
- ▶ A disjunction of literals  $L_1 \vee L_2 \vee \dots \vee L_n$  is valid iff ...
- ▶ A conjunction of literals  $L_1 \wedge L_2 \wedge \dots \wedge L_n$  is satisfiable iff ...

# Normal Forms : CNF Validity

---

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

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

# Lecture 08 CNF to DNF.pdf

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

# Normal Forms : DNF Satisfiability

---

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

- ▶ Checking if  $\varphi$  is valid is NP-complete. Why?
- ▶ Checking if  $\varphi$  is satisfiable is polynomial time. Why?

# Normal Forms from Truth Tables

---

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

# Normal Forms from Truth Tables

---

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

- ▶ Consider for example  $\varphi = p \leftrightarrow q$ .
- ▶ Truth table of  $\varphi$ :  $\varphi$  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  $\varphi$ . Then it is very easy to obtain the equivalent CNF/DNF of  $\varphi$ .

- ▶ Consider for example  $\varphi = p \leftrightarrow q$ .
- ▶ Truth table of  $\varphi$ :  $\varphi$  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

---

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

# CNF to DNF Sizes

---

►  $\varphi = (p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \dots (p_n \vee q_n)$

► What is the equivalent DNF formula?

►

$$\varphi' = \bigvee_{S \subseteq \{1, \dots, n\}} (\bigwedge_{i \in S} p_i \wedge \bigwedge_{i \notin S} q_i)$$

# CNF to DNF Sizes

---

- ▶  $\varphi = (p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \dots (p_n \vee q_n)$

- ▶ What is the equivalent DNF formula?

- ▶

$$\varphi' = \bigvee_{S \subseteq \{1, \dots, n\}} \left( \bigwedge_{i \in S} p_i \wedge \bigwedge_{i \notin S} q_i \right)$$

- ▶ Prove that any equivalent DNF formula has  $2^n$  clauses

# CNF to DNF Sizes

---

$$\varphi = (p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \dots (p_n \vee q_n)$$

▶ What is the equivalent DNF formula?

▶

$$\varphi' = \bigvee_{S \subseteq \{1, \dots, n\}} \left( \bigwedge_{i \in S} p_i \wedge \bigwedge_{i \notin S} q_i \right)$$

▶ 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

---

- ▶  $\varphi = (p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \dots (p_n \vee q_n)$

- ▶ What is the equivalent DNF formula?

- ▶

$$\varphi' = \bigvee_{S \subseteq \{1, \dots, n\}} \left( \bigwedge_{i \in S} p_i \wedge \bigwedge_{i \notin S} q_i \right)$$

- ▶ 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  $\varphi'$
- ▶ Show that no two *minimal* assignments satisfy the same clause of  $\varphi'$  (hence there must be  $2^n$  clauses in  $\varphi'$ )

# CNF to DNF Sizes

---

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

# CNF to DNF Sizes

---

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

# CNF to DNF Sizes

---

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

# CNF to DNF Sizes

---

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

Think of an example where DNF to CNF explodes.

# Lecture 09 DPLL.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# CNF Explosion

---

Consider the formula  $\varphi = (p_1 \wedge p_2 \dots \wedge p_n) \vee (q_1 \wedge q_2 \dots \wedge q_m)$

- ▶ What is the equivalent CNF formula?
- ▶  $\bigwedge_{i \in \{1, \dots, n\}, j \in \{1, \dots, m\}} (p_i \vee q_j)$  has  $mn$  clauses
- ▶ Distributivity explodes the formula

# Tseitin Encoding : The Idea

---

- ▶ Introducing fresh variables, Tseitin encoding can give an equisatisfiable formula without exponential explosion.
- ▶  $\varphi = p \vee (q \wedge r)$
- ▶ Replace  $q \wedge r$  with a fresh variable  $x$  and add a clause which asserts that  $x$  simulates  $q \wedge r$
- ▶  $(p \vee x) \wedge (x \leftrightarrow (q \wedge r))$
- ▶ It is enough to consider (Why?)  $(p \vee x) \wedge (x \rightarrow (q \wedge r))$  which is  $(p \vee x) \wedge (\neg x \vee q) \wedge (\neg x \vee r)$

# Tseitin Encoding

---

- ▶ Assume the input formula is in NNF (all negations attached only to literals) and has only  $\wedge, \vee$
- ▶ Replace each  $G_1 \wedge \cdots \wedge G_n$  just below a  $\vee$  with a fresh variable  $p$  and conjunct  $(\neg p \vee G_1) \wedge \cdots \wedge (\neg p \vee G_n)$  (same as  $p \rightarrow G_1 \wedge \cdots \wedge G_n$ ).

# Tseitin Encoding

---

- ▶  $\varphi = (p_1 \wedge p_2 \cdots \wedge p_n) \vee (q_1 \wedge q_2 \cdots \wedge q_m)$
- ▶ Choose fresh variables  $x, y$
- ▶  $\psi = (x \vee y) \wedge \bigwedge_{i \in \{1, \dots, n\}} (\neg x \vee p_i) \wedge \bigwedge_{j \in \{1, \dots, m\}} (\neg y \vee q_j)$  has  $m + n + 1$  clauses
- ▶  $\varphi$  and  $\psi$  are equisatisfiable. Prove.

(DPLL)Davis-Putnam-Loveland-Logemann Method

# DPLL

---

- ▶ DPLL combines search and deduction to decide CNF satisfiability
- ▶ Underlies most modern SAT solvers

# Partial Assignments

---

An assignment is a function  $\alpha : V \rightarrow \{0, 1\}$  which maps each variable to true(1) or false (0). A partial assignment  $\alpha$  is one under which some variables are unassigned.

# Partial Assignments

---

An assignment is a function  $\alpha : V \rightarrow \{0, 1\}$  which maps each variable to true(1) or false (0). A partial assignment  $\alpha$  is one under which some variables are unassigned.

- ▶ Under a partial assignment  $\alpha$ , the **state** of a variable  $v$  is true if  $\alpha(v) = 1$ , false if  $\alpha(v) = 0$ , and unassigned otherwise.
- ▶ Let  $V = \{x, y, z\}$  and let  $\alpha(x) = 1, \alpha(y) = 0$ . Then the state of  $x$  under  $\alpha$  is true, state of  $y$  is false, and the state of  $z$  is unassigned.

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $C = x \vee y \vee z$  is true

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $C = x \vee y \vee z$  is true
- ▶ The state of  $C = x \vee \neg y \vee z$  is unassigned

# State of a Clause

---

Assume we have a formula in CNF. Under a partial assignment  $\alpha$ ,

- ▶ a clause  $C$  is true if there exists some literal  $\ell$  in  $C$  whose state is true
- ▶ a clause  $C$  is false if the state of all literals in  $C$  is false
- ▶ Otherwise, the state of  $C$  is unassigned

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $C = x \vee y \vee z$  is true
- ▶ The state of  $C = x \vee \neg y \vee z$  is unassigned
- ▶ The state of  $C = x \vee \neg y$  is false

# State of a Formula

---

Under a partial assignment  $\alpha$ ,

- ▶ A CNF formula  $F$  is true if for each  $C \in F$ ,  $C$  is true
- ▶ A CNF formula  $F$  is false if there  $C \in F$  such that  $C$  is false
- ▶ Otherwise,  $F$  is unassigned.

# State of a Formula

---

Under a partial assignment  $\alpha$ ,

- ▶ A CNF formula  $F$  is true if for each  $C \in F$ ,  $C$  is true
- ▶ A CNF formula  $F$  is false if there  $C \in F$  such that  $C$  is false
- ▶ Otherwise,  $F$  is unassigned.

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

# State of a Formula

---

Under a partial assignment  $\alpha$ ,

- ▶ A CNF formula  $F$  is true if for each  $C \in F$ ,  $C$  is true
- ▶ A CNF formula  $F$  is false if there  $C \in F$  such that  $C$  is false
- ▶ Otherwise,  $F$  is unassigned.

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $F = (x \vee y \vee z) \wedge (x \vee \neg y \vee z)$  is unassigned

# State of a Formula

---

Under a partial assignment  $\alpha$ ,

- ▶ A CNF formula  $F$  is true if for each  $C \in F$ ,  $C$  is true
- ▶ A CNF formula  $F$  is false if there  $C \in F$  such that  $C$  is false
- ▶ Otherwise,  $F$  is unassigned.

Consider the partial assignment  $\alpha(x) = 0, \alpha(y) = 1$ .

- ▶ The state of  $F = (x \vee y \vee z) \wedge (x \vee \neg y \vee z)$  is unassigned
- ▶ The state of  $F = (x \vee y \vee z) \wedge (x \vee \neg y)$  is false

# Unit Clause and Unit Literal

---

Let  $C$  be a clause and  $\alpha$  a partial assignment. Then

- ▶  $C$  is a unit clause under  $\alpha$  if there is a literal  $\ell \in C$  which is unassigned, and the rest are false.
- ▶ Then  $\ell$  is a unit literal under  $\alpha$ .

# Unit Clause and Unit Literal

---

Let  $C$  be a clause and  $\alpha$  a partial assignment. Then

- ▶  $C$  is a unit clause under  $\alpha$  if there is a literal  $\ell \in C$  which is unassigned, and the rest are false.
- ▶ Then  $\ell$  is a unit literal under  $\alpha$ .

Let  $\alpha(x) = 0, \alpha(y) = 1$  be a partial assignment.

- ▶  $C = x \vee \neg y \vee \neg z$  is a unit clause and  $\neg z$  is a unit literal

# Unit Clause and Unit Literal

---

Let  $C$  be a clause and  $\alpha$  a partial assignment. Then

- ▶  $C$  is a unit clause under  $\alpha$  if there is a literal  $\ell \in C$  which is unassigned, and the rest are false.
- ▶ Then  $\ell$  is a unit literal under  $\alpha$ .

Let  $\alpha(x) = 0, \alpha(y) = 1$  be a partial assignment.

- ▶  $C = x \vee \neg y \vee \neg z$  is a unit clause and  $\neg z$  is a unit literal
- ▶  $C = x \vee \neg y \vee \neg z \vee w$  is not a unit clause

# Unit Clause and Unit Literal

---

Let  $C$  be a clause and  $\alpha$  a partial assignment. Then

- ▶  $C$  is a unit clause under  $\alpha$  if there is a literal  $\ell \in C$  which is unassigned, and the rest are false.
- ▶ Then  $\ell$  is a unit literal under  $\alpha$ .

Let  $\alpha(x) = 0, \alpha(y) = 1$  be a partial assignment.

- ▶  $C = x \vee \neg y \vee \neg z$  is a unit clause and  $\neg z$  is a unit literal
- ▶  $C = x \vee \neg y \vee \neg z \vee w$  is not a unit clause
- ▶  $C = x \vee \neg y$  is not a unit clause

# DPLL

---

DPLL maintains a partial assignment, to begin with the empty assignment.

- ▶ Assigns unassigned variables 0 or 1 randomly
- ▶ Sometimes, forced to assign 0 or 1 to unit literals

# DPLL Actions

---

- ▶ DPLL has 3 actions : decisions, unit propagation and backtracking
- ▶ Decisions : Decide an assignment for a variable (random choice)
- ▶ Implied assignments or unit propagation : to deal with unit literals
- ▶ Backtrack when in a conflict

# DPLL Algorithm

---

- ▶ At any time, the state of the algorithm is a pair  $(F, \alpha)$  where  $F$  is the CNF and  $\alpha$  is a partial assignment
- ▶ A state  $(F, \alpha)$  is **successful** if  $\alpha$  sets some literal in each clause of  $F$  to be true
- ▶ A **conflict** state is one where  $\alpha$  sets all literals in some clause of  $F$  to be false

# DPLL Algorithm

---

- ▶ Let  $F|\alpha$  denote the set of clauses obtained by deleting from  $F$ , any clause containing a true literal from  $\alpha$ , and deleting from each remaining clause, all literals false under  $\alpha$ . Let  $\alpha(x) = 0, \alpha(y) = 1$ .
- ▶ For  $F = (x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z)$ ,  $F|\alpha = \{\neg z\}$
- ▶ For  $F = (x \vee y) \wedge (\neg x \vee \neg y)$ ,  $F|\alpha = \{\}$ .
- ▶ For  $F = (x \vee \neg y)$ ,  $\perp \in F|\alpha$
- ▶ If  $(F, \alpha)$  is successful, then  $F|\alpha = \{\}$
- ▶ If  $(F, \alpha)$  is in conflict, then the empty clause  $\perp$  is in  $F|\alpha$ .

# The DPLL Algorithm

---

Input : CNF formula  $F$ .

1. Initialise  $\alpha$  as the empty assignment
2. While there is a unit clause  $L$  in  $F|\alpha$ , add  $L = 1$  to  $\alpha$  (unit propagation)
3. If  $F|\alpha$  contains no clauses, then stop and output  $\alpha$
4. If  $F|\alpha$  contains the empty clause, then apply the learning procedure to add a new clause  $C$  to  $F$ . If it is the empty clause, output UNSAT. Otherwise, backtrack to the highest level at which  $C$  is a unit clause, go to line 2.
5. Decide on a new assignment  $p = b$  to be added to  $\alpha$ , goto line 2.

# DPLL Example

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

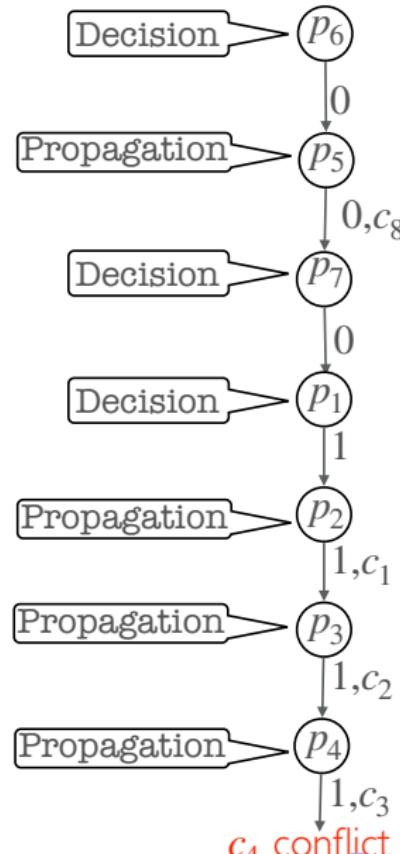
$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$



## Clauses Learning

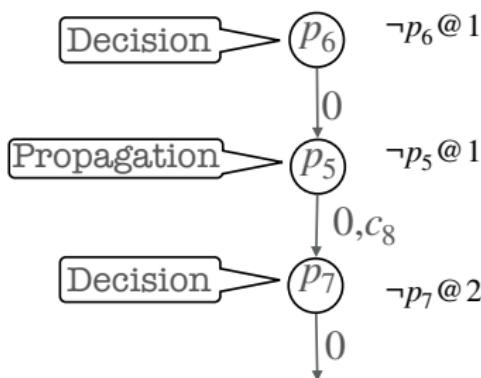
# Run of DPLL

---

The partial assignment in construction is called a **a run of DPLL**. In the previous slide, the run ended in a conflict.

# Decision Level

During a run, the decision level of a true literal is the number of decisions after which the literal was made true.



# Implication Graphs

---

During a DPLL run, we maintain a data structure called an implication graph.

Under a partial assignment  $\alpha$ , the implication graph  $G = (V, E)$ ,

- ▶  $V$  is the set of true literals under  $\alpha$ , and the conflict node
- ▶  $E = \{(\ell_1, \ell_2) \mid \neg\ell_1 \text{ belongs to the clause due to which unit propagation made } \ell_2 \text{ true}\}$

Each node is annotated with the decision level.

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

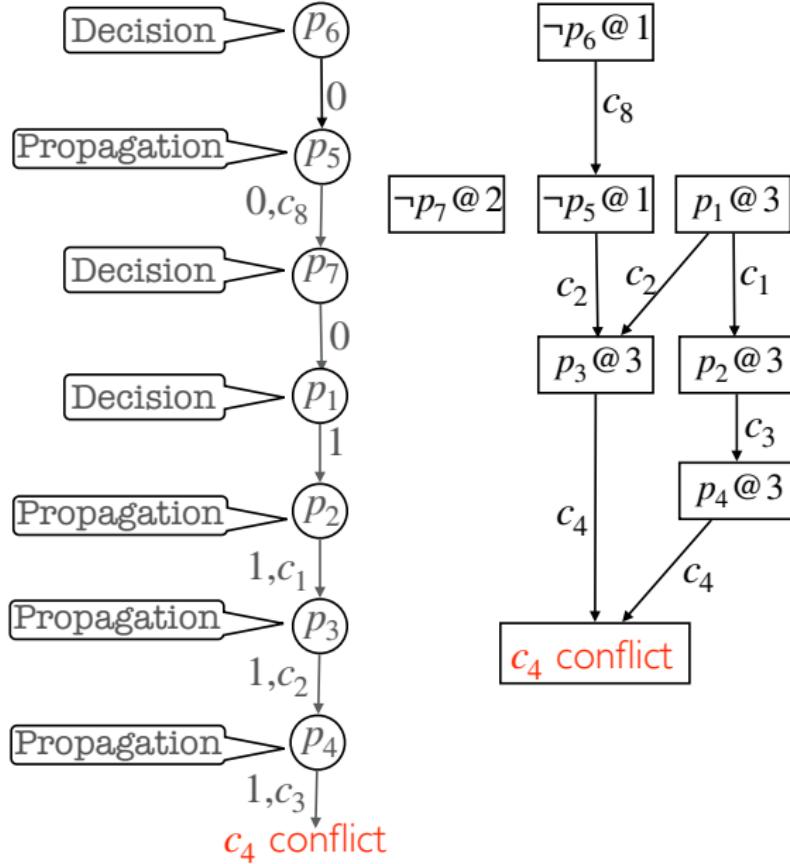
$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

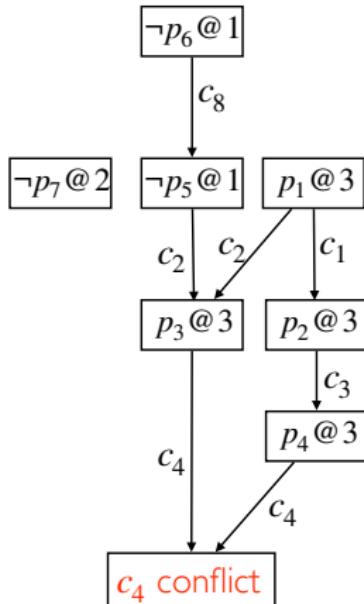
$$c_8 = p_6 \vee \neg p_5$$



# Conflict Clause

Traverse the implication graph backwards to find the set of decisions that created a conflict. The negations of the causing decisions is the **conflict clause**.

$$\begin{aligned}c_1 &= \neg p_1 \vee p_2 \\c_2 &= \neg p_1 \vee p_3 \vee p_5 \\c_3 &= \neg p_2 \vee p_4 \\c_4 &= \neg p_3 \vee \neg p_4 \\c_5 &= p_1 \vee p_5 \vee \neg p_2 \\c_6 &= p_2 \vee p_3 \\c_7 &= p_2 \vee \neg p_3 \vee p_7 \\c_8 &= p_6 \vee \neg p_5\end{aligned}$$



Conflict clause :  $p_6 \vee \neg p_1$  is added : resolve  $c_4$  with  $c_3, c_1, c_2, c_8$

# Clause Learning

---

- ▶ We add the conflict clause to the input set of clauses
- ▶ backtrack to the second last conflicting decision, and proceed like DPLL

## Adding the conflict clause

- ▶ does not affect satisfiability of the original formula (think of resolution)
- ▶ ensures that the conflicting partial assignment will not be tried again

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

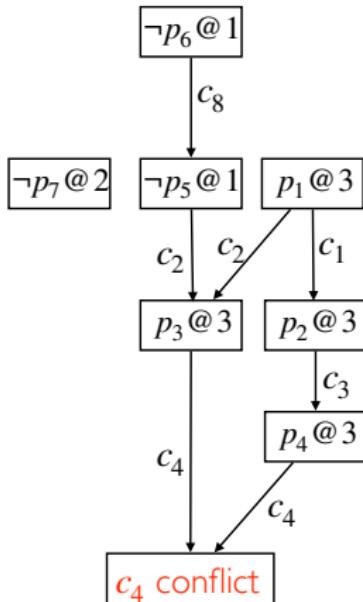
$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$C_8 \equiv p_{\varepsilon} \vee \neg p_{\varepsilon}$$



The second last decision is  $p_6 = 0$ . Unit propagation will force  $p_1 = 0$ .

The combination  $p_6 = 0, p_1 = 1$  will not be tried again.

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

$$c_4 = \neg p_3 \vee \neg p_4$$

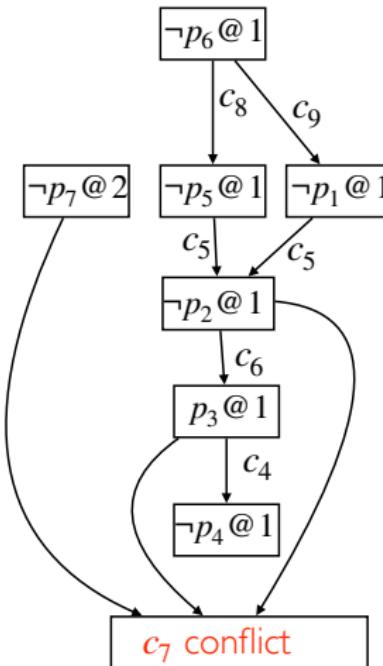
$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$

$$c_9 = p_6 \vee \neg p_1$$



Conflict clause :  $p_7 \vee p_6$  is added and backtrack

Set  $p_7 = 1$  by unit propagation.

$$c_1 = \neg p_1 \vee p_2$$

$$c_2 = \neg p_1 \vee p_3 \vee p_5$$

$$c_3 = \neg p_2 \vee p_4$$

$$c_4 = \neg p_3 \vee \neg p_4$$

$$c_5 = p_1 \vee p_5 \vee \neg p_2$$

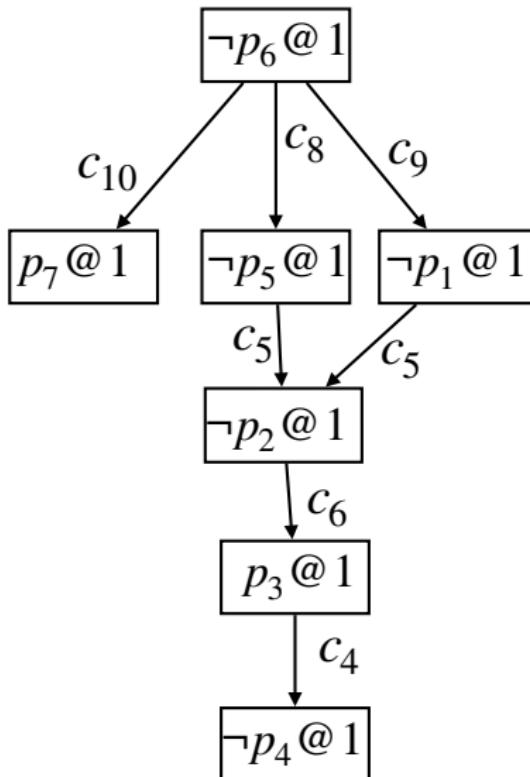
$$c_6 = p_2 \vee p_3$$

$$c_7 = p_2 \vee \neg p_3 \vee p_7$$

$$c_8 = p_6 \vee \neg p_5$$

$$c_9 = p_6 \vee \neg p_1$$

$$c_{10} = p_6 \vee p_7$$



# Lecture 10 FOL Syntax.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# The DPLL Algorithm

---

Input : CNF formula  $F$ .

1. Initialise  $\alpha$  as the empty assignment
2. While there is a unit clause  $L$  in  $F|\alpha$ , add  $L = 1$  to  $\alpha$  (unit propagation)
3. If  $F|\alpha$  contains no clauses, then stop and output  $\alpha$
4. If  $F|\alpha$  contains the empty clause, then apply the learning procedure to add a new clause  $C$  to  $F$ . If it is the empty clause, output UNSAT. Otherwise, backtrack to the highest level at which  $C$  is a unit clause, go to line 2.
5. Decide on a new assignment  $p = b$  to be added to  $\alpha$ , goto line 2.

# DPLL Example

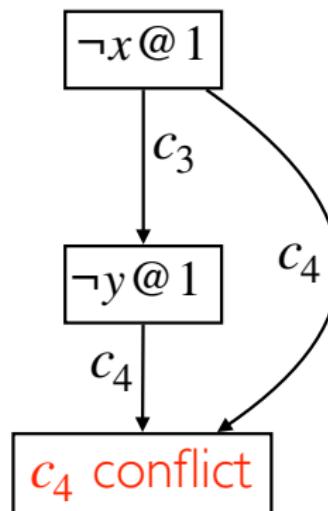
---

$$c_1 = \neg x \vee \neg y$$

$$c_2 = \neg x \vee y$$

$$c_3 = x \vee \neg y$$

$$c_4 = x \vee y$$



Clause learnt :  $x$  (Resolve  $c_4$  with  $c_3$ )

# DPLL Example

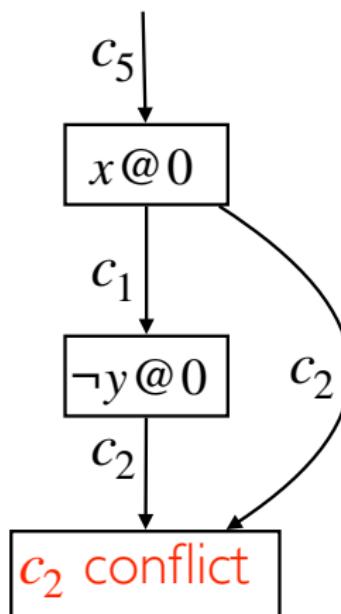
$$c_1 = \neg x \vee \neg y$$

$$c_2 = \neg x \vee y$$

$$c_3 = x \vee \neg y$$

$$c_4 = x \vee y$$

$$c_5 = x$$



Clause learnt : Resolve  $c_2$  with  $c_1, c_5$ . Empty clause.

# DPLL Correctness

---

## Termination

A sequence of decisions which lead to a conflict cannot be repeated : the variables in the learned clause are all decision variables. In a future assignment, if all but one of these are set to false, the remaining one will not be a decision variable.

# DPLL Correctness

## Termination

A sequence of decisions which lead to a conflict cannot be repeated : the variables in the learned clause are all decision variables. In a future assignment, if all but one of these are set to false, the remaining one will not be a decision variable.

## Correctness

Correctness is straightforward :  $F \vdash$  the learned clause. Thus, if the empty clause is learnt, then  $F$  is unsat. Otherwise, if DPLL terminates with a satisfying assignment  $\alpha$ , then the input formula is also satisfied by  $\alpha$ .

# Modern SAT Solvers

---

Numerous enhancements/heuristics

- ▶ Decision heuristics to choose decision variables
- ▶ Random restarts

## First Order Logic

Extends propositional logic

- ▶ Propositional logic : atomic formulas have no internal structure
- ▶ FOL : atomic formulas are predicates that assert a relationship between certain elements
- ▶ Quantification in FOL : ability to assert that a certain property holds for all elements or only for some element.
- ▶ Formulae in FOL are over some signature.

# Signatures

---

- ▶ A **vocabulary** or **signature**  $\tau$  is a set consisting of
  - ▶ constants  $c_1, c_2, \dots$
  - ▶ Relation symbols  $R_1, R_2, \dots$ , each with some arity  $k$ , denoted  $R_i^k$
  - ▶ Function symbols  $f_1, \dots$  each with some arity  $k$ , denoted  $f_i^k$
- ▶ We look at finite signatures
- ▶  $\tau = (E^2, F^3, f^1)$  is a finite signature with two relations,  $E$  with arity 2 and  $F$  with arity 3, and a function  $f$  with arity 1

# Symbols in First Order Logic

---

Formulae of FO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

# Symbols in First Order Logic

---

Formulae of FO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

- ▶ The symbols  $\perp, \top$  called **false, true**

# Symbols in First Order Logic

---

Formulae of FO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

- ▶ The symbols  $\perp, \top$  called **false, true**
- ▶ An element of the infinite set  $\mathcal{V} = \{x_1, x_2, \dots\}$  called **variables**

# Symbols in First Order Logic

---

Formulae of FO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

- ▶ The symbols  $\perp, \top$  called **false, true**
- ▶ An element of the infinite set  $\mathcal{V} = \{x_1, x_2, \dots\}$  called **variables**
- ▶ Constants, relations and functions from  $\tau$

# Symbols in First Order Logic

---

Formulae of FO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

- ▶ The symbols  $\perp, \top$  called **false, true**
- ▶ An element of the infinite set  $\mathcal{V} = \{x_1, x_2, \dots\}$  called **variables**
- ▶ Constants, relations and functions from  $\tau$
- ▶ The symbols  $\rightarrow, \neg, \wedge, \vee$

# Symbols in First Order Logic

---

Formulae of FO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

- ▶ The symbols  $\perp, \top$  called **false, true**
- ▶ An element of the infinite set  $\mathcal{V} = \{x_1, x_2, \dots\}$  called **variables**
- ▶ Constants, relations and functions from  $\tau$
- ▶ The symbols  $\rightarrow, \neg, \wedge, \vee$
- ▶ The symbol  $\forall$  called the **universal quantifier**

# Symbols in First Order Logic

---

Formulae of FO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

- ▶ The symbols  $\perp, \top$  called **false, true**
- ▶ An element of the infinite set  $\mathcal{V} = \{x_1, x_2, \dots\}$  called **variables**
- ▶ Constants, relations and functions from  $\tau$
- ▶ The symbols  $\rightarrow, \neg, \wedge, \vee$
- ▶ The symbol  $\forall$  called the **universal quantifier**
- ▶ The symbol  $\exists$  called the **existential quantifier**

# Symbols in First Order Logic

---

Formulae of FO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

- ▶ The symbols  $\perp, \top$  called **false, true**
- ▶ An element of the infinite set  $\mathcal{V} = \{x_1, x_2, \dots\}$  called **variables**
- ▶ Constants, relations and functions from  $\tau$
- ▶ The symbols  $\rightarrow, \neg, \wedge, \vee$
- ▶ The symbol  $\forall$  called the **universal quantifier**
- ▶ The symbol  $\exists$  called the **existential quantifier**
- ▶ The symbols ( and ) called **paranthesis**

# Terms

---

Given a signature  $\tau$ , the set of  $\tau$ -terms are defined inductively as follows.

- ▶ Each variable is a term
- ▶ Each constant symbol is a term
- ▶ If  $t_1, \dots, t_k$  are terms and  $f$  is a  $k$ -ary function, then  $f(t_1, \dots, t_k)$  is a term
- ▶ Ground Terms : Terms without variables. For instance  $f(c_1, \dots, c_k)$  for constants  $c_1, \dots, c_k$ .

# Well formed Formulae

---

A well-formed formula (wff) over a signature  $\tau$  is inductively defined as follows:

# Well formed Formulae

---

A well-formed formula (wff) over a signature  $\tau$  is inductively defined as follows:

- ▶  $\perp, \top$  are wff

# Well formed Formulae

---

A well-formed formula (wff) over a signature  $\tau$  is inductively defined as follows:

- ▶  $\perp, \top$  are wff
- ▶ If  $t_1, t_2$  are either variables or constants in  $\tau$ , then  $t_1 = t_2$  is a wff

# Well formed Formulae

---

A well-formed formula (wff) over a signature  $\tau$  is inductively defined as follows:

- ▶  $\perp, \top$  are wff
- ▶ If  $t_1, t_2$  are either variables or constants in  $\tau$ , then  $t_1 = t_2$  is a wff
- ▶ If  $t_i$  is a term, for  $1 \leq i \leq k$  and  $R$  is a  $k$ -ary relation symbol in  $\tau$ , then  $R(t_1, \dots, t_k)$  is a wff

# Well formed Formulae

---

A well-formed formula (wff) over a signature  $\tau$  is inductively defined as follows:

- ▶  $\perp, \top$  are wff
- ▶ If  $t_1, t_2$  are either variables or constants in  $\tau$ , then  $t_1 = t_2$  is a wff
- ▶ If  $t_i$  is a term, for  $1 \leq i \leq k$  and  $R$  is a  $k$ -ary relation symbol in  $\tau$ , then  $R(t_1, \dots, t_k)$  is a wff
- ▶ If  $\varphi$  and  $\psi$  are wff, then  $\varphi \rightarrow \psi, \varphi \wedge \psi, \varphi \vee \psi, \neg \psi$  are all wff

# Well formed Formulae

---

A well-formed formula (wff) over a signature  $\tau$  is inductively defined as follows:

- ▶  $\perp, \top$  are wff
- ▶ If  $t_1, t_2$  are either variables or constants in  $\tau$ , then  $t_1 = t_2$  is a wff
- ▶ If  $t_i$  is a term, for  $1 \leq i \leq k$  and  $R$  is a  $k$ -ary relation symbol in  $\tau$ , then  $R(t_1, \dots, t_k)$  is a wff
- ▶ If  $\varphi$  and  $\psi$  are wff, then  $\varphi \rightarrow \psi, \varphi \wedge \psi, \varphi \vee \psi, \neg \psi$  are all wff
- ▶ If  $\varphi$  is a wff and  $x$  is a variable, then  $(\forall x)\varphi$  and  $(\exists x)\varphi$  are wff

# Well formed Formulae

---

A well-formed formula (wff) over a signature  $\tau$  is inductively defined as follows:

- ▶  $\perp, \top$  are wff
- ▶ If  $t_1, t_2$  are either variables or constants in  $\tau$ , then  $t_1 = t_2$  is a wff
- ▶ If  $t_i$  is a term, for  $1 \leq i \leq k$  and  $R$  is a  $k$ -ary relation symbol in  $\tau$ , then  $R(t_1, \dots, t_k)$  is a wff
- ▶ If  $\varphi$  and  $\psi$  are wff, then  $\varphi \rightarrow \psi, \varphi \wedge \psi, \varphi \vee \psi, \neg \psi$  are all wff
- ▶ If  $\varphi$  is a wff and  $x$  is a variable, then  $(\forall x)\varphi$  and  $(\exists x)\varphi$  are wff
- ▶ The second and third are **atomic** formulae.
- ▶ If a formula  $F$  occurs as part of another formula  $G$ , then  $F$  is called a **sub formula** of  $G$ .

# Logical Abbreviations : Boolean Connectives

---

- ▶  $\neg\varphi = \varphi \rightarrow \perp$
- ▶  $\top = \neg\perp$
- ▶  $\varphi \vee \psi = \neg\varphi \rightarrow \psi$
- ▶  $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$
- ▶  $\exists x.\varphi = \neg(\forall x.\neg\varphi)$
- ▶ Precedence of operators : Quantifiers and negation highest, followed by  $\vee, \wedge$ , followed by  $\rightarrow$ .
  - ▶  $\forall x P(x) \wedge R(x)$  is  $[\forall x.[P(x)]] \wedge R(x)$

# An Example

---

Consider the signature  $\tau = \{R\}$  where  $R$  is a binary relation. The following are FO formulae over this signature.

- ▶  $\forall x R(x, x)$  Reflexivity
- ▶  $\forall x (R(x, x) \rightarrow \perp)$  Irreflexivity
- ▶  $\forall x \forall y (R(x, y) \rightarrow R(y, x))$  Symmetry
- ▶  $\forall x \forall y \forall z (R(x, y) \rightarrow (R(y, z) \rightarrow R(x, z)))$  Transitivity

# Lecture 11 FOL Semantics.pdf

# CS 228 : Logic in Computer Science

Krishna. S

## First-Order Logic : Semantics

# Structures

---

- ▶ A structure  $\mathcal{A}$  of signature  $\tau$  consists of

# Structures

---

- ▶ A structure  $\mathcal{A}$  of signature  $\tau$  consists of
  - ▶ A non-empty set  $A$  or  $u(\mathcal{A})$  called the **universe**

# Structures

---

- ▶ A structure  $\mathcal{A}$  of signature  $\tau$  consists of
  - ▶ A non-empty set  $A$  or  $u(\mathcal{A})$  called the **universe**
  - ▶ For each constant  $c$  in the signature  $\tau$ , a fixed element  $c_{\mathcal{A}}$  is assigned from the universe  $A$

# Structures

---

- ▶ A structure  $\mathcal{A}$  of signature  $\tau$  consists of
  - ▶ A non-empty set  $A$  or  $u(\mathcal{A})$  called the **universe**
  - ▶ For each constant  $c$  in the signature  $\tau$ , a fixed element  $c_{\mathcal{A}}$  is assigned from the universe  $A$
  - ▶ For each  $k$ -ary relation  $R^k$  in the signature  $\tau$ , a set of  $k$ -tuples from  $A^k$  is assigned to  $R^{\mathcal{A}}$

# Structures

---

- ▶ A structure  $\mathcal{A}$  of signature  $\tau$  consists of
  - ▶ A non-empty set  $A$  or  $u(\mathcal{A})$  called the **universe**
  - ▶ For each constant  $c$  in the signature  $\tau$ , a fixed element  $c_{\mathcal{A}}$  is assigned from the universe  $A$
  - ▶ For each  $k$ -ary relation  $R^k$  in the signature  $\tau$ , a set of  $k$ -tuples from  $A^k$  is assigned to  $R^{\mathcal{A}}$
  - ▶ The structure  $\mathcal{A}$  is finite if  $A$  (or  $u(\mathcal{A})$ ) is finite

# Examples of Structures : A Graph

---

- ▶  $\tau = \{E\}$ , with  $E$  binary.

# Examples of Structures : A Graph

---

- ▶  $\tau = \{E\}$ , with  $E$  binary.
  - ▶ A graph structure over  $\tau$  is  $\mathcal{G} = (V, E^{\mathcal{G}})$ ,
  - ▶ The **universe**  $u(\mathcal{G})$  is the set of vertices  $V$
  - ▶ The relation  $E$  is the edge relation

# Examples of Structures : A Graph

---

- ▶  $\tau = \{E\}$ , with  $E$  binary.
  - ▶ A graph structure over  $\tau$  is  $\mathcal{G} = (V, E^{\mathcal{G}})$ ,
  - ▶ The **universe**  $u(\mathcal{G})$  is the set of vertices  $V$
  - ▶ The relation  $E$  is the edge relation
  - ▶  $\mathcal{G} = (V = \{1, 2, 3, 4\}, E^{\mathcal{G}} = \{(1, 2), (2, 3), (3, 4), (1, 1)\})$ . We could just as well draw the graph for convenience.

# Examples of Structures : An Order

---

- ▶  $\tau = \{<, S\}$  with  $<$ ,  $S$  binary.

# Examples of Structures : An Order

---

- ▶  $\tau = \{<, S\}$  with  $<$ ,  $S$  binary.
  - ▶ A finite order structure over  $\tau$  is  $\mathcal{O} = (O, <^{\mathcal{O}}, S^{\mathcal{O}})$
  - ▶ The universe  $u(\mathcal{O})$  is the finite ordered set  $O$
  - ▶  $<^{\mathcal{O}}$  is the ordering on  $O$  and  $S^{\mathcal{O}}$  is the successor on  $O$

# Examples of Structures : An Order

---

- ▶  $\tau = \{<, S\}$  with  $<$ ,  $S$  binary.
  - ▶ A finite order structure over  $\tau$  is  $\mathcal{O} = (O, <^{\mathcal{O}}, S^{\mathcal{O}})$
  - ▶ The universe  $u(\mathcal{O})$  is the finite ordered set  $O$
  - ▶  $<^{\mathcal{O}}$  is the ordering on  $O$  and  $S^{\mathcal{O}}$  is the successor on  $O$
  - ▶  $O = (O = \{1, 2, 4\}, <^{\mathcal{O}} = \{(1, 2), (1, 4), (2, 4)\}, S^{\mathcal{O}} = \{(1, 2)\})$

# Examples of Structures : A Word

---

- ▶  $\tau = \{<, S, Q_a, Q_b\}$ , where  $<$ ,  $S$  are binary,  $Q_a, Q_b$  are unary relations.

# Examples of Structures : A Word

---

- ▶  $\tau = \{<, S, Q_a, Q_b\}$ , where  $<$ ,  $S$  are binary,  $Q_a, Q_b$  are unary relations.
  - ▶ A word structure  $\mathcal{W} = (u(\mathcal{W}), <^{\mathcal{W}}, S^{\mathcal{W}}, Q_a{}^{\mathcal{W}}, Q_b{}^{\mathcal{W}})$

# Examples of Structures : A Word

---

- ▶  $\tau = \{<, S, Q_a, Q_b\}$ , where  $<$ ,  $S$  are binary,  $Q_a, Q_b$  are unary relations.
  - ▶ A word structure  $\mathcal{W} = (u(\mathcal{W}), <^{\mathcal{W}}, S^{\mathcal{W}}, Q_a{}^{\mathcal{W}}, Q_b{}^{\mathcal{W}})$
  - ▶ The universe  $u(\mathcal{W})$  consists of the positions in a word  $W$  over symbols  $a, b$

# Examples of Structures : A Word

---

- ▶  $\tau = \{<, S, Q_a, Q_b\}$ , where  $<$ ,  $S$  are binary,  $Q_a, Q_b$  are unary relations.
  - ▶ A word structure  $\mathcal{W} = (u(\mathcal{W}), <^{\mathcal{W}}, S^{\mathcal{W}}, Q_a{}^{\mathcal{W}}, Q_b{}^{\mathcal{W}})$
  - ▶ The universe  $u(\mathcal{W})$  consists of the positions in a word  $W$  over symbols  $a, b$
  - ▶  $<^{\mathcal{W}}$  is the ordering relation on the positions of  $W$

# Examples of Structures : A Word

---

- ▶  $\tau = \{<, S, Q_a, Q_b\}$ , where  $<$ ,  $S$  are binary,  $Q_a, Q_b$  are unary relations.
  - ▶ A word structure  $\mathcal{W} = (u(\mathcal{W}), <^{\mathcal{W}}, S^{\mathcal{W}}, Q_a{}^{\mathcal{W}}, Q_b{}^{\mathcal{W}})$
  - ▶ The universe  $u(\mathcal{W})$  consists of the positions in a word  $W$  over symbols  $a, b$
  - ▶  $<^{\mathcal{W}}$  is the ordering relation on the positions of  $W$
  - ▶  $S^{\mathcal{W}}$  is the successor relation on the positions of  $W$

# Examples of Structures : A Word

---

- ▶  $\tau = \{<, S, Q_a, Q_b\}$ , where  $<$ ,  $S$  are binary,  $Q_a, Q_b$  are unary relations.
  - ▶ A word structure  $\mathcal{W} = (u(\mathcal{W}), <^{\mathcal{W}}, S^{\mathcal{W}}, Q_a{}^{\mathcal{W}}, Q_b{}^{\mathcal{W}})$
  - ▶ The universe  $u(\mathcal{W})$  consists of the positions in a word  $W$  over symbols  $a, b$
  - ▶  $<^{\mathcal{W}}$  is the ordering relation on the positions of  $W$
  - ▶  $S^{\mathcal{W}}$  is the successor relation on the positions of  $W$
  - ▶  $Q_a{}^{\mathcal{W}}$  is the set of positions labeled  $a$  in  $W$

# Examples of Structures : A Word

---

- ▶  $\tau = \{<, S, Q_a, Q_b\}$ , where  $<$ ,  $S$  are binary,  $Q_a, Q_b$  are unary relations.
  - ▶ A word structure  $\mathcal{W} = (u(\mathcal{W}), <^{\mathcal{W}}, S^{\mathcal{W}}, Q_a{}^{\mathcal{W}}, Q_b{}^{\mathcal{W}})$
  - ▶ The universe  $u(\mathcal{W})$  consists of the positions in a word  $W$  over symbols  $a, b$
  - ▶  $<^{\mathcal{W}}$  is the ordering relation on the positions of  $W$
  - ▶  $S^{\mathcal{W}}$  is the successor relation on the positions of  $W$
  - ▶  $Q_a{}^{\mathcal{W}}$  is the set of positions labeled  $a$  in  $W$
  - ▶  $Q_b{}^{\mathcal{W}}$  is the set of positions labeled  $b$  in  $W$

# Examples of Structures : A Word

---

- ▶  $\tau = \{<, S, Q_a, Q_b\}$ , where  $<$ ,  $S$  are binary,  $Q_a, Q_b$  are unary relations.
  - ▶ A word structure  $\mathcal{W} = (u(\mathcal{W}), <^{\mathcal{W}}, S^{\mathcal{W}}, Q_a{}^{\mathcal{W}}, Q_b{}^{\mathcal{W}})$
  - ▶ The universe  $u(\mathcal{W})$  consists of the positions in a word  $W$  over symbols  $a, b$
  - ▶  $<^{\mathcal{W}}$  is the ordering relation on the positions of  $W$
  - ▶  $S^{\mathcal{W}}$  is the successor relation on the positions of  $W$
  - ▶  $Q_a{}^{\mathcal{W}}$  is the set of positions labeled  $a$  in  $W$
  - ▶  $Q_b{}^{\mathcal{W}}$  is the set of positions labeled  $b$  in  $W$
  - ▶ The structure with  $u(\mathcal{W}) = \{0, 1, 2, \dots, 8\}$ ,  
 $Q_a{}^{\mathcal{W}} = \{0, 1, 4, 6, 8\}$ ,  $Q_b{}^{\mathcal{W}} = \{2, 3, 5, 7\}$ ,
  - ▶  $<^{\mathcal{W}} = \{(0, 1), (0, 2), \dots, (7, 8)\}$ ,  $S^{\mathcal{W}} = \{(0, 1), (1, 2), \dots, (7, 8)\}$  uniquely defines the word  $W = aabbababa$ .
  - ▶ For convenience, we can just write the word instead of the structure.

# Free and Bound Variables

---

- ▶ For a wff  $\varphi = \forall x\psi$  or  $\exists x\psi$ ,  $\psi$  is said to be the **scope** of the quantifier  $x$

# Free and Bound Variables

---

- ▶ For a wff  $\varphi = \forall x\psi$  or  $\exists x\psi$ ,  $\psi$  is said to be the **scope** of the quantifier  $x$
- ▶ Every occurrence of  $x$  in  $\forall x\psi$  or  $\exists x\psi$  is **bound**
- ▶ Any occurrence of  $x$  which is not bound is called **free**

# Free and Bound Variables

---

- ▶ For a wff  $\varphi = \forall x\psi$  or  $\exists x\psi$ ,  $\psi$  is said to be the **scope** of the quantifier  $x$
- ▶ Every occurrence of  $x$  in  $\forall x\psi$  or  $\exists x\psi$  is **bound**
- ▶ Any occurrence of  $x$  which is not bound is called **free**
- ▶  $\varphi = P(x, y) \rightarrow \forall x((\forall y R(x, y)) \rightarrow Q(x, y))$ 
  - ▶  $y$  is free in  $Q(x, y)$  and bound in  $R(x, y)$ ,
  - ▶  $x$  is free in  $P(x, y)$ , and bound in  $Q(x, y), R(x, y)$

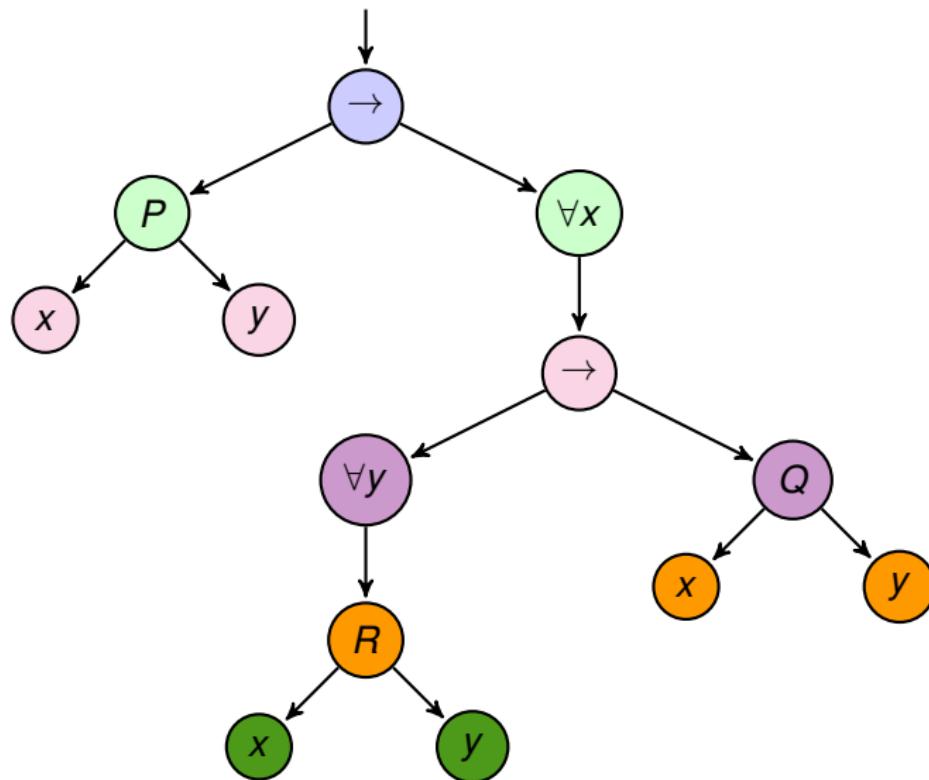
# Free and Bound Variables

---

- ▶ For a wff  $\varphi = \forall x\psi$  or  $\exists x\psi$ ,  $\psi$  is said to be the **scope** of the quantifier  $x$
- ▶ Every occurrence of  $x$  in  $\forall x\psi$  or  $\exists x\psi$  is **bound**
- ▶ Any occurrence of  $x$  which is not bound is called **free**
- ▶  $\varphi = P(x, y) \rightarrow \forall x((\forall y R(x, y)) \rightarrow Q(x, y))$ 
  - ▶  $y$  is free in  $Q(x, y)$  and bound in  $R(x, y)$ ,
  - ▶  $x$  is free in  $P(x, y)$ , and bound in  $Q(x, y), R(x, y)$
- ▶ Given  $\varphi$ , denote by  $\varphi(x_1, \dots, x_n)$ , that  $x_1, \dots, x_n$  are the free variables of  $\varphi$ , also *free*( $\varphi$ )
- ▶ A **sentence** is a formula  $\varphi$  **none** of whose variables are **free**

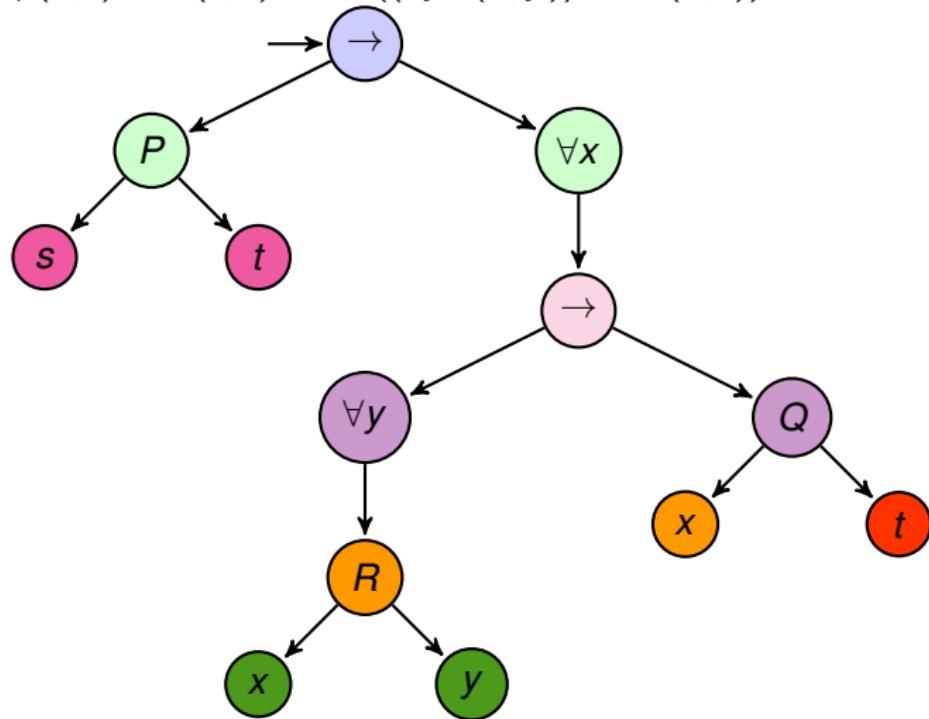
$$P(x, y) \rightarrow \forall x((\forall y R(x, y)) \rightarrow Q(x, y))$$

---



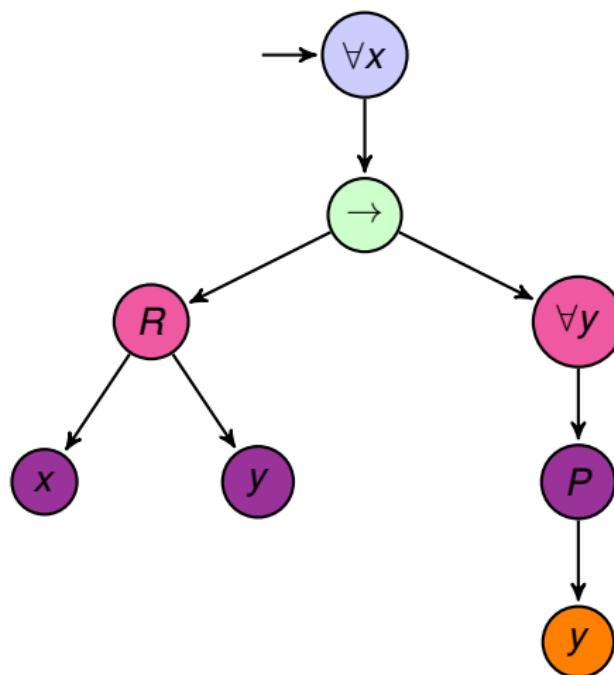
$$P(x, y) \rightarrow \forall x((\forall y R(x, y)) \rightarrow Q(x, y))$$

$$\varphi(s, t) = P(s, t) \rightarrow \forall x((\forall y R(x, y)) \rightarrow Q(x, t))$$



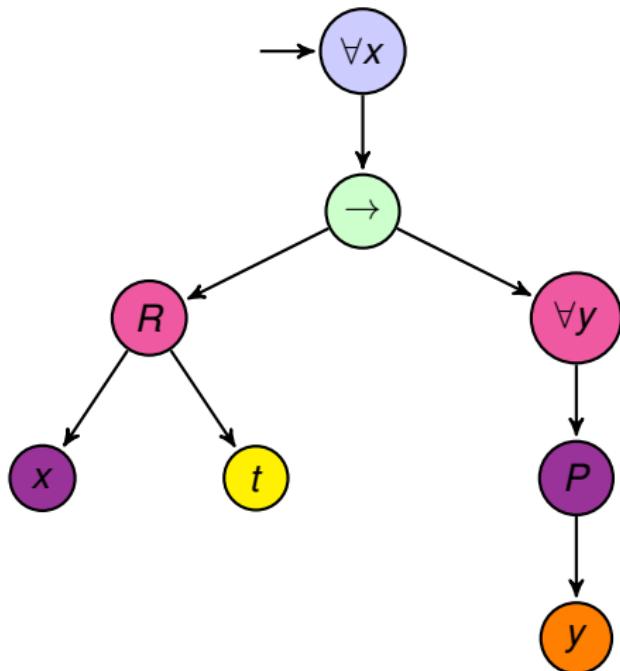
$$\forall x(R(x, y) \rightarrow \forall y P(y))$$

---



$$\forall x(R(x, y) \rightarrow \forall y P(y))$$

---



$$\varphi(t) = \forall x(R(x, t) \rightarrow \forall y P(y))$$

# Assignments on $\tau$ -structures

---

## Assignments

For a  $\tau$ -structure  $\mathcal{A}$ , an assignment over  $\mathcal{A}$  is a function  $\alpha : \mathcal{V} \rightarrow u(\mathcal{A})$  that assigns every variable  $x \in \mathcal{V}$  a value  $\alpha(x) \in u(\mathcal{A})$ . If  $t$  is a constant symbol  $c$ , then  $\alpha(t)$  is  $c^{\mathcal{A}}$

# Assignments on $\tau$ -structures

## Assignments

For a  $\tau$ -structure  $\mathcal{A}$ , an assignment over  $\mathcal{A}$  is a function  $\alpha : \mathcal{V} \rightarrow u(\mathcal{A})$  that assigns every variable  $x \in \mathcal{V}$  a value  $\alpha(x) \in u(\mathcal{A})$ . If  $t$  is a constant symbol  $c$ , then  $\alpha(t)$  is  $c^{\mathcal{A}}$

## Binding on a Variable

For an assignment  $\alpha$  over  $\mathcal{A}$ ,  $\alpha[x \mapsto a]$  is the assignment

$$\alpha[x \mapsto a](y) = \begin{cases} \alpha(y), & y \neq x, \\ a, & y = x \end{cases}$$

# Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

# Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

- ▶  $\mathcal{A} \not\models_{\alpha} \perp$

# Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

- $\mathcal{A} \not\models_{\alpha} \perp$
- $\mathcal{A} \models_{\alpha} t_1 = t_2$  iff  $\alpha(t_1) = \alpha(t_2)$

# Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

- ▶  $\mathcal{A} \not\models_{\alpha} \perp$
- ▶  $\mathcal{A} \models_{\alpha} t_1 = t_2$  iff  $\alpha(t_1) = \alpha(t_2)$
- ▶  $\mathcal{A} \models_{\alpha} R(t_1, \dots, t_k)$  iff  $(\alpha(t_1), \dots, \alpha(t_k)) \in R^{\mathcal{A}}$

# Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

- ▶  $\mathcal{A} \not\models_{\alpha} \perp$
- ▶  $\mathcal{A} \models_{\alpha} t_1 = t_2$  iff  $\alpha(t_1) = \alpha(t_2)$
- ▶  $\mathcal{A} \models_{\alpha} R(t_1, \dots, t_k)$  iff  $(\alpha(t_1), \dots, \alpha(t_k)) \in R^{\mathcal{A}}$
- ▶  $\mathcal{A} \models_{\alpha} (\varphi \rightarrow \psi)$  iff  $\mathcal{A} \not\models_{\alpha} \varphi$  or  $\mathcal{A} \models_{\alpha} \psi$

# Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

- ▶  $\mathcal{A} \not\models_{\alpha} \perp$
- ▶  $\mathcal{A} \models_{\alpha} t_1 = t_2$  iff  $\alpha(t_1) = \alpha(t_2)$
- ▶  $\mathcal{A} \models_{\alpha} R(t_1, \dots, t_k)$  iff  $(\alpha(t_1), \dots, \alpha(t_k)) \in R^{\mathcal{A}}$
- ▶  $\mathcal{A} \models_{\alpha} (\varphi \rightarrow \psi)$  iff  $\mathcal{A} \not\models_{\alpha} \varphi$  or  $\mathcal{A} \models_{\alpha} \psi$
- ▶  $\mathcal{A} \models_{\alpha} (\forall x)\varphi$  iff for every  $a \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} \varphi$

# Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

- ▶  $\mathcal{A} \not\models_{\alpha} \perp$
- ▶  $\mathcal{A} \models_{\alpha} t_1 = t_2$  iff  $\alpha(t_1) = \alpha(t_2)$
- ▶  $\mathcal{A} \models_{\alpha} R(t_1, \dots, t_k)$  iff  $(\alpha(t_1), \dots, \alpha(t_k)) \in R^{\mathcal{A}}$
- ▶  $\mathcal{A} \models_{\alpha} (\varphi \rightarrow \psi)$  iff  $\mathcal{A} \not\models_{\alpha} \varphi$  or  $\mathcal{A} \models_{\alpha} \psi$
- ▶  $\mathcal{A} \models_{\alpha} (\forall x)\varphi$  iff for every  $a \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} \varphi$
- ▶  $\mathcal{A} \models_{\alpha} (\exists x)\varphi$  iff there is some  $a \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} \varphi$

Last two cases,  $\alpha$  has no effect on the value of  $x$ . Thus, assignments matter **only** to free variables.

# Lecture 12 FOL SAT.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Recap

---

Signatures, Formulae over signatures, Structure for a signature

# Example of Satisfaction

---

- ▶  $\mathcal{G} = (\{1, 2, 3\}, E^{\mathcal{G}} = \{(1, 2), (2, 1), (2, 3), (3, 2)\})$ 
  - ▶ For any assignment  $\alpha$ ,  $\mathcal{G} \models_{\alpha} \forall x \forall y (E(x, y) \rightarrow E(y, x))$  iff for every  $a, b \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a, y \mapsto b]} (E(x, y) \rightarrow E(y, x))$

# Example of Satisfaction

---

- ▶  $\mathcal{G} = (\{1, 2, 3\}, E^{\mathcal{G}} = \{(1, 2), (2, 1), (2, 3), (3, 2)\})$ 
  - ▶ For any assignment  $\alpha$ ,  $\mathcal{G} \models_{\alpha} \forall x \forall y (E(x, y) \rightarrow E(y, x))$  iff for every  $a, b \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a, y \mapsto b]} (E(x, y) \rightarrow E(y, x))$
  - ▶ There is an assignment  $\alpha$  which satisfies  
 $\mathcal{G} \models_{\alpha} \exists x (E(x, y) \wedge E(x, z) \wedge y \neq z)$

# Example of Satisfaction

---

- ▶  $\mathcal{G} = (\{1, 2, 3\}, E^{\mathcal{G}} = \{(1, 2), (2, 1), (2, 3), (3, 2)\})$ 
  - ▶ For any assignment  $\alpha$ ,  $\mathcal{G} \models_{\alpha} \forall x \forall y (E(x, y) \rightarrow E(y, x))$  iff for every  $a, b \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a, y \mapsto b]} (E(x, y) \rightarrow E(y, x))$
  - ▶ There is an assignment  $\alpha$  which satisfies  $\mathcal{G} \models_{\alpha} \exists x (E(x, y) \wedge E(x, z) \wedge y \neq z)$
  - ▶ There is no assignment  $\alpha$  which satisfies  $\exists x \forall y (E(x, y))$
- ▶  $\mathcal{W} = abaaa$

# Example of Satisfaction

---

- ▶  $\mathcal{G} = (\{1, 2, 3\}, E^{\mathcal{G}} = \{(1, 2), (2, 1), (2, 3), (3, 2)\})$ 
  - ▶ For any assignment  $\alpha$ ,  $\mathcal{G} \models_{\alpha} \forall x \forall y (E(x, y) \rightarrow E(y, x))$  iff for every  $a, b \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a, y \mapsto b]} (E(x, y) \rightarrow E(y, x))$
  - ▶ There is an assignment  $\alpha$  which satisfies  $\mathcal{G} \models_{\alpha} \exists x (E(x, y) \wedge E(x, z) \wedge y \neq z)$
  - ▶ There is no assignment  $\alpha$  which satisfies  $\exists x \forall y (E(x, y))$
- ▶  $\mathcal{W} = abaaa$ 
  - ▶ There is an assignment  $\alpha$  for which  $\mathcal{W} \models_{\alpha} (Q_a(x) \wedge Q_a(y) \wedge S(x, y))$

# Example of Satisfaction

---

- ▶  $\mathcal{G} = (\{1, 2, 3\}, E^{\mathcal{G}} = \{(1, 2), (2, 1), (2, 3), (3, 2)\})$ 
  - ▶ For any assignment  $\alpha$ ,  $\mathcal{G} \models_{\alpha} \forall x \forall y (E(x, y) \rightarrow E(y, x))$  iff for every  $a, b \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a, y \mapsto b]} (E(x, y) \rightarrow E(y, x))$
  - ▶ There is an assignment  $\alpha$  which satisfies  
 $\mathcal{G} \models_{\alpha} \exists x (E(x, y) \wedge E(x, z) \wedge y \neq z)$
  - ▶ There is no assignment  $\alpha$  which satisfies  $\exists x \forall y (E(x, y))$
- ▶  $\mathcal{W} = abaaa$ 
  - ▶ There is an assignment  $\alpha$  for which  
 $\mathcal{W} \models_{\alpha} (Q_a(x) \wedge Q_a(y) \wedge S(x, y))$
  - ▶ There is no assignment  $\alpha$  which satisfies  
 $\exists x \exists y (Q_b(x) \wedge Q_b(y) \wedge x \neq y)$

# Satisfiability, Validity and Equivalence

---

- ▶ A formula  $\varphi$  over a signature  $\tau$  is said to be **satisfiable** iff for some  $\tau$ -structure  $\mathcal{A}$  and assignment  $\alpha$ ,  $\mathcal{A} \models_\alpha \varphi$

# Satisfiability, Validity and Equivalence

---

- ▶ A formula  $\varphi$  over a signature  $\tau$  is said to be **satisfiable** iff for some  $\tau$ -structure  $\mathcal{A}$  and assignment  $\alpha$ ,  $\mathcal{A} \models_{\alpha} \varphi$
- ▶ A formula  $\varphi$  over a signature  $\tau$  is said to be **valid** iff for every  $\tau$ -structure  $\mathcal{A}$  and assignment  $\alpha$ ,  $\mathcal{A} \models_{\alpha} \varphi$

# Satisfiability, Validity and Equivalence

---

- ▶ A formula  $\varphi$  over a signature  $\tau$  is said to be **satisfiable** iff for some  $\tau$ -structure  $\mathcal{A}$  and assignment  $\alpha$ ,  $\mathcal{A} \models_{\alpha} \varphi$
- ▶ A formula  $\varphi$  over a signature  $\tau$  is said to be **valid** iff for every  $\tau$ -structure  $\mathcal{A}$  and assignment  $\alpha$ ,  $\mathcal{A} \models_{\alpha} \varphi$
- ▶ Formulae  $\varphi(x_1, \dots, x_n)$  and  $\psi(x_1, \dots, x_n)$  are **equivalent** denoted  $\varphi \equiv \psi$  iff for every  $\mathcal{A}$  and  $\alpha$ ,  $\mathcal{A} \models_{\alpha} \varphi$  iff  $\mathcal{A} \models_{\alpha} \psi$

# Equisatisfiability

---

Let  $\varphi_1(x) = \forall y R(x, y)$  and  $\varphi_2 = \exists x \forall y R(x, y)$ .

- ▶ It is clear that whenever  $\mathcal{A} \models \varphi_2$ , one can find an assignment  $\alpha$  such that  $\mathcal{A} \models_{\alpha} \varphi_1(x)$ .
- ▶ Likewise, if  $\mathcal{A} \models_{\alpha} \varphi_1(x)$ , then  $\mathcal{A} \models \varphi_2$ .
- ▶ Thus,  $\varphi_1(x), \varphi_2$  are **equisatisfiable**.

# True or False?

---

For a formula  $\varphi$  and assignments  $\alpha_1$  and  $\alpha_2$  such that for every  $x \in \text{free}(\varphi)$ ,  $\alpha_1(x) = \alpha_2(x)$ ,  $\mathcal{A} \models_{\alpha_1} \varphi$  iff  $\mathcal{A} \models_{\alpha_2} \varphi$

# True or False?

---

For a formula  $\varphi$  and assignments  $\alpha_1$  and  $\alpha_2$  such that for every  $x \in \text{free}(\varphi)$ ,  $\alpha_1(x) = \alpha_2(x)$ ,  $\mathcal{A} \models_{\alpha_1} \varphi$  iff  $\mathcal{A} \models_{\alpha_2} \varphi$

- ▶ For example,  $\varphi(y) = \forall x(R(x, y) \rightarrow \forall z P(z))$

# True or False?

---

For a formula  $\varphi$  and assignments  $\alpha_1$  and  $\alpha_2$  such that for every  $x \in \text{free}(\varphi)$ ,  $\alpha_1(x) = \alpha_2(x)$ ,  $\mathcal{A} \models_{\alpha_1} \varphi$  iff  $\mathcal{A} \models_{\alpha_2} \varphi$

- ▶ For example,  $\varphi(y) = \forall x(R(x, y) \rightarrow \forall z P(z))$
- ▶ Consider two assignments  $\alpha_1, \alpha_2$  such that  $\alpha_1(y) = \alpha_2(y) = \alpha$  (say)

# True or False?

---

For a formula  $\varphi$  and assignments  $\alpha_1$  and  $\alpha_2$  such that for every  $x \in \text{free}(\varphi)$ ,  $\alpha_1(x) = \alpha_2(x)$ ,  $\mathcal{A} \models_{\alpha_1} \varphi$  iff  $\mathcal{A} \models_{\alpha_2} \varphi$

- ▶ For example,  $\varphi(y) = \forall x(R(x, y) \rightarrow \forall z P(z))$
- ▶ Consider two assignments  $\alpha_1, \alpha_2$  such that  $\alpha_1(y) = \alpha_2(y) = \alpha$  (say)
- ▶ Evaluate for all  $a, b \in u(\mathcal{A})$ ,  $R(a, \alpha) \rightarrow P(b)$

# True or False?

---

For a formula  $\varphi$  and assignments  $\alpha_1$  and  $\alpha_2$  such that for every  $x \in \text{free}(\varphi)$ ,  $\alpha_1(x) = \alpha_2(x)$ ,  $\mathcal{A} \models_{\alpha_1} \varphi$  iff  $\mathcal{A} \models_{\alpha_2} \varphi$

- ▶ For example,  $\varphi(y) = \forall x(R(x, y) \rightarrow \forall z P(z))$
- ▶ Consider two assignments  $\alpha_1, \alpha_2$  such that  $\alpha_1(y) = \alpha_2(y) = \alpha$  (say)
- ▶ Evaluate for all  $a, b \in u(\mathcal{A})$ ,  $R(a, \alpha) \rightarrow P(b)$
- ▶  $\mathcal{A} \models_{\alpha_1} \varphi$  iff  $\mathcal{A} \models_{\alpha_2} \varphi$

# True or False?

---

For a sentence  $\varphi$ , and any two assignments  $\alpha_1$  and  $\alpha_2$ ,  $\mathcal{A} \models_{\alpha_1} \varphi$  iff  
 $\mathcal{A} \models_{\alpha_2} \varphi$

# True or False?

---

For a sentence  $\varphi$ , and any two assignments  $\alpha_1$  and  $\alpha_2$ ,  $\mathcal{A} \models_{\alpha_1} \varphi$  iff  
 $\mathcal{A} \models_{\alpha_2} \varphi$

No free variables!

# Check Satisfiability

---

Let  $\tau$  be a signature with a single unary relation  $P$ . Consider the structure  $\mathcal{A} = (U_{\mathcal{A}} = \{0, 1\}, P^{\mathcal{A}} = \{1\})$ .

Let  $\varphi = \forall x_1 \forall x_2 \dots \forall x_n (P(x_1) \rightarrow (P(x_2) \rightarrow (P(x_3) \dots \rightarrow (P(x_n) \rightarrow P(x_1)) \dots)))$ .

Does  $\mathcal{A} \models \varphi$ ?

# Check Satisfiability

---

Let  $\varphi(y) = \exists x(E(x, y) \wedge \neg(y = x) \wedge \forall z[E(z, y) \rightarrow z = x])$  over the signature  $\tau$  containing a binary relation  $E$ . Is  $\varphi(y)$  satisfiable under some graph structure?

# Lecture 13 FOL SAT Translation.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Check Satisfiability

---

Let  $\psi(z) = \exists x [Q_a(x) \wedge \forall y [(y \leq x \wedge Q_b(y)) \rightarrow (z < x \wedge y < z \wedge Q_c(z))]]$   
over the signature  $\tau$  having the relational symbols  $<$ ,  $Q_a$ ,  $Q_b$ ,  $Q_c$  and  
unary function  $S$ . Does  $\psi(z)$  evaluate to true under some word  
structure?

# Check Satisfiability

---

Let  $\zeta = P(0) \wedge \forall x(P(x) \rightarrow P(S(x))) \wedge \exists x \neg P(x)$  over a signature  $\tau$  containing the constant 0, unary function  $S$  and unary relation  $P$ .  
Is  $\zeta$  satisfiable?

## Normal Forms in FOL

# Recap : Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

- ▶  $\mathcal{A} \not\models_{\alpha} \perp$
- ▶  $\mathcal{A} \models_{\alpha} t_1 = t_2$  iff  $\alpha(t_1) = \alpha(t_2)$
- ▶  $\mathcal{A} \models_{\alpha} R(t_1, \dots, t_k)$  iff  $(\alpha(t_1), \dots, \alpha(t_k)) \in R^{\mathcal{A}}$
- ▶  $\mathcal{A} \models_{\alpha} (\varphi \rightarrow \psi)$  iff  $\mathcal{A} \not\models_{\alpha} \varphi$  or  $\mathcal{A} \models_{\alpha} \psi$
- ▶  $\mathcal{A} \models_{\alpha} (\forall x)\varphi$  iff for every  $a \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} \varphi$
- ▶  $\mathcal{A} \models_{\alpha} (\exists x)\varphi$  iff there is some  $a \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} \varphi$

Last two cases,  $\alpha$  has no effect on the value of  $x$ . Thus, assignments matter **only** to free variables.

# Equivalences

---

Let  $F, G$  be arbitrary FOL formulae.

1.  $\neg \forall x F \equiv \exists x \neg F$
2.  $\neg \exists x F \equiv \forall x \neg F$

$\mathcal{A} \models_{\alpha} \neg \forall x F$  iff  $\mathcal{A} \not\models_{\alpha} \forall x F$   
iff  $\mathcal{A} \not\models_{\alpha[x \mapsto a]} F$  for some  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[x \mapsto a]} \neg F$  for some  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha} \exists x \neg F$

# Equivalences

If  $x$  does not occur free in  $G$  then

1.  $(\forall x F \wedge G) \equiv \forall x(F \wedge G)$
2.  $(\forall x F \vee G) \equiv \forall x(F \vee G)$
3.  $(\exists x F \wedge G) \equiv \exists x(F \wedge G)$
4.  $(\exists x F \vee G) \equiv \exists x(F \vee G)$

$\mathcal{A} \models_{\alpha} \forall x F \wedge G$  iff  $\mathcal{A} \models_{\alpha} \forall x F$  and  $\mathcal{A} \models_{\alpha} G$

iff for all  $a \in U^{\mathcal{A}}$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} F$  and  $\mathcal{A} \models_{\alpha} G$

iff for all  $a \in U^{\mathcal{A}}$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} F$  and  $\mathcal{A} \models_{\alpha[x \mapsto a]} G$

iff for all  $a \in U^{\mathcal{A}}$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} (F \wedge G)$

iff  $\mathcal{A} \models \forall x(F \wedge G)$

# Equivalences

---

Let  $F, G$  be arbitrary FOL formulae.

$$1. (\forall x F \wedge \forall x G) \equiv \forall x (F \wedge G)$$

$$2. (\exists x F \vee \exists x G) \equiv \exists x (F \vee G)$$

$$1. \forall x \forall y F \equiv \forall y \forall x F$$

$$2. \exists x \exists y F \equiv \exists y \exists x F$$

# Recap : Terms

---

Given a signature  $\tau$ , the set of  $\tau$ -terms are defined inductively as follows.

- ▶ Each variable is a term
- ▶ Each constant symbol is a term
- ▶ If  $t_1, \dots, t_k$  are terms and  $f$  is a  $k$ -ary function, then  $f(t_1, \dots, t_k)$  is a term
- ▶ Ground Terms : Terms without variables. For instance  $f(c_1, \dots, c_k)$  for constants  $c_1, \dots, c_k$ .

# Translation Lemma

---

## Translation Lemma

If  $t$  is a term and  $F$  is a formula such that no variable in  $t$  occurs bound in  $F$ , then  $\mathcal{A} \models_{\alpha} F[t/x]$  iff  $\mathcal{A} \models_{\alpha[x \mapsto \alpha(t)]} F$ .

$F[t/x]$  denotes substituting  $t$  for  $x$  in  $F$ , where  $x$  is free in  $F$

- ▶ What if  $t$  contains a variable bound in  $F$ ?
- ▶ Results in *Variable Capture*

# Translation Lemma Proof : Optional

---

Proof by Induction on formulae.

- ▶ Base case. Atomic formulae  $P(t_1, \dots, t_k)$ .
- ▶  $\mathcal{A} \models_{\alpha} P(t_1, \dots, t_k)[t/x]$  iff  $\mathcal{A} \models_{\alpha} P(t_1[t/x], \dots, t_k[t/x])$ .
- ▶ Show that  $\mathcal{A} \models_{\alpha[x \mapsto \alpha(t)]} P(t_1, \dots, t_k)$ .
  - ▶ Base Cases within :  $t_i = c$ ,  $t_i = y$  for  $y \neq x$ ,  $t_i = x$  for each  $t_i$ .
  - ▶ Case  $t_i = f(s_1, \dots, s_j)$  for a function  $f$ .
  - ▶  $f(s_1, \dots, s_j)[t/x] = f(s_1[t/x], \dots, s_j[t/x])$
- ▶  $\mathcal{A} \models_{\alpha} P(t_1[t/x], \dots, t_k[t/x])$  iff  $(\alpha(t_1[t/x]), \dots, \alpha(t_k[t/x])) \in P^{\mathcal{A}}$
- ▶ iff  $(\alpha([x \mapsto \alpha(t)](t_1), \dots, \alpha([x \mapsto \alpha(t)](t_k)) \in P^{\mathcal{A}}$
- ▶ iff  $\mathcal{A} \models_{\alpha[x \mapsto \alpha(t)]} P(t_1, \dots, t_k)$
- ▶ Cases for formulae with propositional connectives is routine.
- ▶ Case with quantifier,  $\forall y F[t/x]$ ,  $\exists y F[t/x]$  where  $y \neq x$ .

# Renaming

---

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$

# Renaming

---

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto \alpha[y \mapsto a](y)]} G$  for all  $a \in U^{\mathcal{A}}$   
(Translation Lemma)

# Renaming

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto \alpha[y \mapsto a](y)]} G$  for all  $a \in U^{\mathcal{A}}$   
(Translation Lemma)  
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$

# Renaming

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto \alpha[y \mapsto a](y)]} G$  for all  $a \in U^{\mathcal{A}}$   
(Translation Lemma)  
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$

# Renaming

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto \alpha[y \mapsto a](y)]} G$  for all  $a \in U^{\mathcal{A}}$   
(Translation Lemma)  
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_\alpha \forall x G$

# Lecture 14 FOL RPF Skolemisation.pdf

# CS 228 : Logic in Computer Science

Krishna. S

## Normal Forms in FOL

# Translation Lemma

---

## Translation Lemma

If  $t$  is a term and  $F$  is a formula such that no variable in  $t$  occurs bound in  $F$ , then  $\mathcal{A} \models_{\alpha} F[t/x]$  iff  $\mathcal{A} \models_{\alpha[x \mapsto \alpha(t)]} F$ .

$F[t/x]$  denotes substituting  $t$  for  $x$  in  $F$ , where  $x$  is free in  $F$

- ▶ What if  $t$  contains a variable bound in  $F$ ?
- ▶ Results in *Variable Capture*

# Translation Lemma Proof : Optional

---

Proof by Induction on formulae.

- ▶ Base case. Atomic formulae  $P(t_1, \dots, t_k)$ .
- ▶  $\mathcal{A} \models_{\alpha} P(t_1, \dots, t_k)[t/x]$  iff  $\mathcal{A} \models_{\alpha} P(t_1[t/x], \dots, t_k[t/x])$ .
- ▶ Show that  $\mathcal{A} \models_{\alpha[x \mapsto \alpha(t)]} P(t_1, \dots, t_k)$ .
  - ▶ Base Cases within :  $t_i = c$ ,  $t_i = y$  for  $y \neq x$ ,  $t_i = x$  for each  $t_i$ .
  - ▶ Case  $t_i = f(s_1, \dots, s_j)$  for a function  $f$ .
  - ▶  $f(s_1, \dots, s_j)[t/x] = f(s_1[t/x], \dots, s_j[t/x])$
- ▶  $\mathcal{A} \models_{\alpha} P(t_1[t/x], \dots, t_k[t/x])$  iff  $(\alpha(t_1[t/x]), \dots, \alpha(t_k[t/x])) \in P^{\mathcal{A}}$
- ▶ iff  $(\alpha([x \mapsto \alpha(t)](t_1), \dots, \alpha([x \mapsto \alpha(t)](t_k)) \in P^{\mathcal{A}}$
- ▶ iff  $\mathcal{A} \models_{\alpha[x \mapsto \alpha(t)]} P(t_1, \dots, t_k)$
- ▶ Cases for formulae with propositional connectives is routine.
- ▶ Case with quantifier,  $\forall y F[t/x]$ ,  $\exists y F[t/x]$  where  $y \neq x$ .

# Renaming

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$

# Renaming

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto \alpha[y \mapsto a](y)]} G$  for all  $a \in U^{\mathcal{A}}$   
(Translation Lemma)

# Renaming

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto \alpha[y \mapsto a](y)]} G$  for all  $a \in U^{\mathcal{A}}$   
(Translation Lemma)  
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$

# Renaming

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto \alpha[y \mapsto a](y)]} G$  for all  $a \in U^{\mathcal{A}}$   
(Translation Lemma)  
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$

# Renaming

$\int_0^\infty f(s)ds$  has the same value as  $\int_0^\infty f(t)dt$

## Renaming Lemma

Let  $F = Qx[G]$  be a formula with  $Q \in \{\exists, \forall\}$ . Let  $y$  be a variable which does not appear in  $G$ . Then  $\mathcal{A} \models_\alpha F$  iff  $\mathcal{A} \models_\alpha Qy(G[y/x])$ .

Assume  $Q = \forall$ .

$\mathcal{A} \models_\alpha \forall y G[y/x]$  iff  $\mathcal{A} \models_{\alpha[y \mapsto a]} G[y/x]$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto \alpha[y \mapsto a](y)]} G$  for all  $a \in U^{\mathcal{A}}$   
(Translation Lemma)  
iff  $\mathcal{A} \models_{\alpha[y \mapsto a, x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_{\alpha[x \mapsto a]} G$  for all  $a \in U^{\mathcal{A}}$   
iff  $\mathcal{A} \models_\alpha \forall x G$

# Rectified Formulae

---

A FOL formula is *rectified* if no variable occurs both free and bound, and if all quantifiers in the formula refer to different variables.

$$\forall x \exists y P(x, f(y)) \wedge \forall y (Q(x, y) \vee R(x))$$

is not rectified. By renaming we obtain an equivalent rectified formula

$$\forall u \exists v P(u, f(v)) \wedge \forall y (Q(x, y) \vee R(x))$$

By Renaming Lemma, we can always obtain an equivalent rectified formula by renaming bound variables.

# Prenex Normal Form

---

A formula is in prenex normal form if it can be written as

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n F$$

where  $Q_i \in \{\forall, \exists\}$ ,  $n \geq 0$  and  $F$  has no quantifiers.  $F$  is called the matrix of the formula.

# Prenex Normal Form : Example

---

Convert the rectified formula  $\neg(\exists xP(x, y) \vee \forall zQ(z)) \wedge \exists wQ(w)$  to Prenex Normal Form

- ▶  $(\neg\exists xP(x, y) \wedge \neg\forall zQ(z)) \wedge \exists wQ(w)$
- ▶  $(\forall x\neg P(x, y) \wedge \exists z\neg Q(z)) \wedge \exists wQ(w)$
- ▶  $\forall x\exists z(\neg P(x, y) \wedge \neg Q(z)) \wedge \exists wQ(w)$
- ▶  $\forall x\exists z\exists w((\neg P(x, y) \wedge \neg Q(z)) \wedge Q(w))$
- ▶ Note that we have used the equivalences from the last lecture

# Rectified, Prenex normal form (RPF)

---

- ▶ Given a rectified formula, we can use the equivalences from the last lecture to convert  $F$  into rectified, prenex normal form, by “pushing all quantifiers up front”.
- ▶ Otherwise, rectify the formula first, and then convert to prenex normal form.

Every formula is equivalent to a rectified formula in prenex normal form.

# Skolemisation

---

A formula in RPF is in *Skolem form* if it has no occurrences of the existential quantifier.

We can transform any formula in RPF to an equisatisfiable formula in Skolem form by using extra function symbols.

- ▶  $\forall x \exists y P(x, y)$  is equisatisfiable with  $\forall x P(x, f(x))$ .
- ▶  $\forall x \forall z \exists y P(x, y, z)$  is equisatisfiable with  $\forall x \forall z P(x, f(x, z), z)$ .
- ▶  $\exists x \forall y G(x, y)$  is equisatisfiable with  $\forall y G(c, y)$  where  $c$  is a constant.
- ▶  $\exists x \forall y \exists z \exists w G(x, y, z, w)$  is equisatisfiable with  $\forall y G(c, y, f(y), g(y))$  where  $c$  is a constant.

# Skolemisation

## Skolem Lemma

Let  $F = \forall y_1 \forall y_2 \dots \forall y_n \exists z G$  be in RPF. Given a function symbol  $f$  of arity  $n$  which does not appear in  $F$ , write

$$F' = \forall y_1 \forall y_2 \dots \forall y_n G[f(y_1, \dots, y_n)/z]$$

Then  $F$  and  $F'$  are equisatisfiable.

Assume  $F$  is satisfiable. Let  $\mathcal{A} \models_{\alpha} F$ .

- ▶ Extend structure  $\mathcal{A}$  with an interpretation for a function  $f$  such that  $\mathcal{A}' \models_{\alpha'} F'$ .
- ▶ Given  $a_1, \dots, a_n \in U^{\mathcal{A}}$ , choose  $a \in U^{\mathcal{A}}$  such that  $\mathcal{A} \models_{\alpha[y_1 \mapsto a_1, \dots, y_n \mapsto a_n, z \mapsto a]} G$ , and define  $f^{\mathcal{A}'}(a_1, \dots, a_n) = a$ .
- ▶  $f$  does not appear in  $G$ ,  $\mathcal{A}' \models_{\alpha[y_1 \mapsto a_1, \dots, y_n \mapsto a_n, z \mapsto f^{\mathcal{A}'}(a_1, \dots, a_n)]} G$ ,
- ▶ By Translation Lemma,  $\mathcal{A}' \models_{\alpha[y_1 \mapsto a_1, \dots, y_n \mapsto a_n]} G[f(y_1, \dots, y_n)/z]$
- ▶ Since this holds for any  $a_1, \dots, a_n \in U^{\mathcal{A}}$ ,  
 $\mathcal{A}' \models \forall y_1 \forall y_2 \dots \forall y_n G[f(y_1, \dots, y_n)/z]$

# Skolemisation : Example

---

$$\forall x \exists y \forall z \exists w (\neg P(a, w) \vee Q(f(x), y))$$

- ▶ By Skolem Lemma, eliminate  $\exists y$  and introduce a new function  $g$ , obtaining  $\forall x \forall z \exists w (\neg P(a, w) \vee Q(f(x), g(x)))$
- ▶ By Skolem Lemma, eliminate  $\exists w$  introducing a new function  $h$  obtaining  $\forall x \forall z (\neg P(a, h(x, z)) \vee Q(f(x), g(x)))$

# Conversion to Skolem Form : Summary

---

Convert an arbitrary FOL formula to an equisatisfiable formula in Skolem formula as follows:

1. Rectify  $F$  systematically renaming bound variables, obtaining an equivalent formula  $F_1$
2. Use the equivalences in the beginning and move all quantifiers outside, yielding an equivalent formula  $F_2$  in prenex normal form
3. Repeatedly eliminate the outermost existential quantifier in  $F_2$  until an equisatisfiable formula  $F_3$  is obtained in Skolem form.

# Semi Decidability of Satisfiability

---

- ▶ Given a FOL formula in Skolem normal form, if  $F$  is unsatisfiable, there is a technique of *Ground Resolution* which gives  $\perp$  and terminates.
- ▶ However, if  $F$  is satisfiable, then this process may go on forever.
- ▶ Validity is *semi decidable* : a valid formula  $F$  has a finite witness of its validity, namely, a finite resolution refutation for  $\neg F$ .
- ▶ If  $F$  is not valid, and satisfiable, then there may not be a finite witness.
- ▶ This is for general FOL : however, we can focus on FOL over some special signatures where satisfiability is decidable.

# Lecture 15 FOL Words.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Satisfaction, Validity

---

- ▶ Given a FO formula  $\varphi(x_1, \dots, x_n)$  over a signature  $\tau$ , is it satisfiable/valid?
  - ▶ Satisfiable, if there exists a  $\tau$ -structure  $\mathcal{A}$  and an assignment  $\alpha$  for  $x_1, \dots, x_n$  in  $u(\mathcal{A})$  such that  $\mathcal{A} \models_\alpha \varphi(x_1, \dots, x_n)$

# Satisfaction, Validity

---

- ▶ Given a FO formula  $\varphi(x_1, \dots, x_n)$  over a signature  $\tau$ , is it satisfiable/valid?
  - ▶ Satisfiable, if there exists a  $\tau$ -structure  $\mathcal{A}$  and an assignment  $\alpha$  for  $x_1, \dots, x_n$  in  $u(\mathcal{A})$  such that  $\mathcal{A} \models_{\alpha} \varphi(x_1, \dots, x_n)$
  - ▶ Valid, if for any  $\tau$ -structure  $\mathcal{A}$  and any assignment  $\alpha$  for  $x_1, \dots, x_n$  in  $u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha} \varphi(x_1, \dots, x_n)$
- ▶ Assume we fix the type of the structure  $\mathcal{A}$ , say words (why words?)

# FOL over Words

---

- ▶ Given an FO sentence  $\varphi$  over words, is it satisfiable/valid?
- ▶ Satisfiable (**Valid**) iff some word (**all words**) satisfies  $\varphi$

# FOL over Words

---

- ▶ Given an FO sentence  $\varphi$  over words, is it satisfiable/valid?
- ▶ Satisfiable (Valid) iff some word (all words) satisfies  $\varphi$
- ▶ There could be infinitely many words  $w$  satisfying  $\varphi$
- ▶  $L(\varphi) = \{ \text{words } w \mid w \models \varphi\}$  is called the language of  $\varphi$

# FOL over Words

---

- ▶ Given an FO sentence  $\varphi$  over words, is it satisfiable/valid?
- ▶ Satisfiable (Valid) iff some word (all words) satisfies  $\varphi$
- ▶ There could be infinitely many words  $w$  satisfying  $\varphi$
- ▶  $L(\varphi) = \{ \text{words } w \mid w \models \varphi\}$  is called the language of  $\varphi$
- ▶ Given  $\varphi$ , write an algorithm to check  $L(\varphi) = \emptyset$

# First-Order Logic over Words

# Expressiveness and Satisfiability

---

- ▶ Signature for words :  $<$ ,  $S$  and  $Q_a$  for finitely many symbols  $a$
- ▶ Given a FO formula over words, the signature is fixed

# Expressiveness and Satisfiability

---

- ▶ Signature for words :  $<$ ,  $S$  and  $Q_a$  for finitely many symbols  $a$
- ▶ Given a FO formula over words, the signature is fixed
- ▶ Expressiveness
  - ▶ Given a set of words or a language  $L$ , can you write a FO formula  $\varphi$  such that  $L(\varphi) = L$

# Expressiveness and Satisfiability

---

- ▶ Signature for words :  $<$ ,  $S$  and  $Q_a$  for finitely many symbols  $a$
- ▶ Given a FO formula over words, the signature is fixed
- ▶ Expressiveness
  - ▶ Given a set of words or a language  $L$ , can you write a FO formula  $\varphi$  such that  $L(\varphi) = L$
  - ▶ If you can, FO is expressive enough to capture your language/specification/property

# Expressiveness and Satisfiability

---

- ▶ Signature for words :  $<$ ,  $S$  and  $Q_a$  for finitely many symbols  $a$
- ▶ Given a FO formula over words, the signature is fixed
- ▶ Expressiveness
  - ▶ Given a set of words or a language  $L$ , can you write a FO formula  $\varphi$  such that  $L(\varphi) = L$
  - ▶ If you can, FO is expressive enough to capture your language/specification/property
  - ▶ If you cannot, show that FO cannot capture your property.
- ▶ Satisfiability

# Expressiveness and Satisfiability

---

- ▶ Signature for words :  $<$ ,  $S$  and  $Q_a$  for finitely many symbols  $a$
- ▶ Given a FO formula over words, the signature is fixed
- ▶ Expressiveness
  - ▶ Given a set of words or a language  $L$ , can you write a FO formula  $\varphi$  such that  $L(\varphi) = L$
  - ▶ If you can, FO is expressive enough to capture your language/specification/property
  - ▶ If you cannot, show that FO cannot capture your property.
- ▶ Satisfiability
  - ▶ Given a FO formula  $\varphi$  over words, is  $L(\varphi)$  non-empty?

# A Primer for Words

# Alphabet

---

- ▶ An alphabet  $\Sigma$  is a finite set
  - ▶  $\Sigma = \{a, b, \dots, z\}$
  - ▶  $\Sigma = \{+, \alpha, 100, B\}$
- ▶ Elements of  $\Sigma$  called letters or symbols
- ▶ A word or string over  $\Sigma$  is a finite sequence of symbols from  $\Sigma$
- ▶ If  $\Sigma = \{a, b\}$ , then *abababa* is a word of length 7
- ▶ The length of a word  $w$  is denoted  $|w|$
- ▶ There is a unique word of length 0 denoted  $\epsilon$ , called the empty word
- ▶  $|\epsilon| = 0$

# Notations for Words

---

- ▶  $aaaaa$  denoted  $a^5$
- ▶  $a^0 = \epsilon$
- ▶  $a^{n+1} = a^n.a = a.a^n$
- ▶ The set of all words over  $\Sigma$  is denoted  $\Sigma^*$ 
  - ▶  $\{a, b\}^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, \dots\}$
  - ▶  $\{a\}^* = \{\epsilon, a, aa, aaa, \dots\} = \{a^n \mid n \geq 0\}$
- ▶ By convention,  $\{\}^* = \{\epsilon\}$

# Notations for Words

---

- ▶  $\Sigma$  is a finite set
- ▶  $\Sigma^*$  is the infinite set of all finite words over alphabet  $\Sigma$
- ▶ Each  $w \in \Sigma^*$  is a finite word
  - ▶  $\{a, b\} = \{b, a\}$  but  $ab \neq ba$
  - ▶  $\{a, a, b\} = \{a, b\}$  but  $aab \neq ab$
  - ▶  $\emptyset$  is the set consisting of no words
  - ▶  $\{\epsilon\}$  is a set having the single word  $\epsilon$
  - ▶  $\epsilon$  is a word

# Operations on Words

---

- ▶ Concatenation of words :  $x.y = xy$ 
  - ▶ Concatenation is associative :  $x.(yz) = (xy).z$
  - ▶ Concatenation not commutative in general  $x.y \neq y.x$
  - ▶  $\epsilon$  is the identity for concatenation  $\epsilon.x = x.\epsilon = x$
  - ▶  $|x.y| = |x| + |y|$
- ▶  $x^n$  : catenating word  $x$   $n$  times
  - ▶  $(aab)^5 = \textcolor{red}{aab} \textcolor{blue}{aba} \textcolor{magenta}{aab} \textcolor{green}{aba} \textcolor{blue}{aab}$
  - ▶  $(aab)^0 = \epsilon$
  - ▶  $(aab)^* = \{\epsilon, aab, aabaab, aabaabaab, \dots\}$
  - ▶  $x^{n+1} = x^n x$

# More Operations on Words

---

- ▶ For  $a \in \Sigma$  and  $x \in \Sigma^*$ ,

$|x|_a = \text{number of times the symbol } a \text{ occurs in the word } x$

- ▶  $|aabbaa|_a = 4, |aabbaa|_b = 2$
- ▶  $|\epsilon|_a = 0$
- ▶ Prefix of a word  $w \in \Sigma^*$  is an initial subword of  $w$

$$\text{Pref}(w) = \{x \in \Sigma^* \mid \exists y \in \Sigma^*, w = x.y\}$$

- ▶  $\text{Pref}(aaba} = \{\epsilon, a, aa, aab, aaba\}$
- ▶ Proper prefixes =  $\{a, aa, aab\}$
- ▶  $\epsilon, aaba$  improper prefixes

# Operation on Sets

---

Given a finite alphabet  $\Sigma$ , denote by  $A, B, C, \dots$  subsets of  $\Sigma^*$

- ▶ Subsets of  $\Sigma^*$  are called languages
- ▶  $A \cup B = \{x \in \Sigma^* \mid x \in A \text{ or } x \in B\}$ 
  - ▶  $A = a^*, B = \{b, bb\}, A \cup B = a^* \cup \{b, bb\}$
- ▶  $A \cap B = \{x \in \Sigma^* \mid x \in A \text{ and } x \in B\}$ 
  - ▶  $A = (ab)^*, B = \{abab, \epsilon, bb\}, A \cap B = \{\epsilon, abab\}$
- ▶  $\overline{A} = \{x \in \Sigma^* \mid x \notin A\}$ 
  - ▶ For  $\Sigma = \{a\}$  and  $A = (aa)^*$ ,  $\overline{A} = \{a, a^3, a^5, \dots\}$
- ▶  $AB = \{xy \mid x \in A, y \in B\}$ 
  - ▶  $A = \{a, ba\}, B = \{\epsilon, aa, bb\}$
  - ▶  $AB = \{a, a^3, abb, ba, ba^3, babb\}$
  - ▶  $BA = \{a, ba, a^3, aaba, bba, bbba\}$

# Operation on Sets

---

For a set  $A \subseteq \Sigma^*$ ,

- ▶  $A^0 = \{\epsilon\}$
- ▶  $A^{n+1} = A \cdot A^n$ 
  - ▶  $\{a, ab\}^2 = \{a, ab\} \cdot \{a, ab\} = \{aa, aab, aba, abab\}$
  - ▶  $\{a, b\}^n = \{x \in \{a, b\}^* \mid |x| = n\}$
- ▶  $A^* = A^0 \cup A \cup A^2 \cup \dots = \bigcup_{i \geq 0} A^i$
- ▶  $A^+ = AA^* = A \cup A^2 \cup \dots = \bigcup_{i \geq 1} A^i$
- ▶ Union : Associative, commutative
- ▶ Concatenation : Associative, Non commutative
- ▶  $A \cup \emptyset = \emptyset \cup A = A$
- ▶  $\{\epsilon\}A = A\{\epsilon\} = A$
- ▶  $\emptyset A = A\emptyset = \emptyset$

# Operation on Sets

---

- ▶ Union, Intersection distribute over union
  - ▶  $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
  - ▶  $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- ▶ Concatenation distributes over union
  - ▶  $A(\cup_{i \in I} B_i) = \cup_{i \in I} AB_i$
  - ▶  $(\cup_{i \in I} B_i)A = \cup_{i \in I} B_i A$
- ▶ Concatenation does not distribute over intersection
  - ▶  $A = \{a, ab\}, B = \{b\}, C = \{\epsilon\}$
  - ▶  $A(B \cap C) \neq AB \cap AC$

# FO for Languages

# Formalize in FO

---

Write FO formulae  $\varphi_i$  such that  $L(\varphi_i) = L_i$  for  $i = 1, \dots, 5$ .

- ▶  $L_1$  = Words that have exactly one occurrence of the letter  $c$
- ▶  $L_2$  = Words that begin with  $a$  and end with  $b$
- ▶  $L_3$  = Words that have no two consecutive  $a$ 's
- ▶  $L_4$  = Words in which any  $a$  is followed immediately by a  $b$
- ▶  $L_5$  = Words in which whenever an  $a$  occurs, it is followed eventually by a  $b$ , and no  $c$  occurs in between the  $a$  and the  $b$   
 $aabbabab, aabbcbccaaab \in L_5, aacaab \notin L_5.$

# Satisfiability of FO over Words

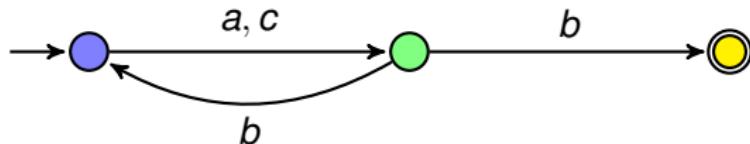
---

- ▶ Recall : Given an FO sentence  $\varphi$  over words, is  $L(\varphi) = \emptyset$ ?
- ▶ Algorithm?

# Core Idea

---

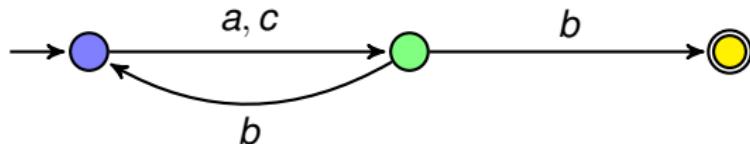
- Given FO formula  $\varphi$  over an alphabet  $\Sigma$ , construct an **edge labeled graph**  $G_\varphi$ : a graph whose edges are **labeled** by  $\Sigma$ .



# Core Idea

---

- Given FO formula  $\varphi$  over an alphabet  $\Sigma$ , construct an **edge labeled graph**  $G_\varphi$ : a graph whose edges are **labeled** by  $\Sigma$ .

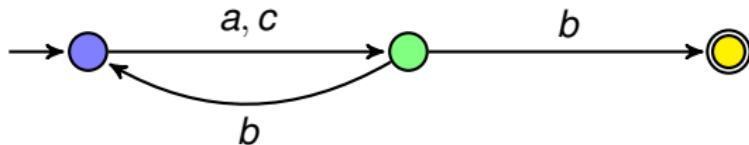


- Each path in the graph gives rise to a word over  $\Sigma$ , obtained by reading off the labels on the edges

# Core Idea

---

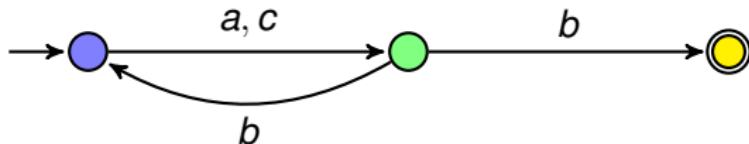
- Given FO formula  $\varphi$  over an alphabet  $\Sigma$ , construct an **edge labeled graph**  $G_\varphi$ : a graph whose edges are **labeled** by  $\Sigma$ .



- Each path in the graph gives rise to a word over  $\Sigma$ , obtained by reading off the labels on the edges
- $G_\varphi$  has some **special** kinds of vertices
  - There is a unique vertex called the **start** vertex (**blue** vertex)
  - There are some vertices called **good** vertices (**yellow** vertex)

# Core Idea

- Given FO formula  $\varphi$  over an alphabet  $\Sigma$ , construct an **edge labeled graph**  $G_\varphi$ : a graph whose edges are **labeled** by  $\Sigma$ .



- Each path in the graph gives rise to a word over  $\Sigma$ , obtained by reading off the labels on the edges
- $G_\varphi$  has some **special** kinds of vertices
  - There is a unique vertex called the **start** vertex (**blue** vertex)
  - There are some vertices called **good** vertices (**yellow** vertex)
- Read off words on paths from the start vertex to any final vertex and call this set of words  $L(G_\varphi)$
- Ensure that  $G_\varphi$  is constructed such that  $L(\varphi) = L(G_\varphi)$ .

# Core Idea

---

- ▶ Why does this help?

# Core Idea

---

- ▶ Why does this help?
- ▶ We know how to check the existence of a path between 2 vertices in a graph **easily**

# Core Idea

---

- ▶ Why does this help?
- ▶ We know how to check the existence of a path between 2 vertices in a graph **easily**
- ▶ If **somewhat** we manage to construct  $G_\varphi$  **correctly**, then checking satisfiability of  $\varphi$  is same as checking the **reachability** of some good vertex from the start vertex of  $G_\varphi$ .

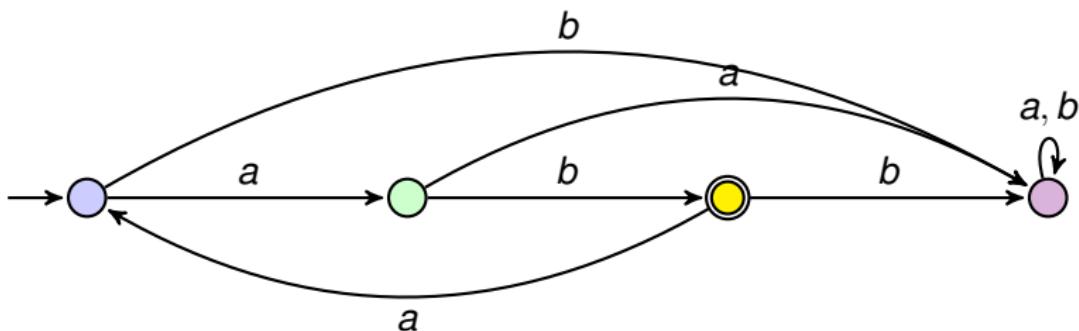
# Core Idea

---

- ▶ Why does this help?
- ▶ We know how to check the existence of a path between 2 vertices in a graph **easily**
- ▶ If **somewhat** we manage to construct  $G_\varphi$  **correctly**, then checking satisfiability of  $\varphi$  is same as checking the **reachability** of some good vertex from the start vertex of  $G_\varphi$ .
- ▶ How to construct  $G_\varphi$ ?

# A First Machine A

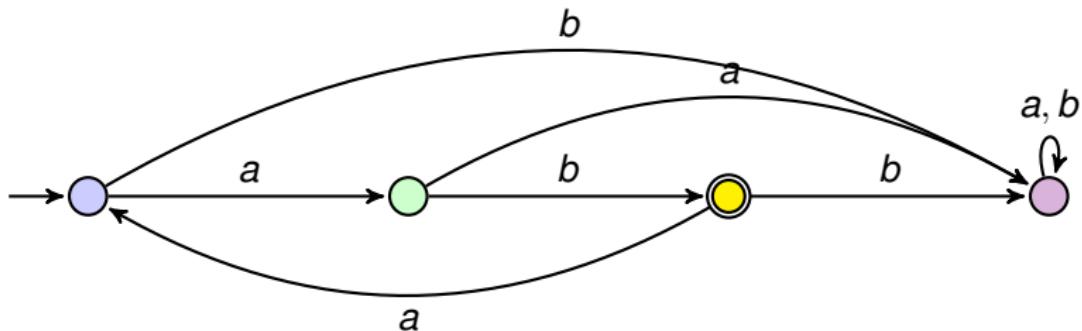
---



- ▶ Colored circles called **states**
- ▶ Arrows between circles called **transitions**
- ▶ Blue state called an **initial state**
- ▶ Doubly circled state called a **final state**

# A First Machine A

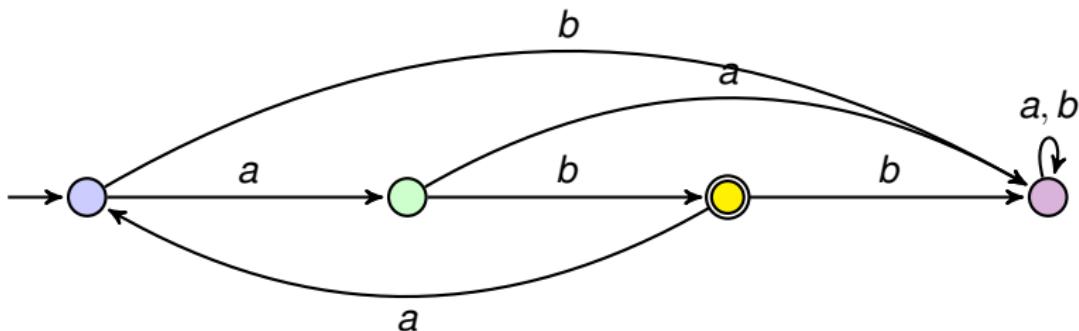
---



- ▶ A path from one state to another gives a word over  $\Sigma = \{a, b, c\}$
- ▶ The machine **accepts** words along paths from an initial state to a final state
- ▶ The set of words accepted by the machine is called the **language** accepted by the machine

# A First Machine A

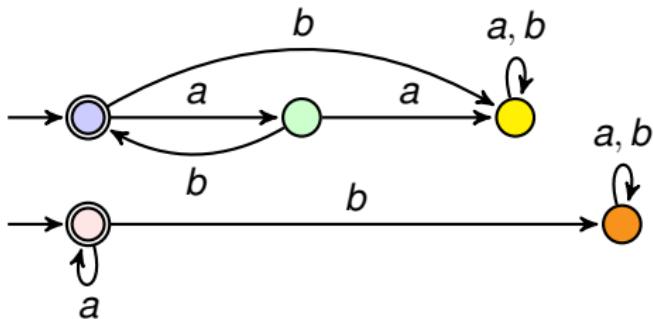
---



- ▶ What is the language  $L$  accepted by this machine,  $L(A)$ ?
- ▶ Write an FO formula  $\varphi$  such that  $L(\varphi) = L(A)$

# A Second and a Third Machine $B, C$

---



- ▶ What are  $L(B), L(C)$ ?
- ▶ Give an FO formula  $\varphi$  such that  $L(\varphi) = L(B) \cup L(C)$

# Lecture 16 FOL FSM 01.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Finite State Machines

---

A deterministic finite state automaton (DFA)  $A = (Q, \Sigma, \delta, q_0, F)$

- ▶  $Q$  is a finite set of states
- ▶  $\Sigma$  is a finite alphabet
- ▶  $\delta : Q \times \Sigma \rightarrow Q$  is the transition function
- ▶  $q_0 \in Q$  is the initial state
- ▶  $F \subseteq Q$  is the set of final states
- ▶  $L(A)$ =all words leading from  $q_0$  to some  $f \in F$

# Languages, Machines and Logic

---

A language  $L \subseteq \Sigma^*$  is called **regular** iff there exists some DFA  $A$  such that  $L = L(A)$ .

A language  $L \subseteq \Sigma^*$  is called **FO-definable** iff there exists an FO formula  $\varphi$  such that  $L = L(\varphi)$ .

# Is it Regular? Is it FO-definable?

---

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Begins with  $a$ , ends with  $b$ , and has a pair of consecutive  $a$ 's
- ▶ Contains a  $b$  and ends with  $aa$
- ▶ Contains  $abb$
- ▶ There are two occurrences of  $b$  between which only  $a$ 's occur
- ▶ Right before the last position is an  $a$
- ▶ Even length words

# Is it Regular? Is it FO-definable?

---

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

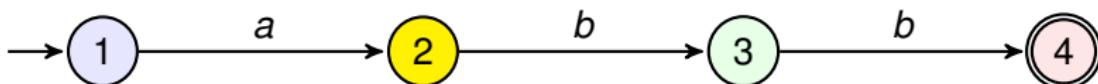
- Contains *abb*

# Is it Regular? Is it FO-definable?

---

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- Contains *abb*



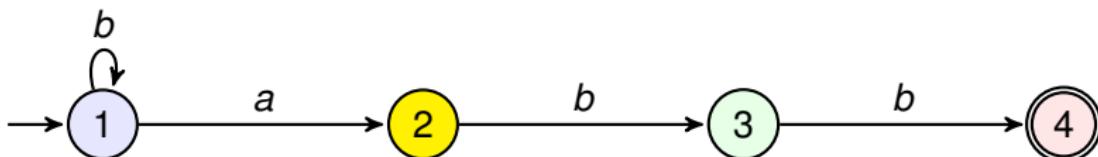
$$\exists x \exists y \exists z (Q_a(x) \wedge Q_b(y) \wedge Q_b(z) \wedge S(x, y) \wedge S(y, z))$$

# Is it Regular? Is it FO-definable?

---

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- Contains *abb*

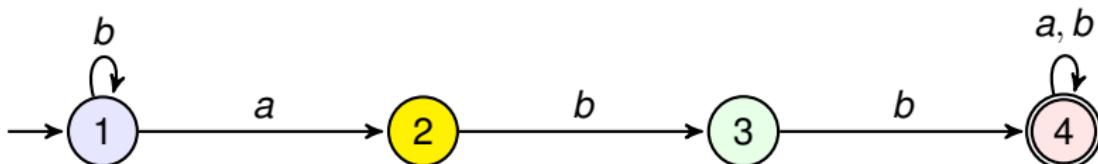


$$\exists x \exists y \exists z (Q_a(x) \wedge Q_b(y) \wedge Q_b(z) \wedge S(x, y) \wedge S(y, z))$$

# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- Contains *abb*

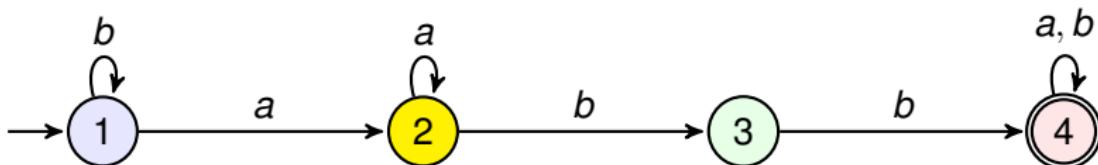


$$\exists x \exists y \exists z (Q_a(x) \wedge Q_b(y) \wedge Q_b(z) \wedge S(x, y) \wedge S(y, z))$$

# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- Contains *abb*



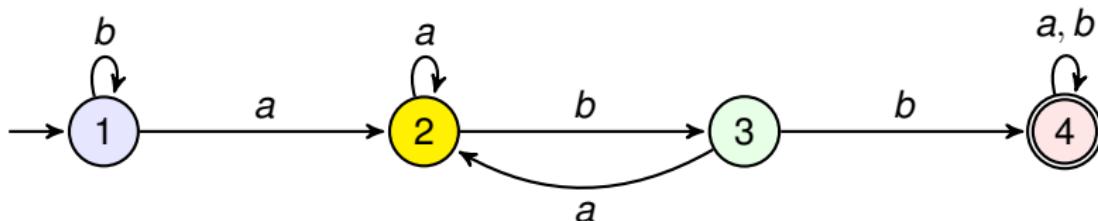
$$\exists x \exists y \exists z (Q_a(x) \wedge Q_b(y) \wedge Q_b(z) \wedge S(x, y) \wedge S(y, z))$$

# Is it Regular? Is it FO-definable?

---

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

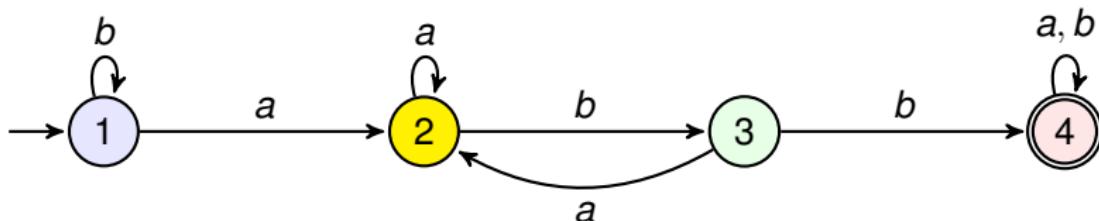
- Contains *abb*



# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- Contains *abb*



$$\exists x \exists y \exists z (Q_a(x) \wedge Q_b(y) \wedge Q_b(z) \wedge S(x, y) \wedge S(y, z))$$

# Is it Regular? Is it FO-definable?

---

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

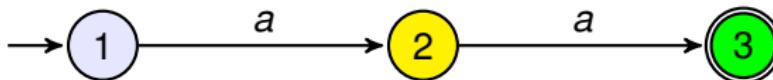
# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

Examples :  $ab, babbaa, bbab$

Non examples :  $ba, bb, aba$



$$\exists x [Q_a(x) \wedge \exists y (S(x, y) \wedge \forall z (z \leq y))]$$

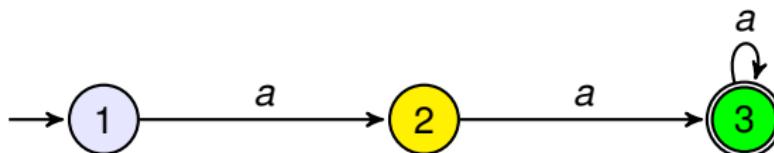
# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

Examples :  $ab, babbaa, bbab$

Non examples :  $ba, bb, aba$



$$\exists x [Q_a(x) \wedge \exists y (S(x, y) \wedge \forall z (z \leq y))]$$

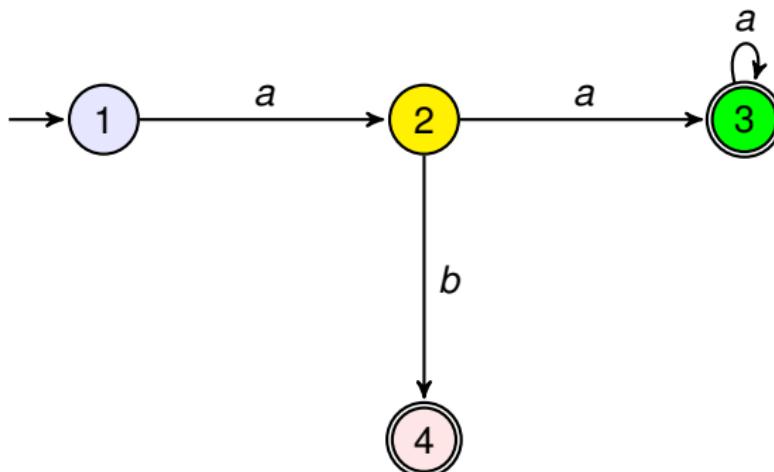
# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

Examples :  $ab, babbaa, bbab$

Non examples :  $ba, bb, aba$



$$\exists x [Q_a(x) \wedge \exists y (S(x, y) \wedge \forall z (z \leq y))]$$

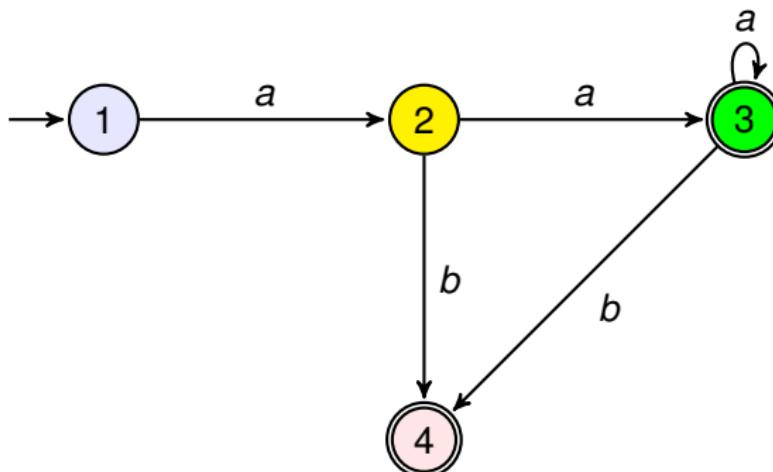
# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

Examples :  $ab, babbaa, bbab$

Non examples :  $ba, bb, aba$



$$\exists x [Q_a(x) \wedge \exists y (S(x, y) \wedge \forall z (z \leq y))]$$

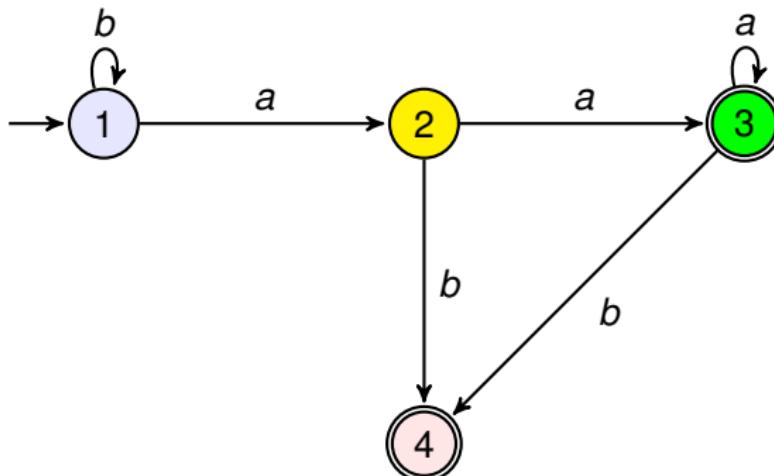
# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

Examples :  $ab, babbaa, bbab$

Non examples :  $ba, bb, aba$



$$\exists x [Q_a(x) \wedge \exists y (S(x, y) \wedge \forall z (z \leq y))]$$

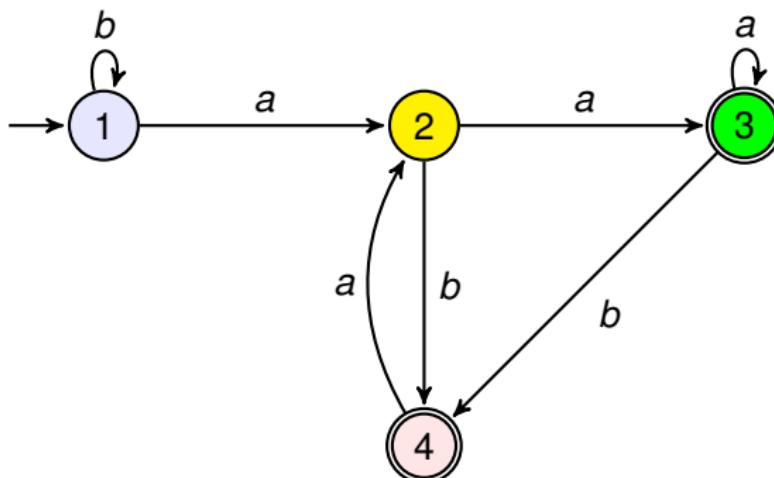
# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

Examples :  $ab, babbaa, bbab$

Non examples :  $ba, bb, aba$



$$\exists x [Q_a(x) \wedge \exists y (S(x, y) \wedge \forall z (z \leq y))]$$

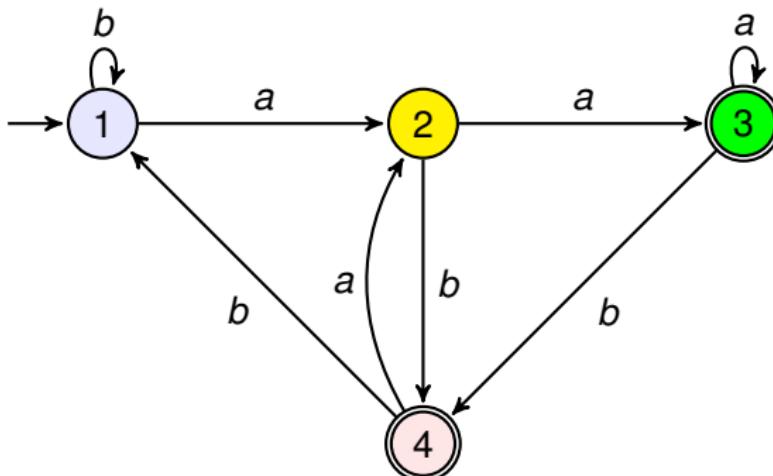
# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

Examples :  $ab, babbaa, bbab$

Non examples :  $ba, bb, aba$



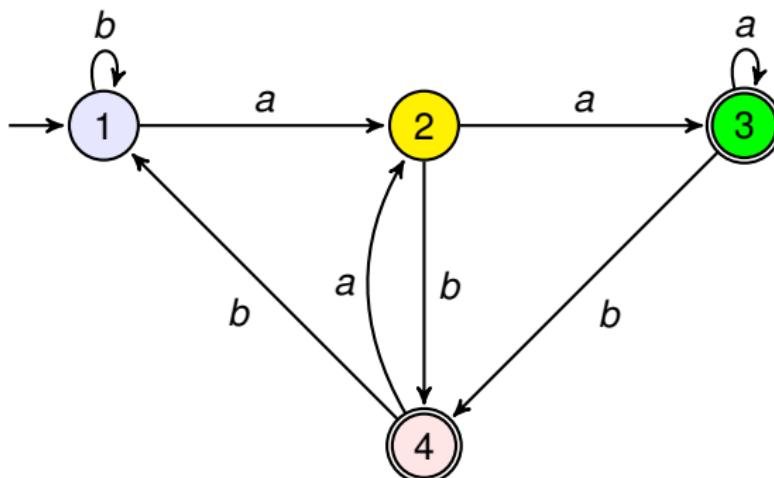
# Is it Regular? Is it FO-definable?

$\Sigma = \{a, b\}$ . Consider the following languages  $L \subseteq \Sigma^*$ :

- ▶ Right before the last position is an  $a$  :

Examples :  $ab, babbaa, bbab$

Non examples :  $ba, bb, aba$



$$\exists x [Q_a(x) \wedge \exists y (S(x, y) \wedge \forall z (z \leq y))]$$

# Deterministic Finite Automata

---

- ▶ Every state on every symbol goes to a unique state
  - ▶  $\delta : Q \times \Sigma \rightarrow Q$  is a transition function
- ▶ Given a string  $w \in \Sigma^*$  and a state  $q \in Q$ , iteratively apply  $\delta$ 
  - ▶  $w = aab$
  - ▶  $\delta(q, a) = q_1,$

# Deterministic Finite Automata

---

- ▶ Every state on every symbol goes to a unique state
  - ▶  $\delta : Q \times \Sigma \rightarrow Q$  is a transition function
- ▶ Given a string  $w \in \Sigma^*$  and a state  $q \in Q$ , iteratively apply  $\delta$ 
  - ▶  $w = aab$
  - ▶  $\delta(q, a) = q_1, \delta(\delta(q, a), a) = \delta(q_1, a) = q_2,$

# Deterministic Finite Automata

---

- ▶ Every state on every symbol goes to a unique state
  - ▶  $\delta : Q \times \Sigma \rightarrow Q$  is a transition function
- ▶ Given a string  $w \in \Sigma^*$  and a state  $q \in Q$ , iteratively apply  $\delta$ 
  - ▶  $w = aab$
  - ▶  $\delta(q, a) = q_1, \delta(\delta(q, a), a) = \delta(q_1, a) = q_2,$   
 $\delta(\delta(\delta(q, a), a), b) =$

# Deterministic Finite Automata

---

- ▶ Every state on every symbol goes to a unique state
  - ▶  $\delta : Q \times \Sigma \rightarrow Q$  is a transition function
- ▶ Given a string  $w \in \Sigma^*$  and a state  $q \in Q$ , iteratively apply  $\delta$ 
  - ▶  $w = aab$
  - ▶  $\delta(q, a) = q_1, \delta(\delta(q, a), a) = \delta(q_1, a) = q_2,$   
 $\delta(\delta(\delta(q, a), a), b) = \delta(\delta(q_1, a), b) =$

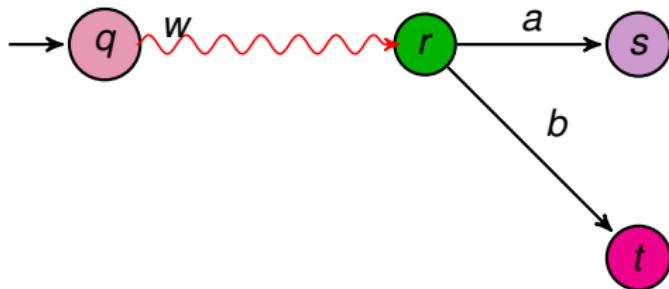
# Deterministic Finite Automata

---

- ▶ Every state on every symbol goes to a unique state
  - ▶  $\delta : Q \times \Sigma \rightarrow Q$  is a transition function
- ▶ Given a string  $w \in \Sigma^*$  and a state  $q \in Q$ , iteratively apply  $\delta$ 
  - ▶  $w = aab$
  - ▶  $\delta(q, a) = q_1, \delta(\delta(q, a), a) = \delta(q_1, a) = q_2,$   
 $\delta(\delta(\delta(q, a), a), b) = \delta(\delta(q_1, a), b) = \delta(q_2, b) = q_3$
  - ▶  $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$  extension of  $\delta$  to strings
    - ▶  $\hat{\delta}(q, \epsilon) = q$
    - ▶  $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$

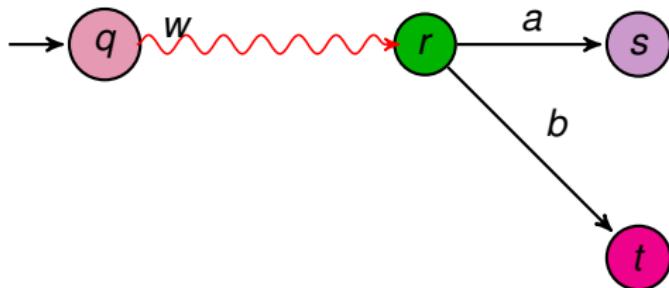
# DFA : Transition Function on Words

---



# DFA : Transition Function on Words

---



- ▶  $\hat{\delta}(q, wa) = s = \delta(\hat{\delta}(q, w), a) = \delta(r, a)$
- ▶  $\hat{\delta}(q, wb) = t = \delta(\hat{\delta}(q, w), b) = \delta(r, b)$

# DFA Acceptance

---

- ▶  $w \in \Sigma^*$  is accepted iff  $\hat{\delta}(q_0, w) \in F$
- ▶  $w \in \Sigma^*$  is rejected iff  $\hat{\delta}(q_0, w) \notin F$
- ▶ Any string  $w \in \Sigma^*$  is either accepted or rejected by a DFA  $A$
- ▶  $L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$
- ▶  $\Sigma^* = L(A) \cup \overline{L(A)}$

# DFA States

---

- ▶ Each state is a **bucket** holding infinitely many words
- ▶ Thus we have good and bad buckets
- ▶ The buckets partition  $\Sigma^*$
- ▶ **Good buckets** determine the language accepted by the DFA
- ▶ Words that land in bad buckets are not accepted by the DFA

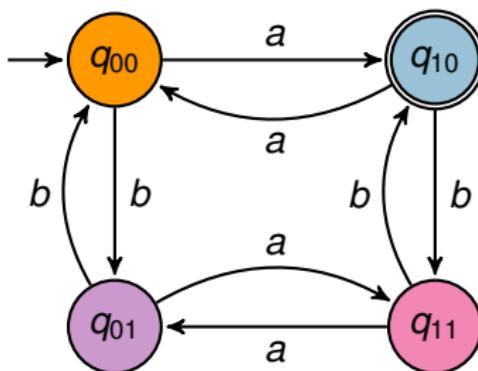
# Lecture 17 FOL FSM 02.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Language Acceptance : Proof

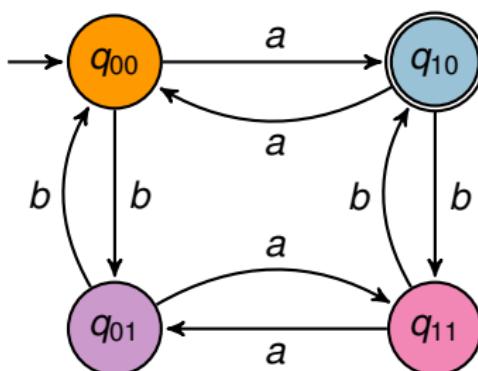
---



- ▶ Prove by induction on  $|w|$

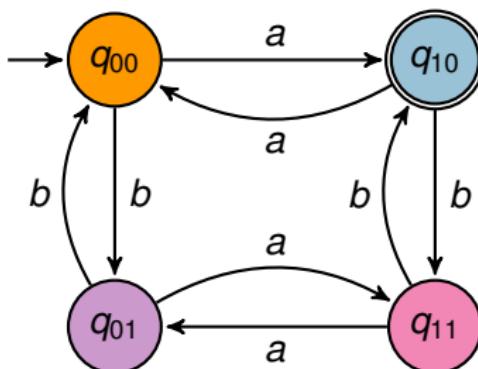
# Language Acceptance : Proof

---



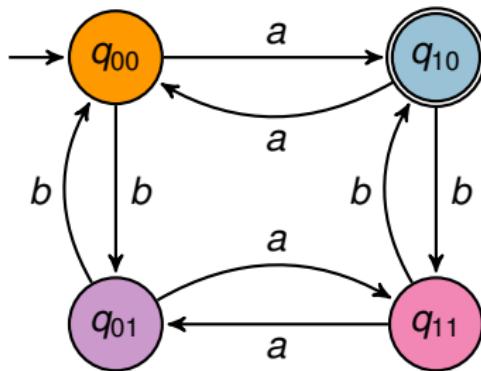
- ▶ Prove by induction on  $|w|$
- ▶ Base case : For  $|w| = \epsilon$ ,  $\hat{\delta}(q_{00}, \epsilon) = q_{00}$

# Language Acceptance : Proof



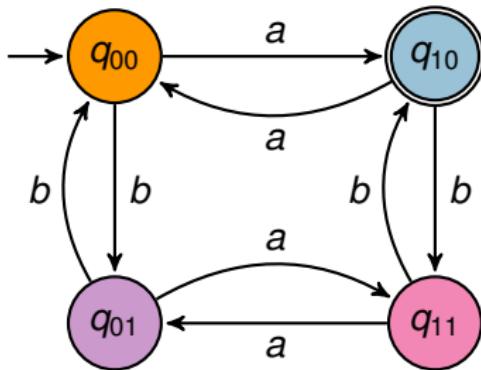
- ▶ Prove by induction on  $|w|$
- ▶ Base case : For  $|w| = \epsilon$ ,  $\hat{\delta}(q_{00}, \epsilon) = q_{00}$
- ▶ Assume the claim for  $x \in \Sigma^*$ , and show it for  $xc, c \in \{a, b\}$ .

# Language Acceptance : Proof



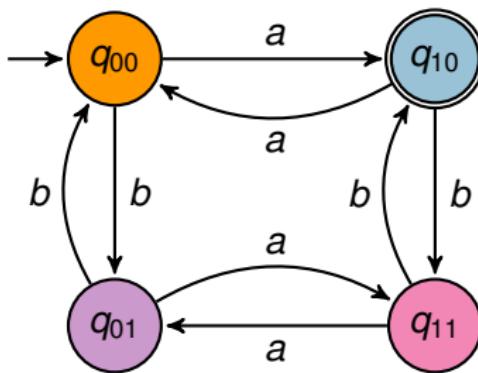
- ▶  $\hat{\delta}(q_{00}, xc) = \delta(\hat{\delta}(q_{00}, x), c)$

# Language Acceptance : Proof



- ▶  $\hat{\delta}(q_{00}, xc) = \delta(\hat{\delta}(q_{00}, x), c)$
- ▶ By induction hypothesis,  $\hat{\delta}(q_{00}, x) = q_{ij}$  iff
  - ▶ parity of  $i$  and  $|x|_a$  are the same
  - ▶ parity of  $j$  and  $|x|_b$  are the same

# Language Acceptance : Proof



- ▶ Case Analysis : If  $|x|_a$  odd and  $|x|_b$  even, then  $i = 1, j = 0$ 
  - ▶  $\delta(q_{10}, a) = q_{00}$ ,  $\delta(q_{10}, b) = q_{11}$
  - ▶  $|xa|_a$  is even and  $|xa|_b$  is even
  - ▶  $|xb|_a$  is odd and  $|xb|_b$  is odd
- ▶ Other Cases : Similar
- ▶  $\hat{\delta}(q_{00}, x) = q_{10}$  iff  $|x|_a$  odd and  $|x|_b$  even

## Closure Properties : DFA

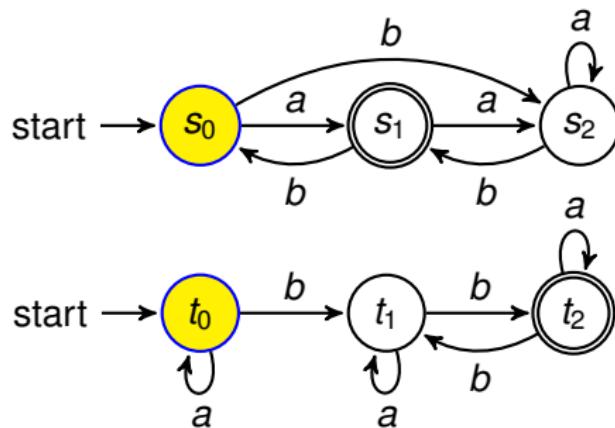
# Closure under Complementation

---

- ▶ If  $L$  is regular, so is  $\bar{L}$ 
  - ▶ Let  $A = (Q, q_0, \Sigma, \delta, F)$  be the DFA such that  $L = L(A)$
  - ▶ For every  $w \in L$ ,  $\hat{\delta}(q_0, w) = f$  for some  $f \in F$
  - ▶ For every  $w \notin L$ ,  $\hat{\delta}(q_0, w) = q$  for some  $q \notin F$
  - ▶ Construct  $\bar{A} = (Q, q_0, \Sigma, \delta, Q - F)$ 
    - ▶  $w \in L(\bar{A})$  iff  $\hat{\delta}(q_0, w) \in Q - F$  iff  $w \notin L(A)$
    - ▶  $L(\bar{A}) = \overline{L(A)}$

# Closure under Intersection

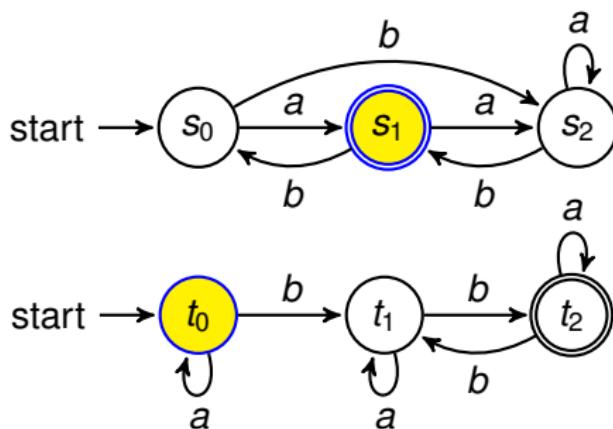
▶  $aaab$



# Closure under Intersection

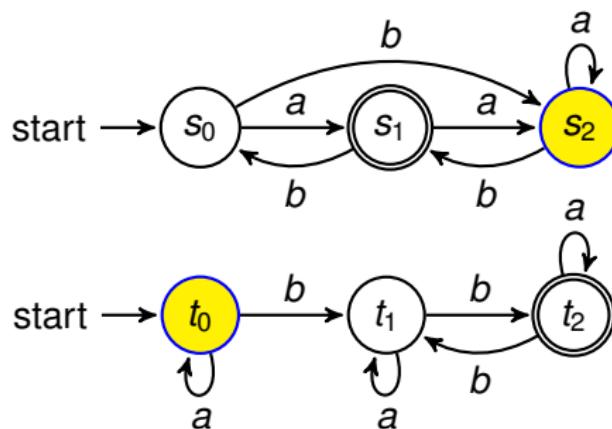
---

►  $aabb$



# Closure under Intersection

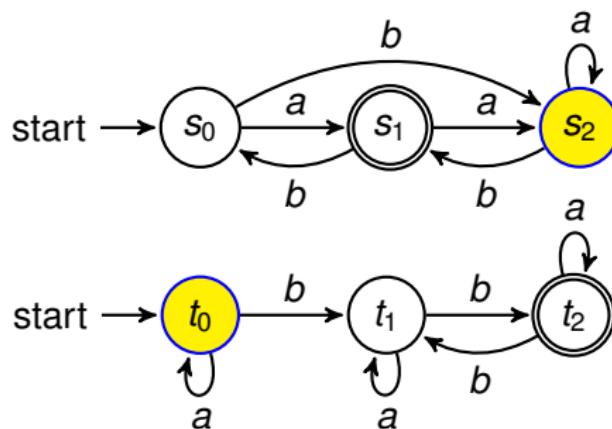
- ▶  $a \textcolor{red}{a}ab$



# Closure under Intersection

---

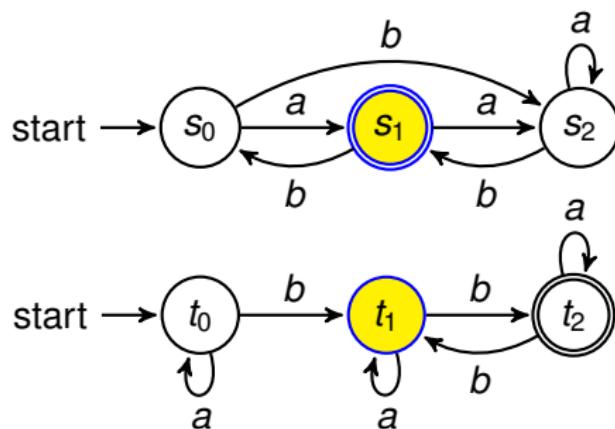
►  $aa\textcolor{red}{ab}$



# Closure under Intersection

---

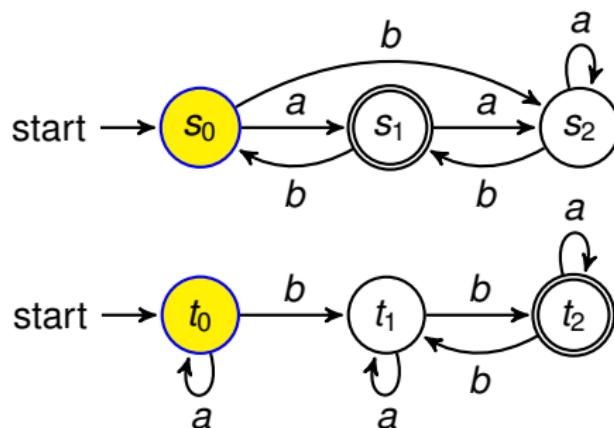
►  $aaa\text{b}$



# Closure under Intersection

---

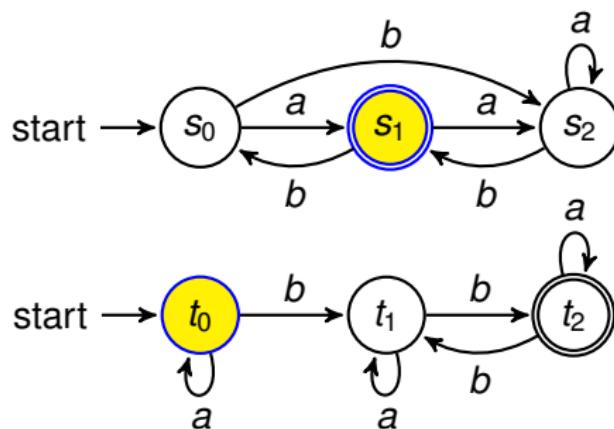
►  $aabba$



# Closure under Intersection

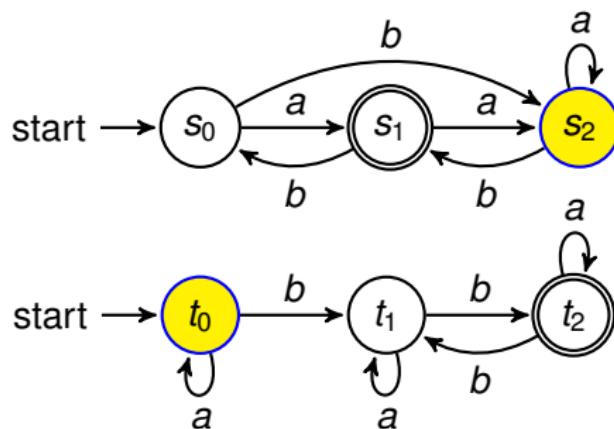
---

►  $aabba$



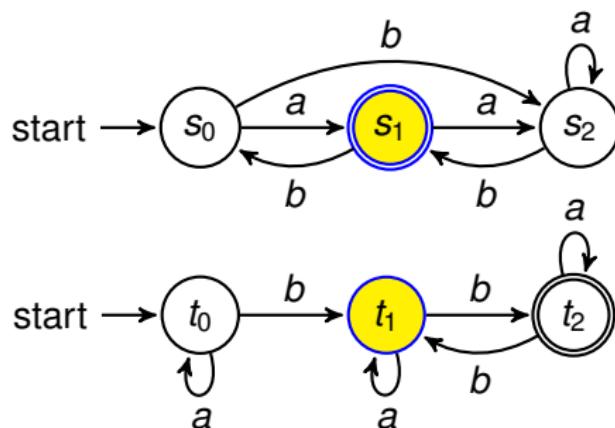
# Closure under Intersection

- ▶  $a\textcolor{red}{abba}$



# Closure under Intersection

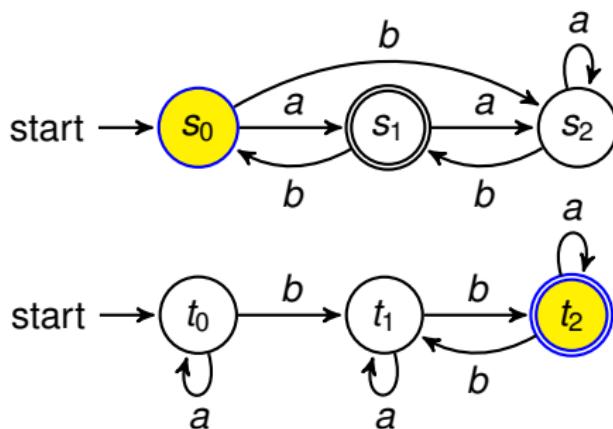
►  $aabbba$



# Closure under Intersection

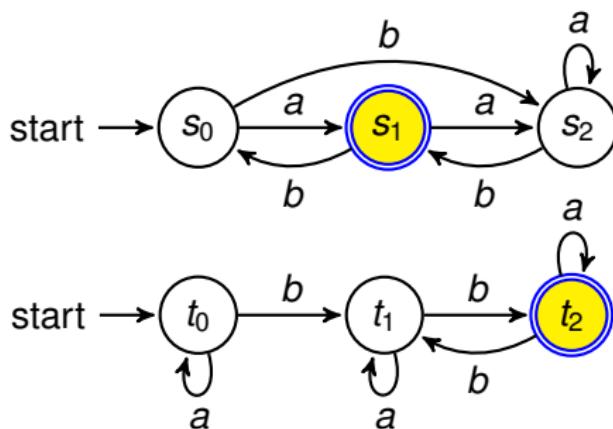
---

►  $aab\textcolor{red}{ba}$



# Closure under Intersection

►  $aabbba$



# Closure under Intersection

---

- ▶  $A_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$
- ▶  $A_2 = (Q_2, \Sigma, \delta_2, s_0, F_2)$
- ▶  $A = (Q_1 \times Q_2, \Sigma, \delta, (q_0, s_0), F)$ ,
  - ▶  $\delta((q, s), a) = (\delta_1(q, a), \delta_2(s, a))$
  - ▶  $F = F_1 \times F_2$

# Closure under Intersection

---

- ▶  $A_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$
- ▶  $A_2 = (Q_2, \Sigma, \delta_2, s_0, F_2)$
- ▶  $A = (Q_1 \times Q_2, \Sigma, \delta, (q_0, s_0), F)$ ,
  - ▶  $\delta((q, s), a) = (\delta_1(q, a), \delta_2(s, a))$
  - ▶  $F = F_1 \times F_2$
- ▶ Show that for all  $x \in \Sigma^*$ ,  $\hat{\delta}((p, q), x) = (\hat{\delta}_1(p, x), \hat{\delta}_2(q, x))$
- ▶  $x \in L(A)$  iff  $\hat{\delta}((q_0, s_0), x) \in F$

# Closure under Intersection

---

- ▶  $A_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$
  - ▶  $A_2 = (Q_2, \Sigma, \delta_2, s_0, F_2)$
  - ▶  $A = (Q_1 \times Q_2, \Sigma, \delta, (q_0, s_0), F)$ ,
    - ▶  $\delta((q, s), a) = (\delta_1(q, a), \delta_2(s, a))$
    - ▶  $F = F_1 \times F_2$
  - ▶ Show that for all  $x \in \Sigma^*$ ,  $\hat{\delta}((p, q), x) = (\hat{\delta}_1(p, x), \hat{\delta}_2(q, x))$
- $x \in L(A)$  iff  $\hat{\delta}((q_0, s_0), x) \in F$  iff  $(\hat{\delta}_1(q_0, x), \hat{\delta}_2(s_0, x)) \in F_1 \times F_2$

# Closure under Intersection

---

- ▶  $A_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$
- ▶  $A_2 = (Q_2, \Sigma, \delta_2, s_0, F_2)$
- ▶  $A = (Q_1 \times Q_2, \Sigma, \delta, (q_0, s_0), F)$ ,
  - ▶  $\delta((q, s), a) = (\delta_1(q, a), \delta_2(s, a))$
  - ▶  $F = F_1 \times F_2$
- ▶ Show that for all  $x \in \Sigma^*$ ,  $\hat{\delta}((p, q), x) = (\hat{\delta}_1(p, x), \hat{\delta}_2(q, x))$

$x \in L(A)$  iff  $\hat{\delta}((q_0, s_0), x) \in F$  iff  $(\hat{\delta}_1(q_0, x), \hat{\delta}_2(s_0, x)) \in F_1 \times F_2$  iff  
 $\hat{\delta}_1(q_0, x) \in F_1$  and  $\hat{\delta}_2(s_0, x) \in F_2$

# Closure under Intersection

---

- ▶  $A_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$
- ▶  $A_2 = (Q_2, \Sigma, \delta_2, s_0, F_2)$
- ▶  $A = (Q_1 \times Q_2, \Sigma, \delta, (q_0, s_0), F)$ ,
  - ▶  $\delta((q, s), a) = (\delta_1(q, a), \delta_2(s, a))$
  - ▶  $F = F_1 \times F_2$
- ▶ Show that for all  $x \in \Sigma^*$ ,  $\hat{\delta}((p, q), x) = (\hat{\delta}_1(p, x), \hat{\delta}_2(q, x))$

$x \in L(A)$  iff  $\hat{\delta}((q_0, s_0), x) \in F$  iff  $(\hat{\delta}_1(q_0, x), \hat{\delta}_2(s_0, x)) \in F_1 \times F_2$  iff  
 $\hat{\delta}_1(q_0, x) \in F_1$  and  $\hat{\delta}_2(s_0, x) \in F_2$  iff  $x \in L(A_1)$  and  $x \in L(A_2)$

# Closure under Union

---

- ▶  $A_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$
- ▶  $A_2 = (Q_2, \Sigma, \delta_2, s_0, F_2)$
- ▶  $A = (Q_1 \times Q_2, \Sigma, \delta, (q_0, s_0), F)$ ,
  - ▶  $\delta((q, s), a) = (\delta_1(q, a), \delta_2(s, a))$

# Closure under Union

---

- ▶  $A_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$
- ▶  $A_2 = (Q_2, \Sigma, \delta_2, s_0, F_2)$
- ▶  $A = (Q_1 \times Q_2, \Sigma, \delta, (q_0, s_0), F)$ ,
  - ▶  $\delta((q, s), a) = (\delta_1(q, a), \delta_2(s, a))$
  - ▶  $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- ▶ Show that for all  $x \in \Sigma^*$ ,  $\hat{\delta}((p, q), x) = (\hat{\delta}_1(p, x), \hat{\delta}_2(q, x))$

$x \in L(A)$  iff  $x \in L(A_1)$  or  $x \in L(A_2)$

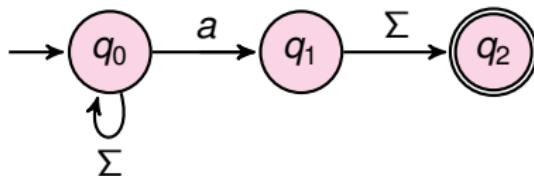
# Moving on to Non-determinism

---

- ▶ We looked at DFA
- ▶ Showed closure under union, intersection and complementation
- ▶ Before we examine closure under concatenation, we look at a more relaxed model, which is as good as a DFA

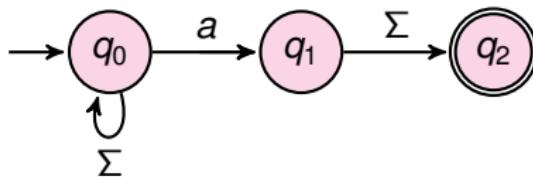
# The Comfort of Non-determinism

---



# The Comfort of Non-determinism

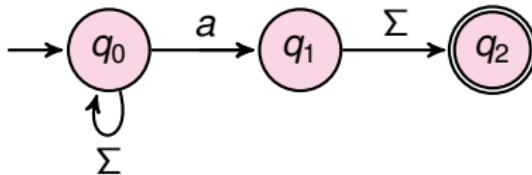
---



- ▶ Assume we relax the condition on transitions, and allow
  - ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$
  - ▶  $\delta(q_0, a) = \{q_0, q_1\}, \delta(q_2, a) = \emptyset$

# The Comfort of Non-determinism

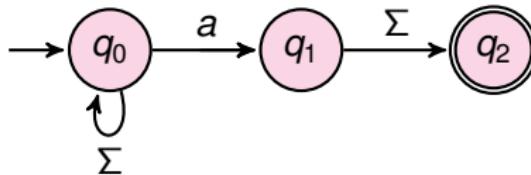
---



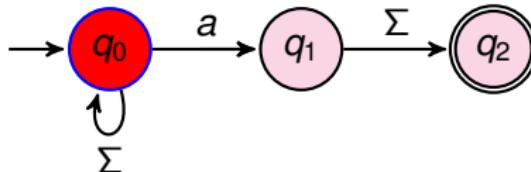
- ▶ Assume we relax the condition on transitions, and allow
  - ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$
  - ▶  $\delta(q_0, a) = \{q_0, q_1\}, \delta(q_2, a) = \emptyset$
  - ▶ Is  $aabb$  accepted?

# The Comfort of Non-determinism

---



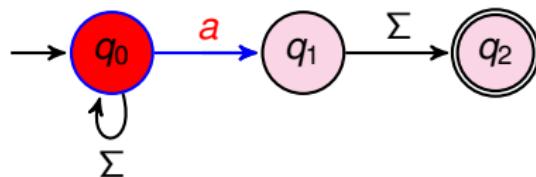
- ▶ Assume we relax the condition on transitions, and allow
  - ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$
  - ▶  $\delta(q_0, a) = \{q_0, q_1\}, \delta(q_2, a) = \emptyset$
  - ▶ Is  $aabb$  accepted?



# One run of $aabb$

---

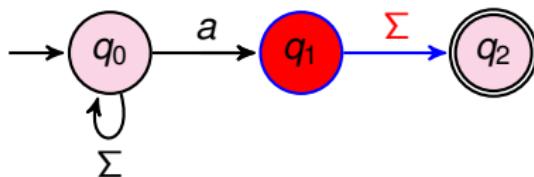
Is  $aabb$  accepted?



# One run of $aabb$

---

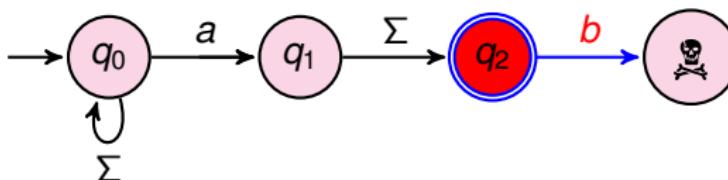
Is  $aabb$  accepted?



# One run of $aabb$

---

Is  $aabb$  accepted?

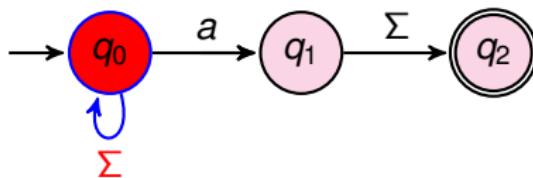


- ▶ A non-accepting run for  $aabb$

# Another run of $aabb$

---

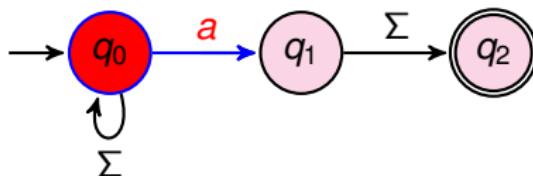
Is  $aabb$  accepted?



# Another run of $aabb$

---

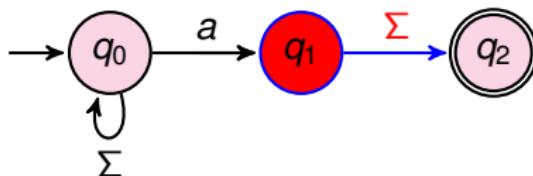
Is  $aabb$  accepted?



# Another run of $aabb$

---

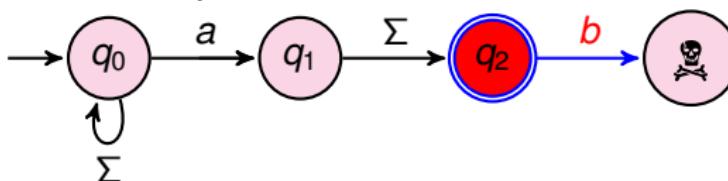
Is  $aabb$  accepted?



# Another run of $aabb$

---

Is  $aabb$  accepted?

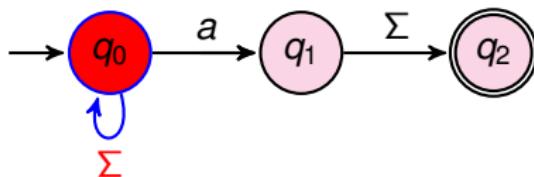


- ▶ A non-accepting run for  $aabb$

# A run of $aaab$

---

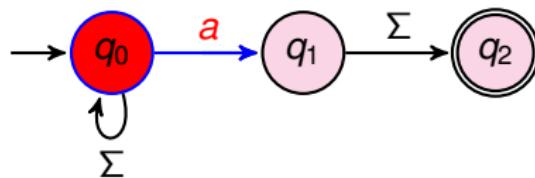
Is  $aaab$  accepted?



# A run of $aaab$

---

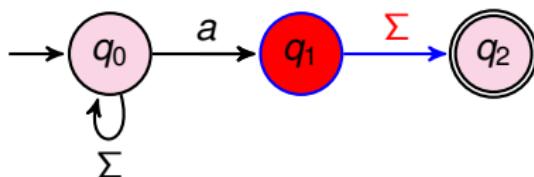
Is  $aaab$  accepted?



# A run of $aaab$

---

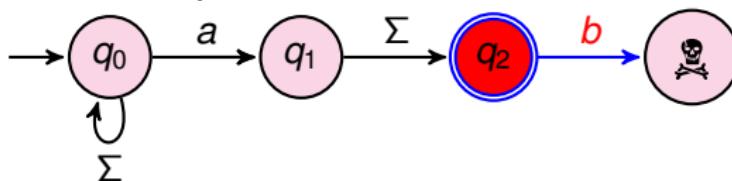
Is  $aaab$  accepted?



# A run of $aaab$

---

Is  $aaab$  accepted?

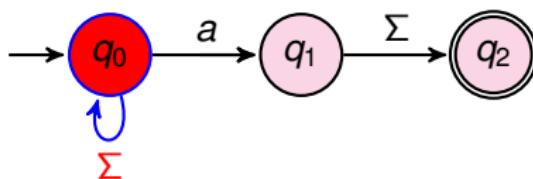


- ▶ A non-accepting run for  $aaab$

# Another run of $aaab$

---

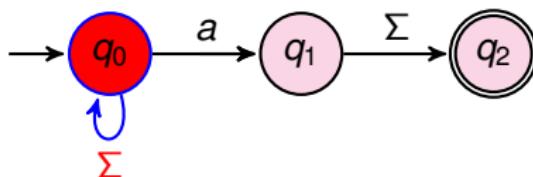
Is  $aaab$  accepted?



# Another run of $aaab$

---

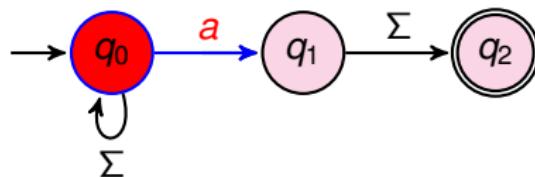
Is  $aaab$  accepted?



# Another run of $aaab$

---

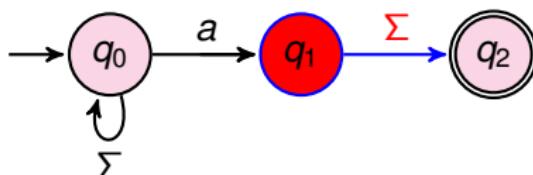
Is  $aaab$  accepted?



# Another run of $aaab$

---

Is  $aaab$  accepted?



- ▶ An accepting run for  $aaab$

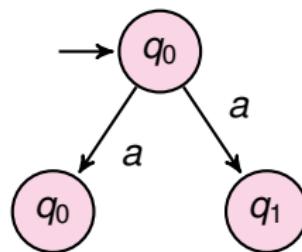
# Nondeterministic Finite Automata(NFA)

---

- ▶  $N = (Q, \Sigma, \delta, Q_0, F)$ 
  - ▶  $Q$  is a finite set of states
  - ▶  $Q_0 \subseteq Q$  is the set of initial states
  - ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$  is the transition function
  - ▶  $F \subseteq Q$  is the set of final states
- ▶ Acceptance condition : A word  $w$  is accepted iff it has atleast one accepting path

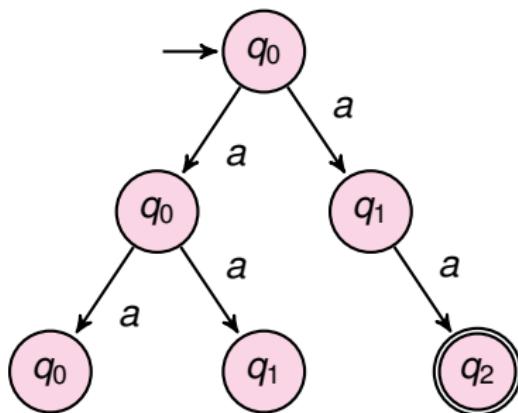
# Run Tree of aaab

---



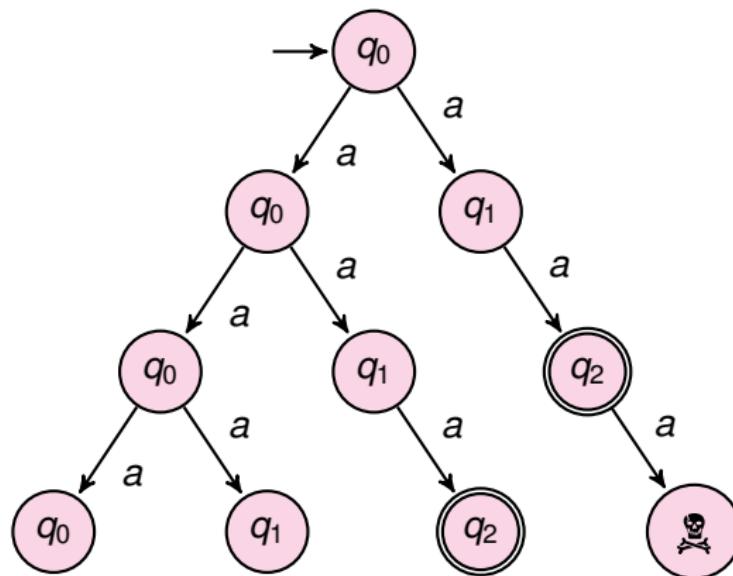
# Run Tree of aaab

---



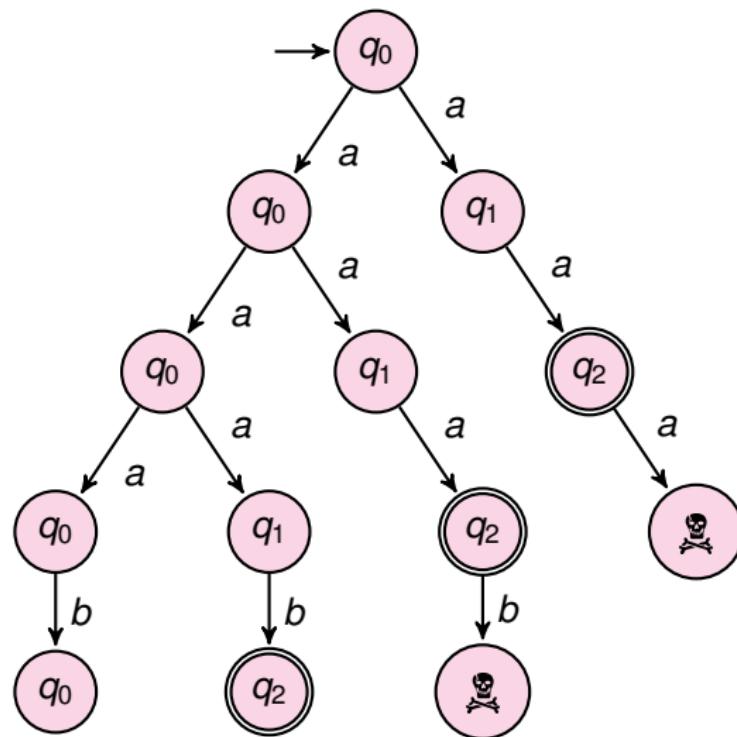
# Run Tree of aaab

---



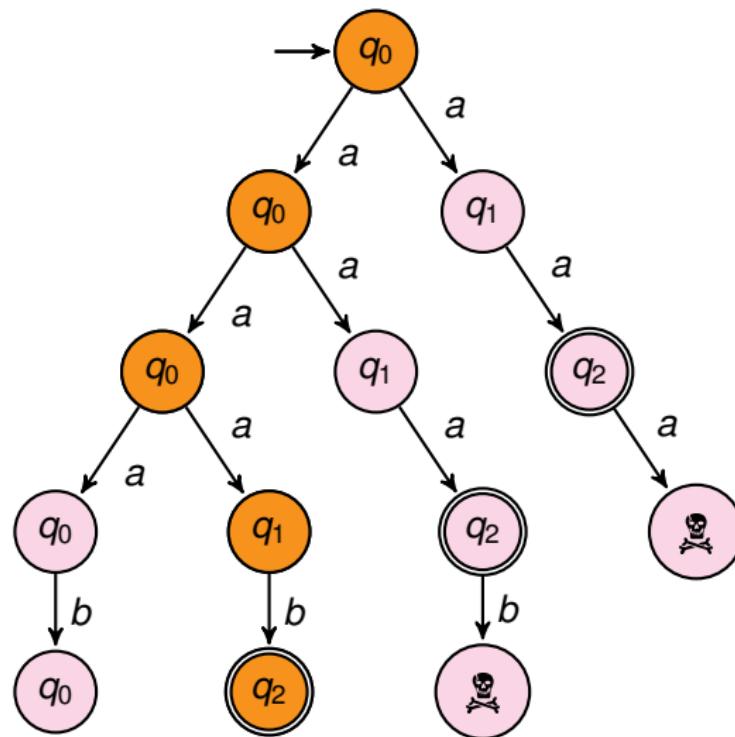
# Run Tree of aaab

---



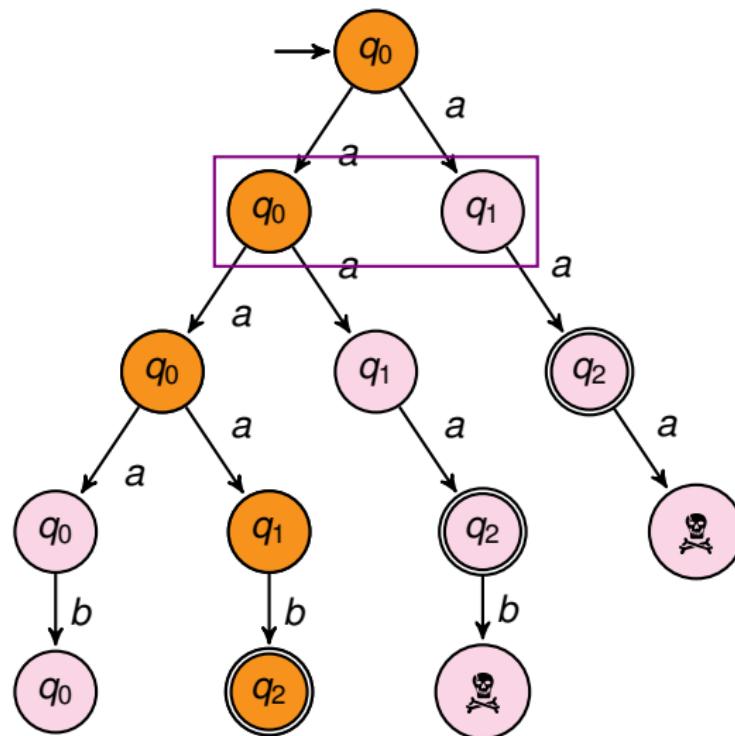
# Run Tree of aaab

---

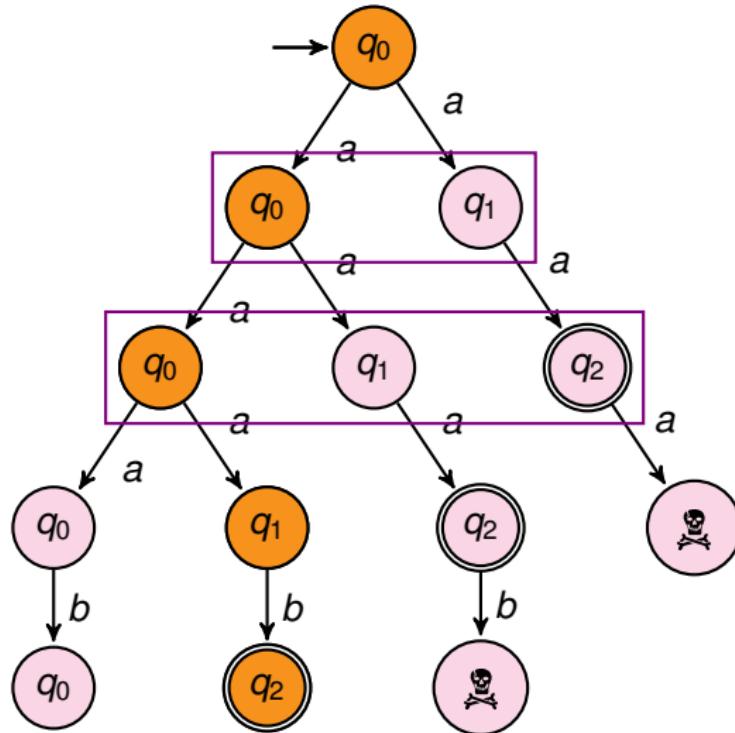


# Run Tree of aaab

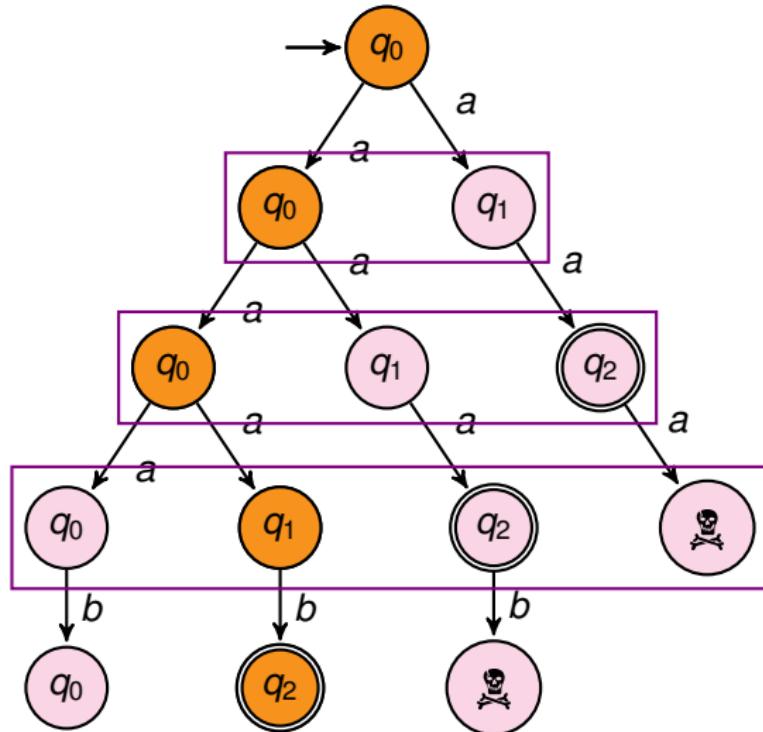
---



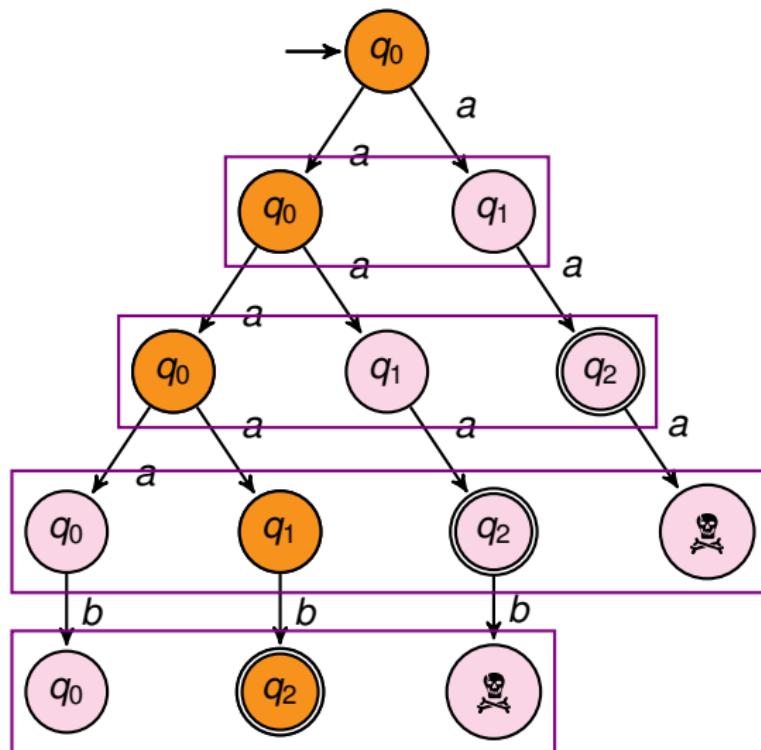
# Run Tree of aaab



# Run Tree of aaab

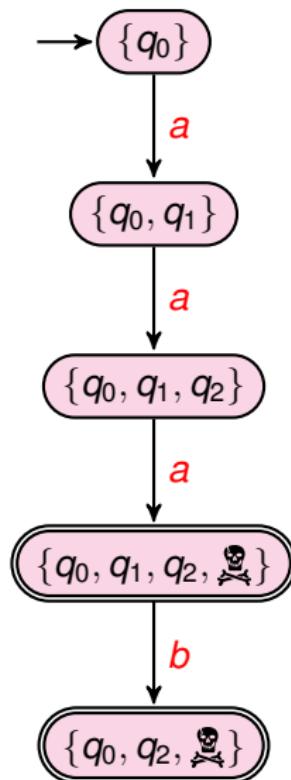


# Run Tree of *aaab*



# The Single Run

---



# Lecture 18 FOL NFA.pdf

# CS 228 : Logic in Computer Science

Krishna. S

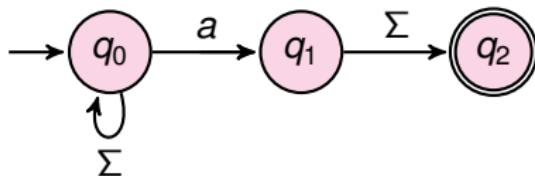
# Recap

---

- ▶ FOL over words : Satisfiability
- ▶ Translation from formulae  $\varphi$  to equivalent DFA  $A$ ,  $L(\varphi) = L(A)$
- ▶ Proof by structural induction, with  $\neg, \wedge, \vee$  mapping to complementation, intersection and union
  - ▶ Union in DFA  $\rightarrow$  disjunction in logic
  - ▶ Intersection in DFA  $\rightarrow$  conjunction in logic
  - ▶ Complementation in DFA  $\rightarrow$  Negation in logic
- ▶ How to handle quantifiers?

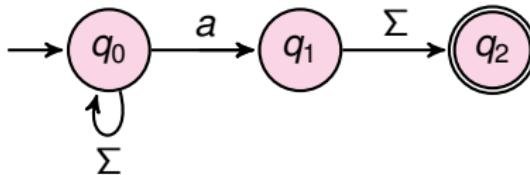
# Non-determinism

---



# Non-determinism

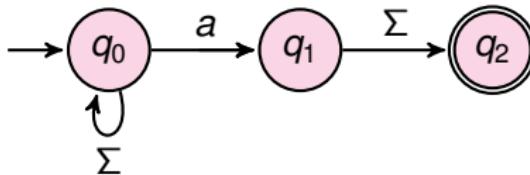
---



- ▶ Assume we relax the condition on transitions, and allow
  - ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$
  - ▶  $\delta(q_0, a) = \{q_0, q_1\}, \delta(q_2, a) = \emptyset$

# Non-determinism

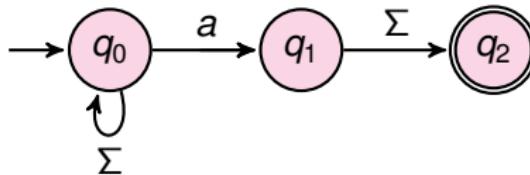
---



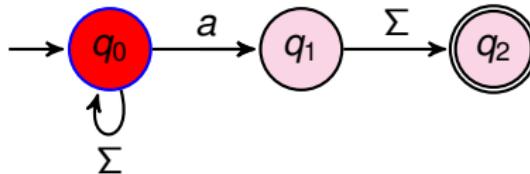
- ▶ Assume we relax the condition on transitions, and allow
  - ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$
  - ▶  $\delta(q_0, a) = \{q_0, q_1\}, \delta(q_2, a) = \emptyset$
  - ▶ Is  $aabb$  accepted?

# Non-determinism

---



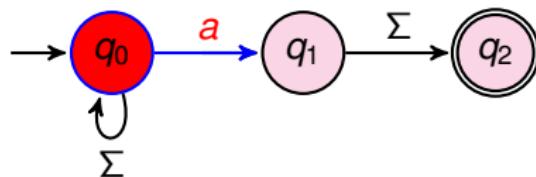
- ▶ Assume we relax the condition on transitions, and allow
  - ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$
  - ▶  $\delta(q_0, a) = \{q_0, q_1\}, \delta(q_2, a) = \emptyset$
  - ▶ Is  $aabb$  accepted?



# One run of $aabb$

---

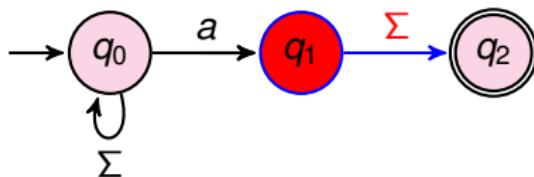
Is  $aabb$  accepted?



# One run of $aabb$

---

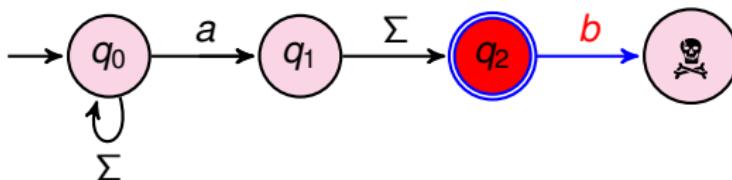
Is  $aabb$  accepted?



# One run of $aabb$

---

Is  $aabb$  accepted?

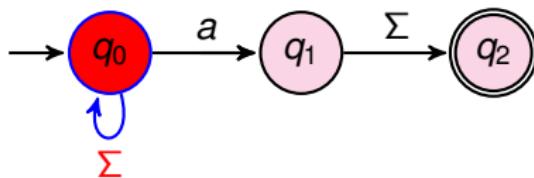


- ▶ A non-accepting run for  $aabb$

# Another run of $aabb$

---

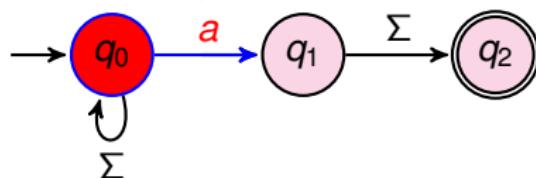
Is  $aabb$  accepted?



# Another run of $aabb$

---

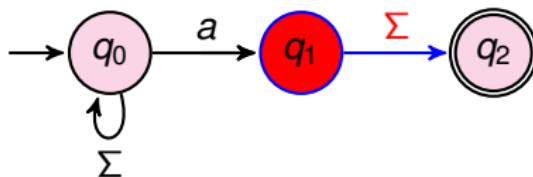
Is  $aabb$  accepted?



# Another run of $aabb$

---

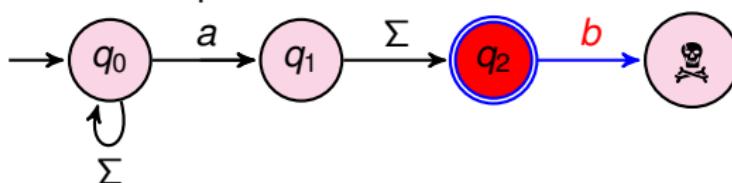
Is  $aabb$  accepted?



# Another run of $aabb$

---

Is  $aabb$  accepted?

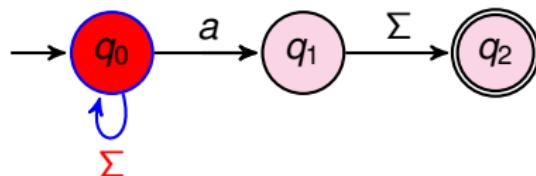


- ▶ A non-accepting run for  $aabb$

# A run of $aaab$

---

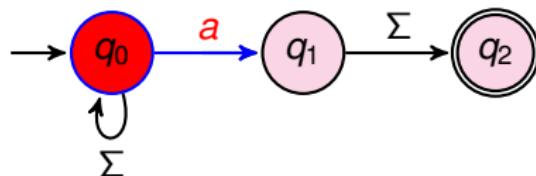
Is  $aaab$  accepted?



# A run of $aaab$

---

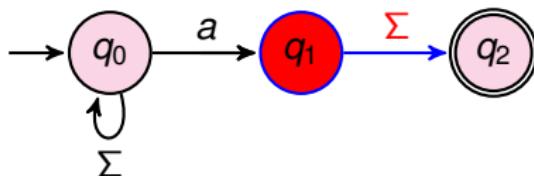
Is  $aaab$  accepted?



# A run of $aaab$

---

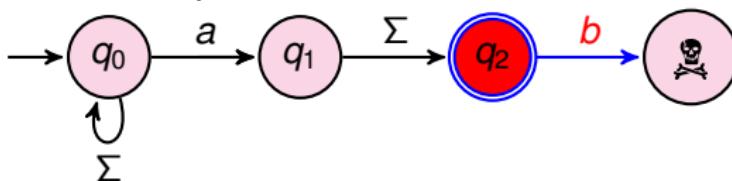
Is  $aaab$  accepted?



# A run of $aaab$

---

Is  $aaab$  accepted?

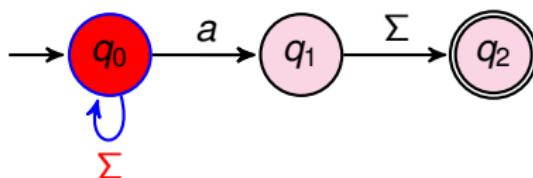


- ▶ A non-accepting run for  $aaab$

# Another run of $aaab$

---

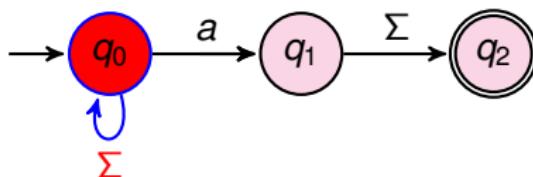
Is  $aaab$  accepted?



# Another run of $aaab$

---

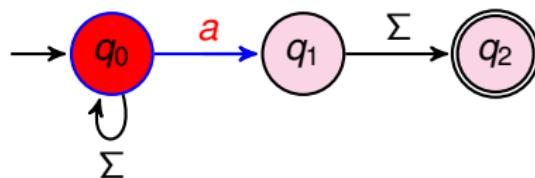
Is  $aaab$  accepted?



# Another run of $aaab$

---

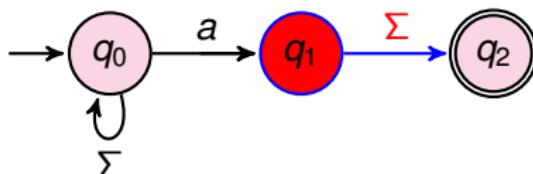
Is  $aaab$  accepted?



# Another run of $aaab$

---

Is  $aaab$  accepted?



- ▶ An accepting run for  $aaab$

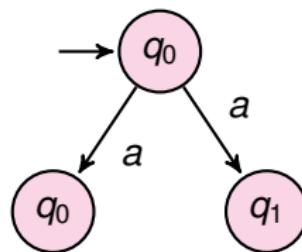
# Nondeterministic Finite Automata(NFA)

---

- ▶  $N = (Q, \Sigma, \delta, Q_0, F)$ 
  - ▶  $Q$  is a finite set of states
  - ▶  $Q_0 \subseteq Q$  is the set of initial states
  - ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$  is the transition function
  - ▶  $F \subseteq Q$  is the set of final states
- ▶ Acceptance condition : A word  $w$  is accepted iff it has atleast one accepting path

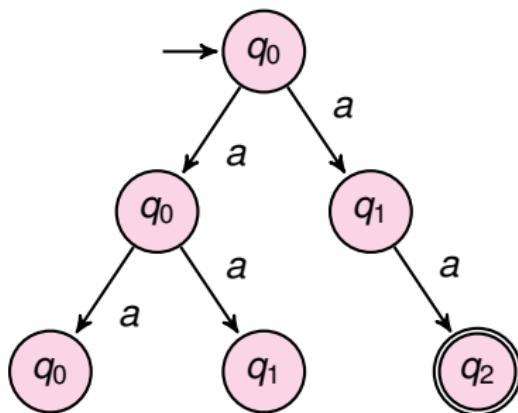
# Run Tree of aaab

---



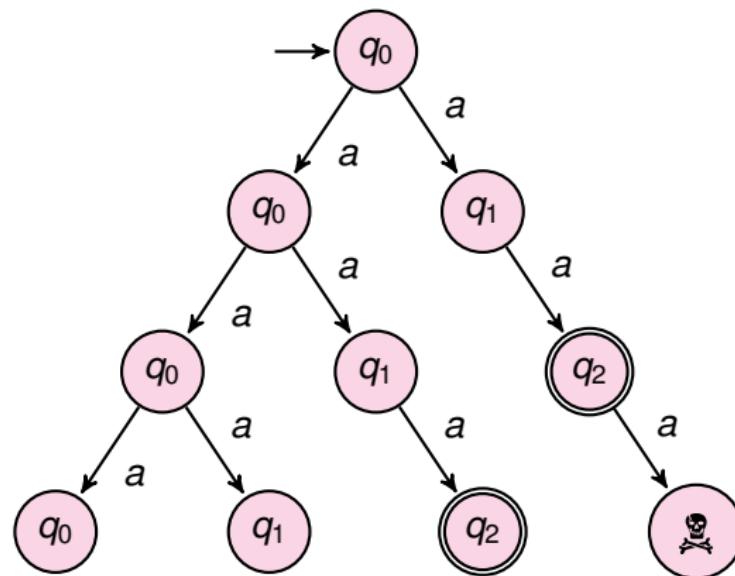
# Run Tree of aaab

---



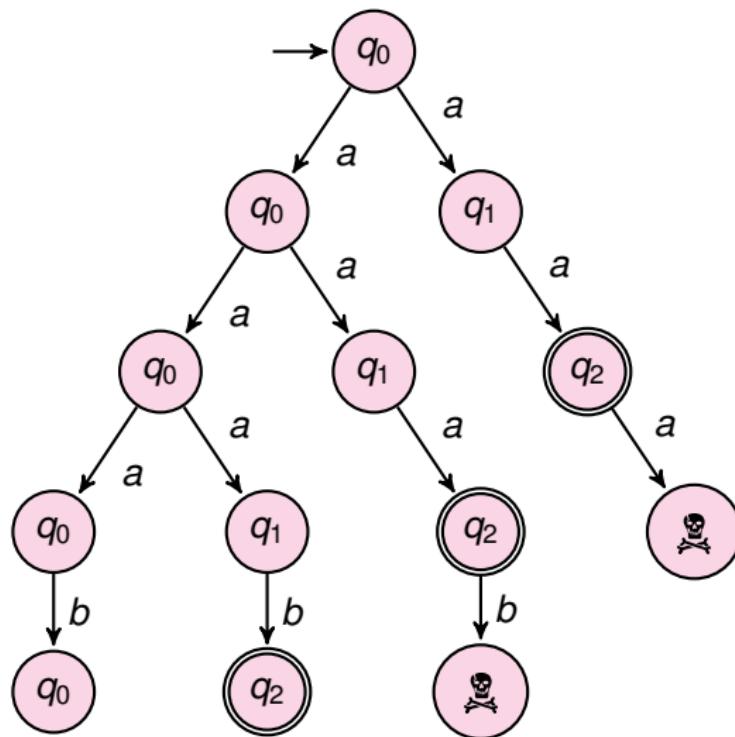
# Run Tree of aaab

---



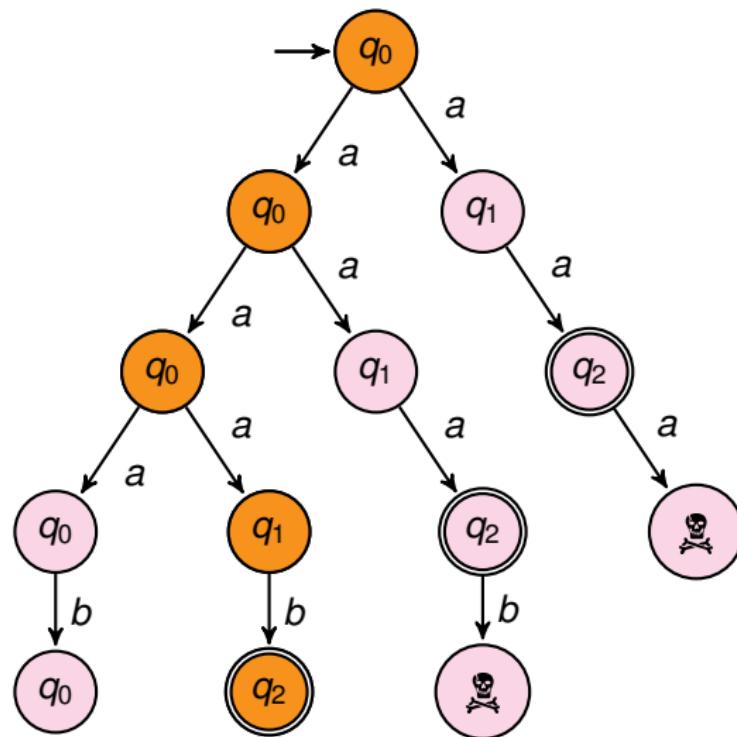
# Run Tree of aaab

---

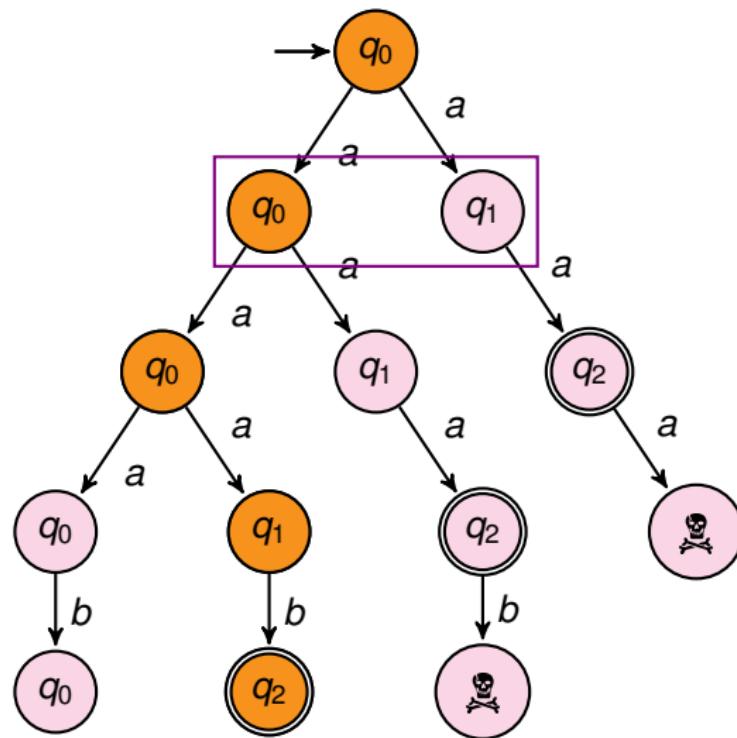


# Run Tree of aaab

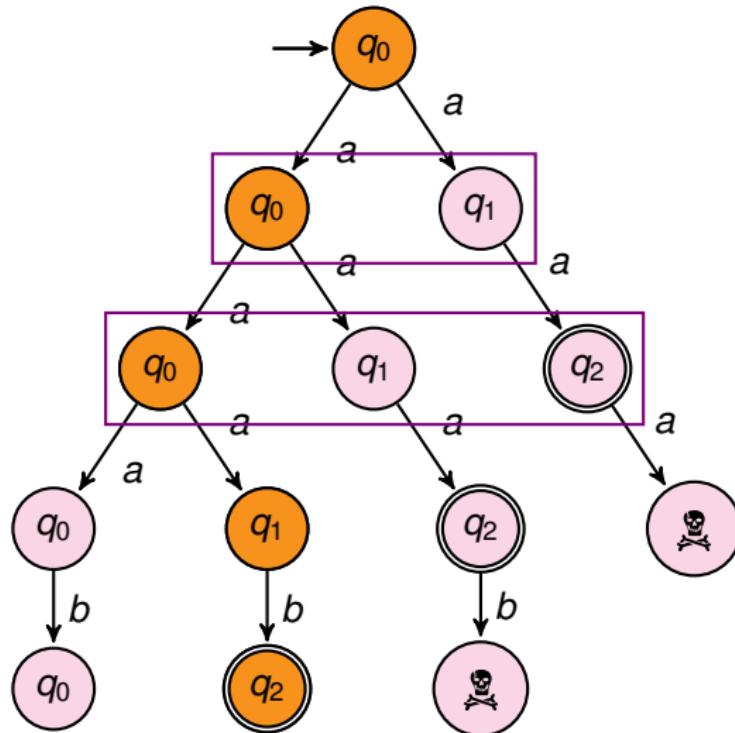
---



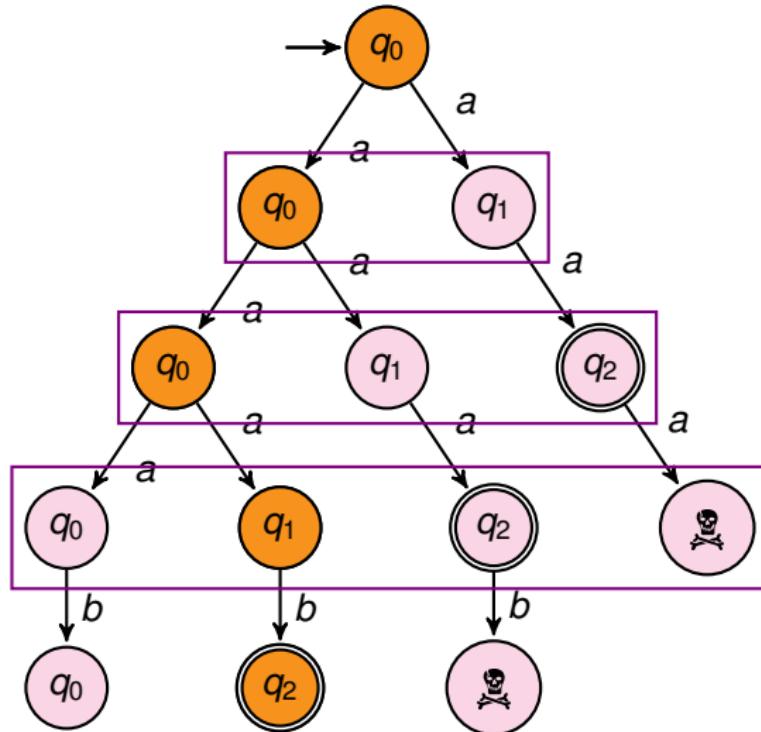
# Run Tree of aaab



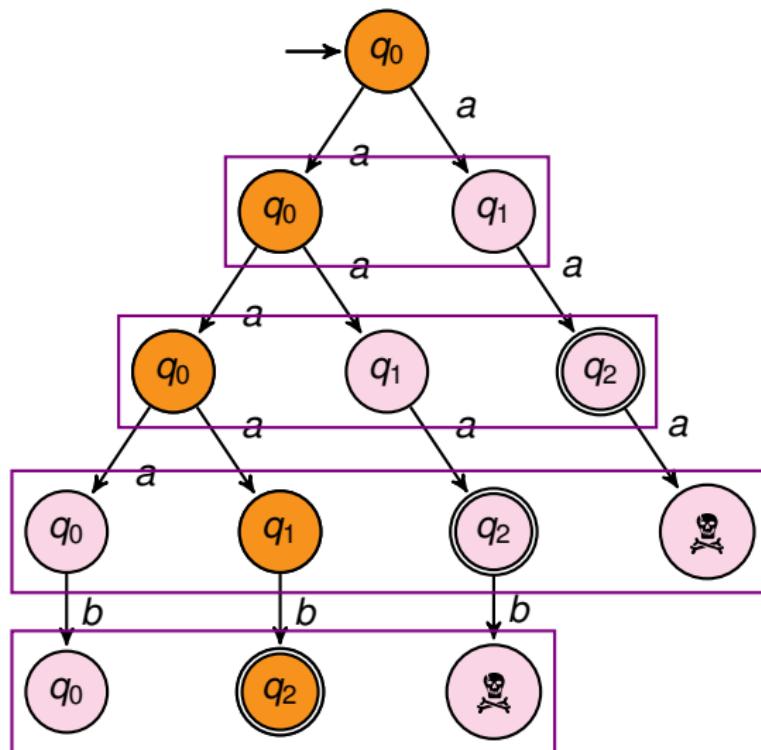
# Run Tree of aaab



# Run Tree of aaab

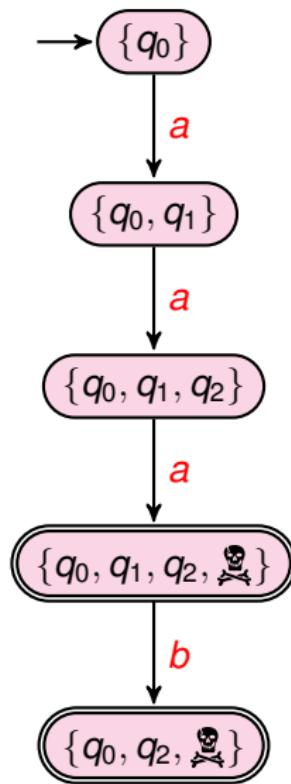


# Run Tree of aaab



# The Single Run

---



# NFA and DFA

---

- ▶ Any DFA is also an NFA

# NFA and DFA

---

- ▶ Any DFA is also an NFA
- ▶ Any NFA can be converted into a language equivalent DFA

# NFA and DFA

---

- ▶ Any DFA is also an NFA
- ▶ Any NFA can be converted into a language equivalent DFA
  - ▶ Combine all the runs of  $w$  in the NFA into a single run in the DFA

# NFA and DFA

---

- ▶ Any DFA is also an NFA
- ▶ Any NFA can be converted into a language equivalent DFA
  - ▶ Combine all the runs of  $w$  in the NFA into a single run in the DFA
  - ▶ Combine states occurring in various runs to obtain a set of states

# NFA and DFA

---

- ▶ Any DFA is also an NFA
- ▶ Any NFA can be converted into a language equivalent DFA
  - ▶ Combine all the runs of  $w$  in the NFA into a single run in the DFA
  - ▶ Combine states occurring in various runs to obtain a set of states
  - ▶ A set of states evolves into another set of states

# NFA and DFA

---

- ▶ Any DFA is also an NFA
- ▶ Any NFA can be converted into a language equivalent DFA
  - ▶ Combine all the runs of  $w$  in the NFA into a single run in the DFA
  - ▶ Combine states occurring in various runs to obtain a set of states
  - ▶ A set of states evolves into another set of states
  - ▶ Use  $\delta : Q \times \Sigma \rightarrow 2^Q$ , obtain  $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$

# NFA and DFA

---

- ▶ Any DFA is also an NFA
- ▶ Any NFA can be converted into a language equivalent DFA
  - ▶ Combine all the runs of  $w$  in the NFA into a single run in the DFA
  - ▶ Combine states occurring in various runs to obtain a set of states
  - ▶ A set of states evolves into another set of states
  - ▶ Use  $\delta : Q \times \Sigma \rightarrow 2^Q$ , obtain  $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$
  - ▶  $\Delta$  is an extension of  $\delta$

# NFA and DFA

---

- ▶ Any DFA is also an NFA
- ▶ Any NFA can be converted into a language equivalent DFA
  - ▶ Combine all the runs of  $w$  in the NFA into a single run in the DFA
  - ▶ Combine states occurring in various runs to obtain a set of states
  - ▶ A set of states evolves into another set of states
  - ▶ Use  $\delta : Q \times \Sigma \rightarrow 2^Q$ , obtain  $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$
  - ▶  $\Delta$  is an extension of  $\delta$
  - ▶ Accept if the obtained set of states contains a final state

# NFA and DFA

---

Given NFA  $N = (Q, \Sigma, Q_0, \delta, F)$ , obtain the DFA  $D = (2^Q, \Sigma, Q_0, \Delta, F')$

# NFA and DFA

---

Given NFA  $N = (Q, \Sigma, Q_0, \delta, F)$ , obtain the DFA  $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$  is defined by  $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$

# NFA and DFA

---

Given NFA  $N = (Q, \Sigma, Q_0, \delta, F)$ , obtain the DFA  $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- ▶  $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$  is defined by  $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- ▶  $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

# NFA and DFA

---

Given NFA  $N = (Q, \Sigma, Q_0, \delta, F)$ , obtain the DFA  $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- ▶  $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$  is defined by  $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- ▶  $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

Note that  $\hat{\delta}(A, a) = \bigcup_{q \in A} \delta(q, a) = \Delta(A, a)$

# NFA and DFA

---

Given NFA  $N = (Q, \Sigma, Q_0, \delta, F)$ , obtain the DFA  $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- ▶  $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$  is defined by  $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- ▶  $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

Note that  $\hat{\delta}(A, a) = \bigcup_{q \in A} \delta(q, a) = \Delta(A, a)$

Show that

- ▶  $\hat{\Delta} : 2^Q \times \Sigma^* \rightarrow 2^Q$  is same as  $\hat{\delta} : 2^Q \times \Sigma^* \rightarrow 2^Q$  (recall  $\delta : Q \times \Sigma \rightarrow 2^Q$ )

# NFA and DFA

---

Given NFA  $N = (Q, \Sigma, Q_0, \delta, F)$ , obtain the DFA  $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- ▶  $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$  is defined by  $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- ▶  $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

Note that  $\hat{\delta}(A, a) = \bigcup_{q \in A} \delta(q, a) = \Delta(A, a)$

Show that

- ▶  $\hat{\Delta} : 2^Q \times \Sigma^* \rightarrow 2^Q$  is same as  $\hat{\delta} : 2^Q \times \Sigma^* \rightarrow 2^Q$  (recall  $\delta : Q \times \Sigma \rightarrow 2^Q$ )
- ▶  $\hat{\Delta}(A, xa) = \Delta(\hat{\Delta}(A, x), a) = \bigcup_{q \in \hat{\Delta}(A, x)} \delta(q, a)$

# NFA and DFA

---

Given NFA  $N = (Q, \Sigma, Q_0, \delta, F)$ , obtain the DFA  $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- ▶  $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$  is defined by  $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- ▶  $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

Note that  $\hat{\delta}(A, a) = \bigcup_{q \in A} \delta(q, a) = \Delta(A, a)$

Show that

- ▶  $\hat{\Delta} : 2^Q \times \Sigma^* \rightarrow 2^Q$  is same as  $\hat{\delta} : 2^Q \times \Sigma^* \rightarrow 2^Q$  (recall  $\delta : Q \times \Sigma \rightarrow 2^Q$ )
- ▶  $\hat{\Delta}(A, xa) = \Delta(\hat{\Delta}(A, x), a) = \bigcup_{q \in \hat{\Delta}(A, x)} \delta(q, a)$
- ▶  $\hat{\delta}(A, xa) = \bigcup_{q \in \hat{\delta}(A, x)} \delta(q, a)$

# NFA = DFA

---

$$x \in L(D) \leftrightarrow \hat{\Delta}(Q_0, x) \in F'$$

$\leftrightarrow$

$$\hat{\delta}(Q_0, x) \in F'$$

$\leftrightarrow$

$$\hat{\delta}(Q_0, x) \cap F \neq \emptyset$$

$\leftrightarrow$

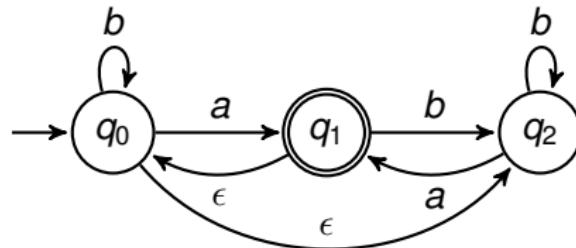
$$x \in L(N)$$

# Regularity

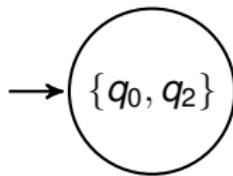
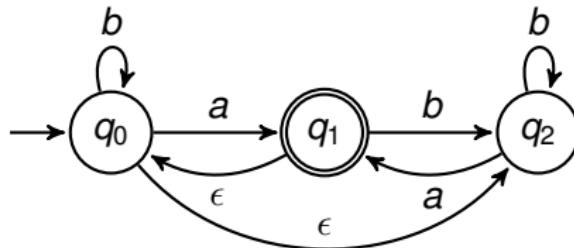
---

A language  $L$  is regular iff there exists an NFA  $A$  such that  $L = L(A)$

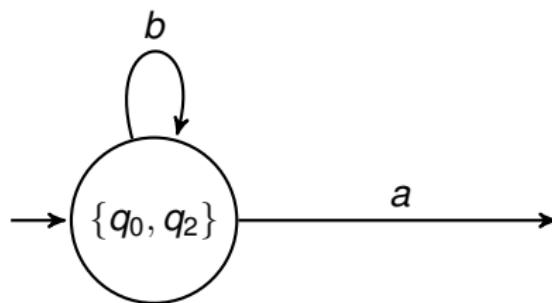
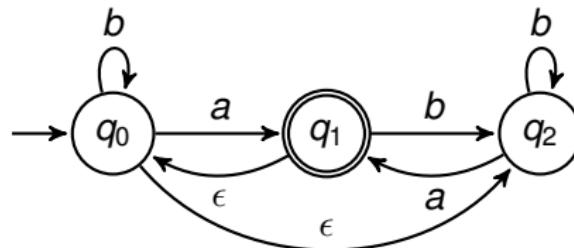
# $\epsilon$ -NFA



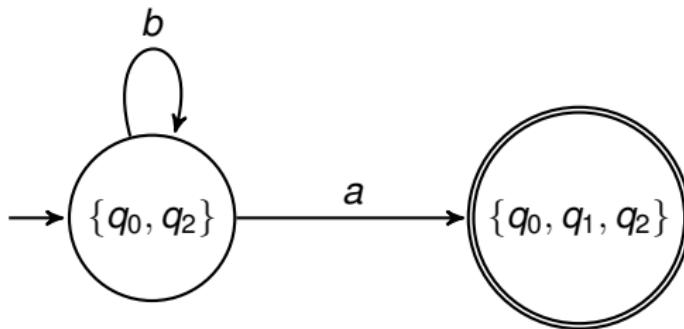
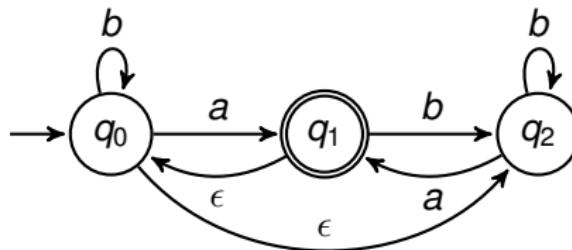
# $\epsilon$ -NFA



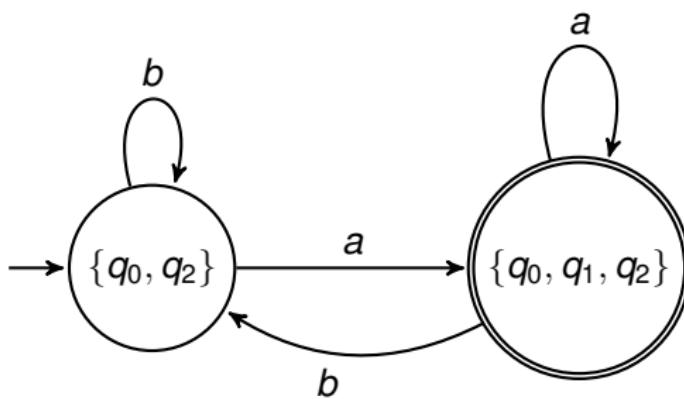
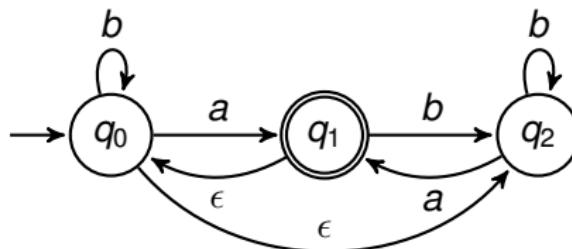
# $\epsilon$ -NFA



# $\epsilon$ -NFA



# $\epsilon$ -NFA



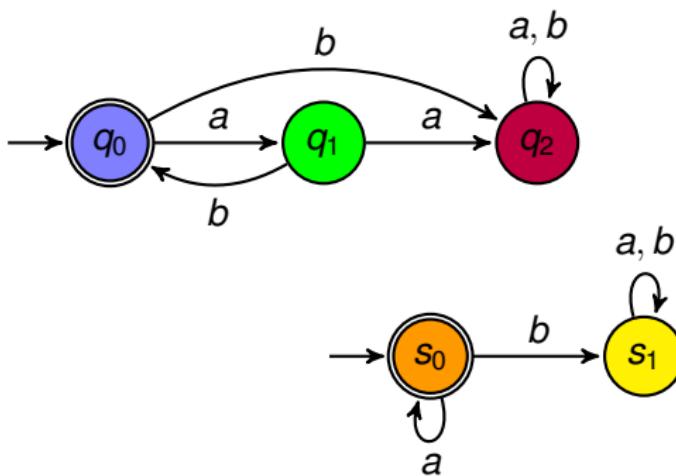
# $\epsilon$ -NFA and DFA

---

- ▶  $\epsilon$ -close the initial states of the  $\epsilon$ -NFA to obtain initial state of DFA
- ▶ From a state  $S$ , compute  $\Delta(S, a)$  and  $\epsilon$ -close it
- ▶ All states in the DFA are  $\epsilon$ -closed
- ▶ Final states are those which contain a final state of the  $\epsilon$ -NFA

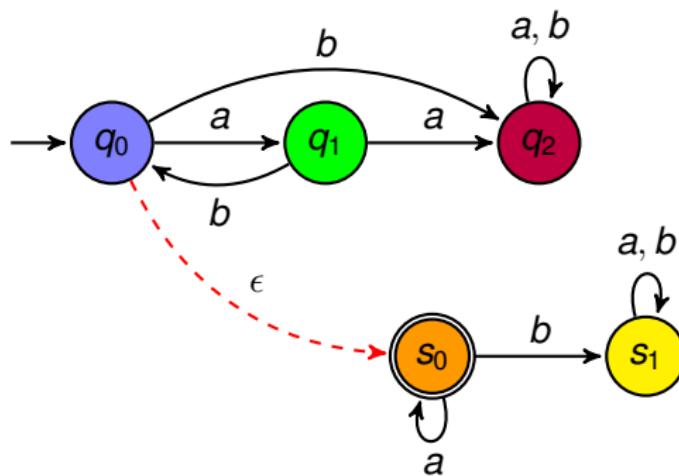
# Closure under Concatenation

- Given regular languages  $L_1, L_2$ , is  $L_1 \cdot L_2$  regular



# Closure under Concatenation

- Given regular languages  $L_1, L_2$ , is  $L_1 \cdot L_2$  regular?



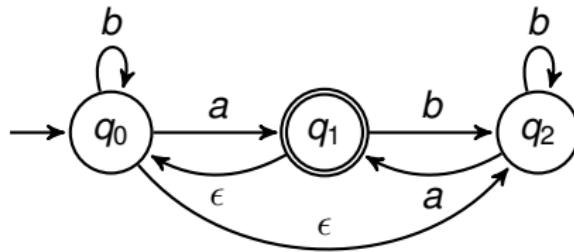
# Lecture 19 FOL NFA to DFA 1.pdf

# CS 228 : Logic in Computer Science

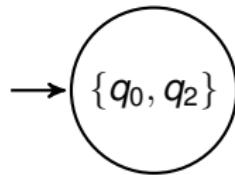
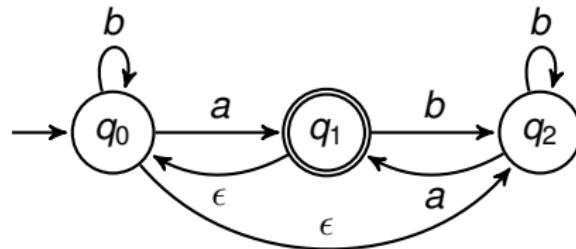
Krishna. S

# $\epsilon$ -NFA

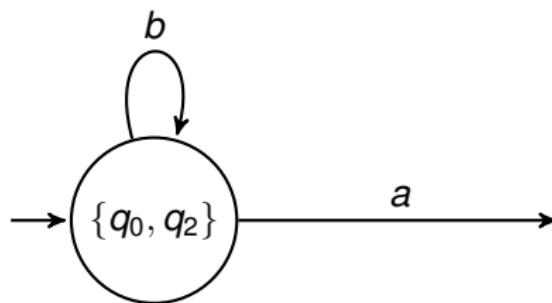
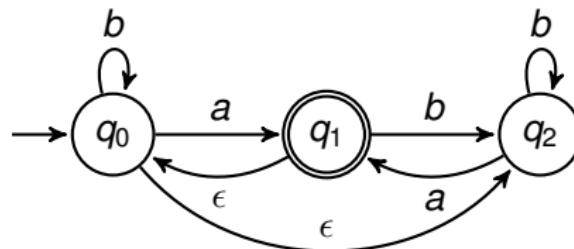
---



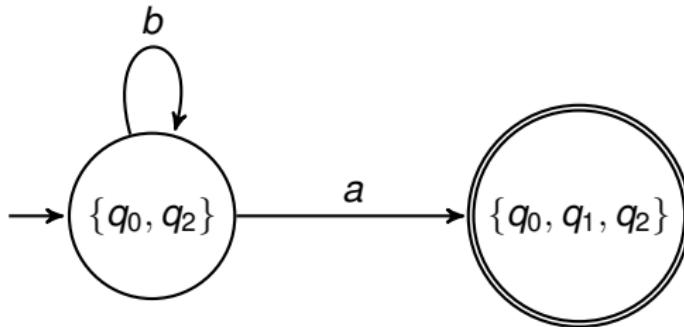
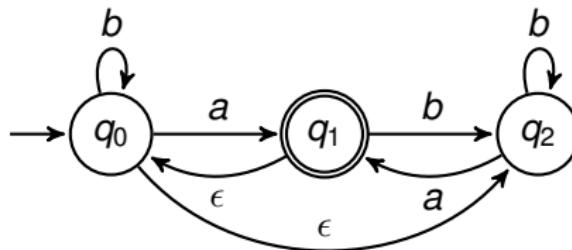
# $\epsilon$ -NFA



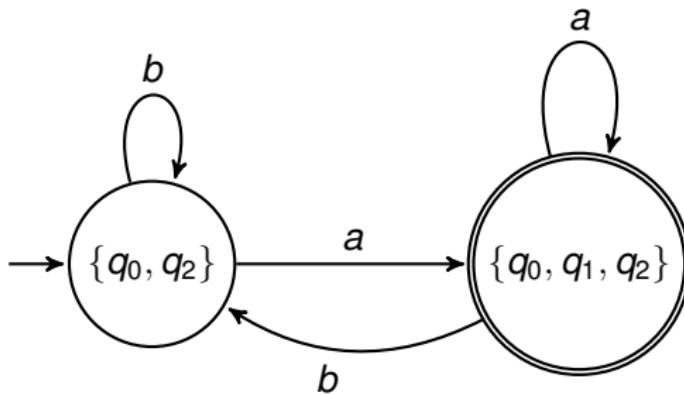
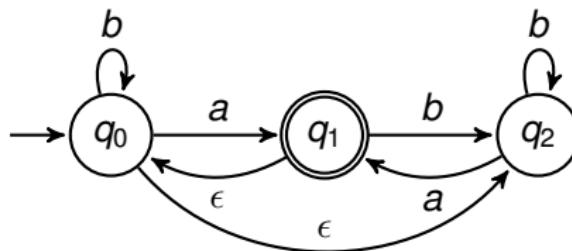
# $\epsilon$ -NFA



# $\epsilon$ -NFA



# $\epsilon$ -NFA



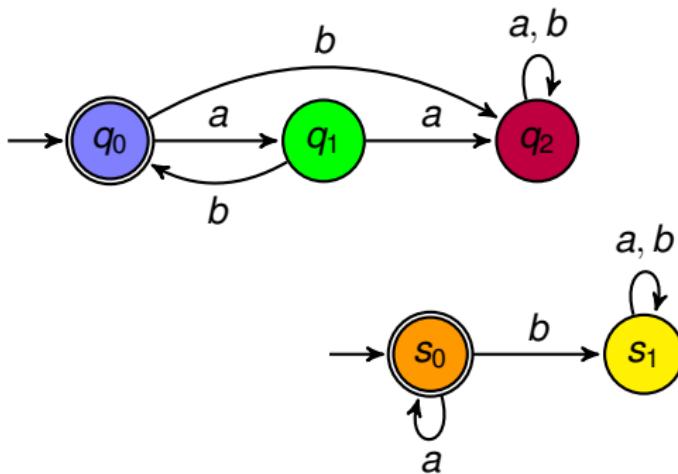
# $\epsilon$ -NFA and DFA

---

- ▶  $\epsilon$ -close the initial states of the  $\epsilon$ -NFA to obtain initial state of DFA
- ▶ From a state  $S$ , compute  $\Delta(S, a)$  and  $\epsilon$ -close it
- ▶ All states in the DFA are  $\epsilon$ -closed
- ▶ Final states are those which contain a final state of the  $\epsilon$ -NFA

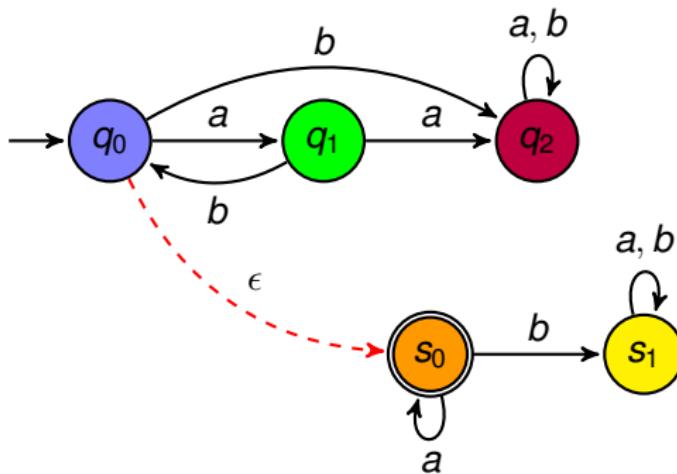
# Closure under Concatenation

- Given regular languages  $L_1, L_2$ , is  $L_1 \cdot L_2$  regular



# Closure under Concatenation

- Given regular languages  $L_1, L_2$ , is  $L_1 \cdot L_2$  regular?



# Formulae to Automaton

---

- ▶ FO-definable  $\Rightarrow$  regular
- ▶ Given an FO formula  $\varphi$ , construct a DFA  $A_\varphi$  such that  $L(\varphi) = L(A_\varphi)$
- ▶ If  $L(A_\varphi) = \emptyset$ , then  $\varphi$  is unsatisfiable
- ▶ If  $L(A_\varphi) \neq \emptyset$ , then  $\varphi$  is satisfiable

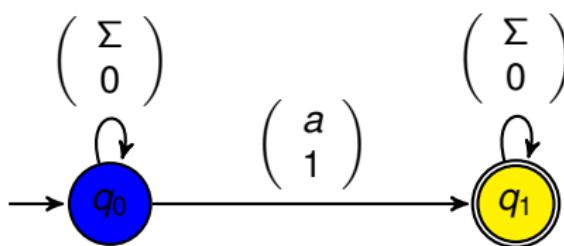
# FO to Regular Languages

---

- ▶ Every FO sentence  $\varphi$  over words can be converted into a DFA  $A_\varphi$  such that  $L(\varphi) = L(A_\varphi)$ .
- ▶ Start with atomic formulae, construct DFA for each of them.
- ▶ Conjunctions, disjunctions, negation of formulae easily handled via union, intersection and complementation of respective DFA
- ▶ Handling quantifiers?

# Atomic Formulae to DFA

- ▶  $Q_a(x)$  : All words which have an  $a$ . Need to fix a position for  $x$ , where  $a$  holds.
- ▶  $baab$  satisfies  $Q_a(x)$  with assignment  $x = 1$  or  $x = 2$ .
- ▶ Think of this as  $\begin{matrix} baab \\ 0010 \end{matrix}$  or  $\begin{matrix} baab \\ 0100 \end{matrix}$
- ▶ The first row is over  $\Sigma$ , and the second row captures a possible assignment to  $x$
- ▶ Think of an extended alphabet  $\Sigma' = \Sigma \times \{0, 1\}$ , and construct an automaton over  $\Sigma'$ .
- ▶ Deterministic, not complete.



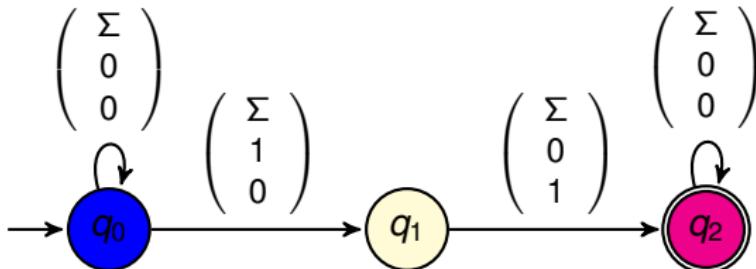
# Atomic Formulae to DFA : $S(x, y)$

- ▶  $bab$  satisfies  $S(x, y)$  with assignment  $x = 0$  or  $y = 1$  or  $x = 1, y = 2$ .

$bab$        $bab$

- ▶ Think of this as  
100      010  
010      001

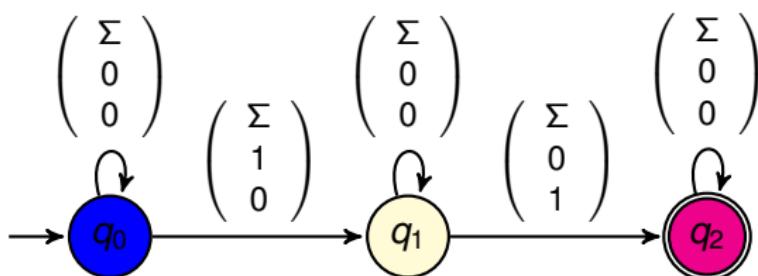
- ▶ The first row is over  $\Sigma$ , and the second, third rows capture a possible assignment to  $x, y$
- ▶ Think of an extended alphabet  $\Sigma' = \Sigma \times \{0, 1\}^2$ , and construct an automaton over  $\Sigma'$ .
- ▶ Deterministic, not complete.



# Atomic Formulae to DFA : $x < y$

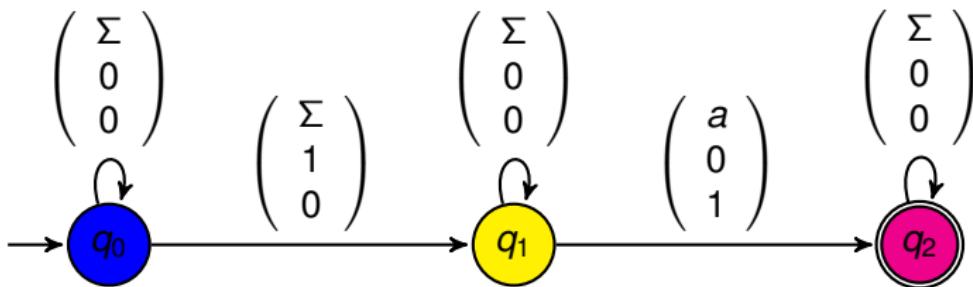
---

- bab satisfies  $x < y$  with assignment  $x = 0$  or  $y = 1$  or  $x = 1, y = 2$  or  $x = 0, y = 2$ .



# Simple Formulae to DFA

- ▶  $x < y \wedge Q_a(y)$
- ▶  $\Sigma' = \Sigma \times \{0, 1\} \times \{0, 1\}$
- ▶ Obtain intersection of DFA for  $x < y$  and  $Q_a(y)$



# Lecture 20 FOL NFA to DFA 2.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Formulae to DFA

---

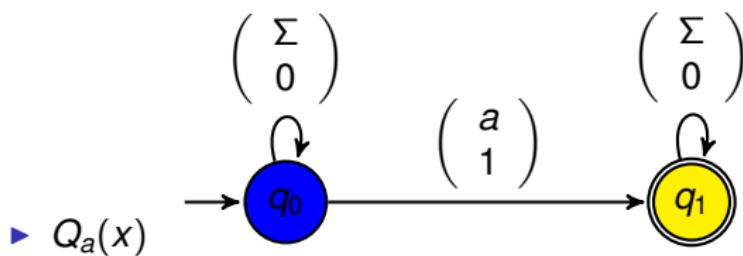
- Given  $\varphi(x_1, \dots, x_n)$ , a FO formula over  $\Sigma$ , consider the extended alphabet

$$\Sigma' = \Sigma \times \{0, 1\}^n$$

- Assign values to  $x_i$  at every position as seen in the cases of atomic formulae
- Keep in mind that every  $x_i$  can be assigned 1 at a unique position

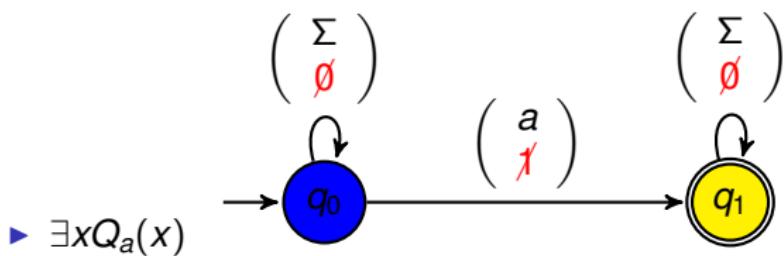
# Quantifiers

---



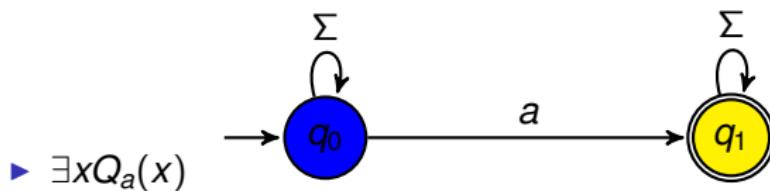
# Handling Quantifiers

---



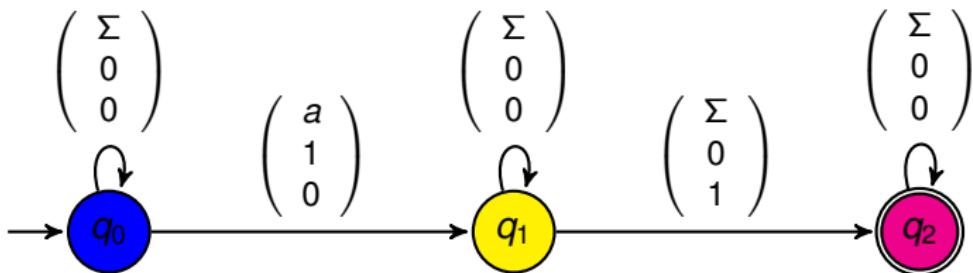
# Handling Quantifiers

---



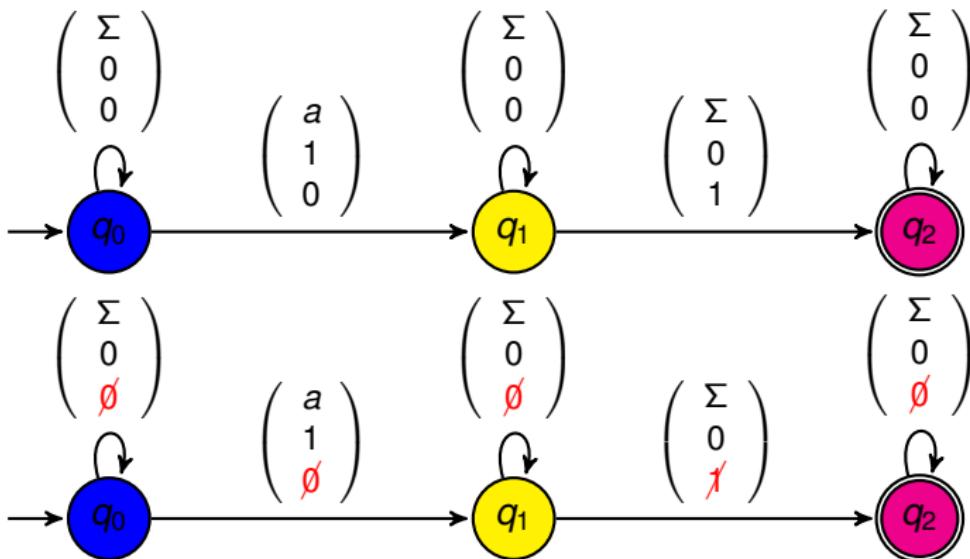
# Handling Quantifiers

- ▶  $Q_a(x) \wedge \exists y(x < y)$



# Handling Quantifiers

- $Q_a(x) \wedge \exists y(x < y)$



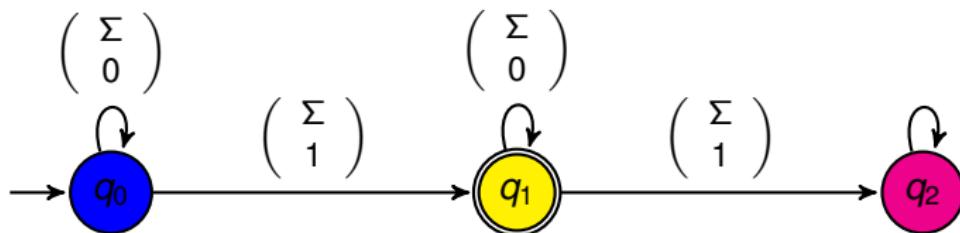
# Handling Quantifiers: $\forall x(x \neq x)$

---

# Handling Quantifiers: $\forall x(x \neq x)$

---

- ▶  $(x = x)$

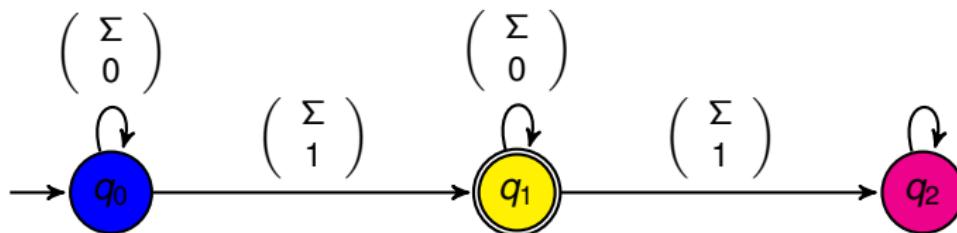


- ▶  $\exists x(x = x)$

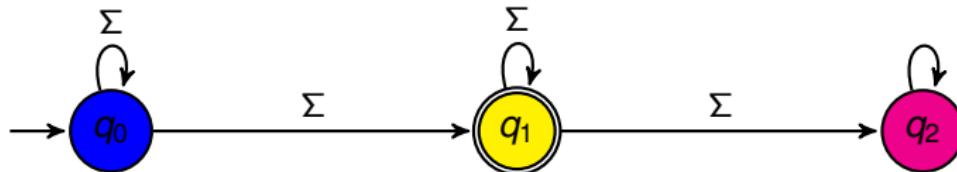
# Handling Quantifiers: $\forall x(x \neq x)$

---

- ▶  $(x = x)$



- ▶  $\exists x(x = x)$



- ▶  $\neg \exists x(x = x)$

# Handling Quantifiers : Summary

---

- ▶ Let  $L \subseteq (\Sigma \times \{0, 1\}^n)^*$  be defined by  $\varphi(x_1, \dots, x_n)$ .
- ▶ Let  $f : (\Sigma \times \{0, 1\}^n)^* \rightarrow (\Sigma \times \{0, 1\}^{n-1})^*$  be the projection  $f(w, c_1, \dots, c_n) = (w, c_1, \dots, c_{n-1})$ .
- ▶ Then  $\exists x_n \varphi(x_1, \dots, x_{n-1})$  defines  $f(L)$ .

# Handling Quantifiers : Done on Board

---

- ▶  $\exists x \forall y [x > y \vee \neg Q_a(x)] = \exists x [\neg \exists y [x \leq y \wedge Q_a(x)]]$
- ▶ Draw the automaton for  $[x \leq y \wedge Q_a(x)]$
- ▶ Project out the  $y$ -row
- ▶ Determinize it, and complement it
- ▶ Fix the  $x$ -row : Intersect with  $\left(\begin{array}{c} \Sigma \\ 0 \end{array}\right)^* \left(\begin{array}{c} \Sigma \\ 1 \end{array}\right) \left(\begin{array}{c} \Sigma \\ 0 \end{array}\right)^*$
- ▶ Project the  $x$ -row

# Points to Remember

---

- ▶ Given  $\varphi(x_1, \dots, x_n)$ , construct automaton for atomic FO formulae over the extended alphabet  $\Sigma \times \{0, 1\}^n$
- ▶ Intersect with the regular language where every  $x_i$  is assigned 1 exactly at one position
- ▶ Given a sentence  $Q_{x_1} \dots Q_{x_n} \varphi$ , first construct the automaton for the formula  $\varphi(x_1, \dots, x_n)$
- ▶ Replace  $\forall$  in terms of  $\exists$

# Points to Remember

---

- ▶ Given the automaton for  $\varphi(x_1, \dots, x_n)$ , the automaton for  $\exists x_i \varphi(x_1, \dots, x_n)$  is obtained by **projecting out** the row of  $x_i$
- ▶ This may result in an NFA
- ▶ Determinize it and complement it to get a DFA for  $\neg \exists x_i \varphi(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$
- ▶ Intersect with the regular language where each of  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$  are assigned 1 exactly at one position

# The Computational Effort

---

Given NFAs  $A_1, A_2$  each with atmost  $n$  states,

- ▶ The union has atmost  $2n + 1$  states
- ▶ Intersection has atmost  $n^2$  states
- ▶ The complement has atmost  $2^n$  states
- ▶ The projection has atmost  $n$  states

# The Computational Effort

---

- ▶  $\psi = Q_1 \dots Q_n \varphi$ . If  $Q_i = \exists$  for all  $i$ , then size of  $A_\psi$  is same the size of  $A_\varphi$ .
- ▶ When  $Q_1 = \exists, Q_2 = \forall, \dots$  : each  $\forall$  quantifier can create a  $2^n$  blowup in automaton size
- ▶ Size of automaton is

$$2^{2^{2^{2^{2^n}}}}$$

where the tower height  $k$  is the quantifier alternation size.

- ▶ This number is indeed a lower bound!

# The Automaton-Logic Connection

---

Given any FO sentence  $\varphi$ , one can construct a DFA  $A_\varphi$  such that  $L(\varphi) = L(A_\varphi)$ .

# Summary

---

- ▶ Given FO formula  $\varphi$ , build an automaton  $A_\varphi$  preserving the language
- ▶ Satisfiability of FO reduces to non-emptiness of underlying automaton

# Lecture 21 FOL NFA to DFA 3.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Handling Quantifiers : Done on Board

---

- ▶  $\exists x \forall y [x > y \vee \neg Q_a(x)] = \exists x [\neg \exists y [x \leq y \wedge Q_a(x)]]$
- ▶ Draw the automaton for  $[x \leq y \wedge Q_a(x)]$
- ▶ Project out the  $y$ -row
- ▶ Determinize it, and complement it
- ▶ Fix the  $x$ -row : Intersect with  $\left(\begin{array}{c} \Sigma \\ 0 \end{array}\right)^* \left(\begin{array}{c} \Sigma \\ 1 \end{array}\right) \left(\begin{array}{c} \Sigma \\ 0 \end{array}\right)^*$
- ▶ Project the  $x$ -row

# Points to Remember

---

- ▶ Given  $\varphi(x_1, \dots, x_n)$ , construct automaton for atomic FO formulae over the extended alphabet  $\Sigma \times \{0, 1\}^n$
- ▶ Intersect with the regular language where every  $x_i$  is assigned 1 exactly at one position
- ▶ Given a sentence  $Q_{x_1} \dots Q_{x_n} \varphi$ , first construct the automaton for the formula  $\varphi(x_1, \dots, x_n)$
- ▶ Replace  $\forall$  in terms of  $\exists$

# Points to Remember

---

- ▶ Given the automaton for  $\varphi(x_1, \dots, x_n)$ , the automaton for  $\exists x_i \varphi(x_1, \dots, x_n)$  is obtained by **projecting out** the row of  $x_i$
- ▶ This may result in an NFA
- ▶ Determinize it and complement it to get a DFA for  $\neg \exists x_i \varphi(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$
- ▶ Intersect with the regular language where each of  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$  are assigned 1 exactly at one position

# The Computational Effort

---

Given NFAs  $A_1, A_2$  each with atmost  $n$  states,

- ▶ The union has atmost  $2n$  states
- ▶ Intersection has almost  $n^2$  states
- ▶ The complement has atmost  $2^n$  states
- ▶ The projection has atmost  $n$  states

# Cost of determinization : $n + 1$ to $2^n$

---

- ▶  $\Sigma = \{0, 1\}$ , languages where the  $n^{th}$  bit from the right is a 1.
- ▶ NFA has  $n + 1$  states.
- ▶ Size of corresponding DFA?

# The Computational Effort

---

- ▶  $\psi = Q_1 \dots Q_n \varphi$ . If  $Q_i = \exists$  for all  $i$ , then size of  $A_\psi$  is same the size of  $A_\varphi$ .
- ▶ When  $Q_1 = \exists, Q_2 = \forall, \dots$  : each  $\forall$  quantifier can create a  $2^n$  blowup in automaton size
- ▶ Size of automaton is

$$2^{2^{2^{2^{2^n}}}}$$

where the tower height  $k$  is the quantifier alternation size.

- ▶ This number is indeed a lower bound!

# The Automaton-Logic Connection

---

Given any FO sentence  $\varphi$ , one can construct a DFA  $A_\varphi$  such that  $L(\varphi) = L(A_\varphi)$ .

# Summary

---

- ▶ Given FO formula  $\varphi$ , build an automaton  $A_\varphi$  preserving the language
- ▶ Satisfiability of FO reduces to non-emptiness of underlying automaton

# Satisfiability to Model Checking

---

- ▶ Satisfiability of FO over words
- ▶ Model checking
  - ▶ System abstracted as a model DFA/NFA  $A$
  - ▶ Specification written in FO as formula  $\varphi$
  - ▶ Does system model  $\models \varphi$
  - ▶  $L(A) \subseteq L(\varphi)$ ?
  - ▶  $L(A) \cap \overline{L(\varphi)} = \emptyset$ ?
- ▶ FO-definable  $\subseteq REG$

# Next directions

---

- ▶ Going back to general FO, and discuss the nontermination of the satisfiability checking procedure (Shawn Hedman)
- ▶ Inexpressiveness of FO : EF games (Straubing)
- ▶ MSO logic that can capture exactly regular languages (Wolfgang Thomas AAT)
- ▶ Temporal Logics (only LTL) (Baier-Katoen)
- ▶ Immediate next : MSO

## Monadic Second Order Logic (MSO)

# Symbols in MSO

---

Formulae of MSO, over signature  $\tau$ , are sequences of symbols, where each symbol is one of the following:

- ▶ The symbol  $\perp$  called **false**
- ▶ An element of the infinite set  $\mathcal{V}_1 = \{x_1, x_2, \dots\}$  of **first order variables**
- ▶ An element of the infinite set  $\mathcal{V}_2 = \{X_1, X_2, \dots\}$  of **second order variables** where each variable has arity 1 (**new!**)
- ▶ Constants and relations from  $\tau$
- ▶ The connectives  $\rightarrow, \wedge, \vee, \neg$
- ▶ The quantifiers  $\forall, \exists$
- ▶ Paranthesis

# Well formed Formulae

---

A well-formed formula (wff) over a signature  $\tau$  is inductively defined as follows:

- ▶  $\perp$  is a wff
- ▶ If  $t_1, t_2$  are either variables or constants in  $\tau$ , then  $t_1 = t_2$  is a wff
- ▶ If  $t_i$ 's are terms for  $1 \leq i \leq k$  and  $R$  is a  $k$ -ary relation symbol in  $\tau$ , then  $R(t_1, \dots, t_k)$  is a wff
- ▶ If  $t$  is either a first order variable or a constant,  $X$  is a second order variable, then  $X(t)$  is a wff
- ▶ If  $\varphi$  and  $\psi$  are wff, then  $\varphi \rightarrow \psi$ ,  $\varphi \wedge \psi$ ,  $\varphi \vee \psi$  and  $\neg\varphi$  are wff
- ▶ If  $\varphi$  is a wff and  $x$  is a first order variable, then  $(\forall x)\varphi$  and  $(\exists x)\varphi$  are wff
- ▶ If  $\varphi$  is a wff and  $X$  is a second order variable, then  $(\forall X)\varphi$  and  $(\exists X)\varphi$  are wff

# Lecture 22 MSO.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Free and Bound Variables in MSO

---

- ▶ Free, Bound Variables and Scope same as in FO
- ▶ In a wff  $\varphi = \forall X\psi$  or  $\exists X\psi$  every occurrence of  $X$  in  $\psi$  is bound
- ▶ A sentence is a formula with no free first order and second order variables

# Assignments on $\tau$ -structures

## Assignments

For a  $\tau$ -structure  $\mathcal{A}$ , an assignment over  $\mathcal{A}$  is a pair of functions  $(\alpha_1, \alpha_2)$ , where

- ▶  $\alpha_1 : \mathcal{V}_1 \rightarrow u(\mathcal{A})$  assigns every first order variable  $x \in \mathcal{V}_1$  a value  $\alpha_1(x) \in u(\mathcal{A})$ . If  $t$  is a constant symbol  $c$ , then  $\alpha_1(t)$  is  $c^{\mathcal{A}}$ .
- ▶  $\alpha_2 : \mathcal{V}_2 \rightarrow 2^{u(\mathcal{A})}$  assigns to every second order variable  $X \in \mathcal{V}_2$ ,  $\alpha_2(X) \subseteq u(\mathcal{A})$ .

## Binding on a Variable

For an assignment  $\alpha = (\alpha_1, \alpha_2)$  over  $\mathcal{A}$ , and  $x \in \mathcal{V}_i$ ,  $i = 1, 2$ ,

$$\alpha_i[x \mapsto a] \text{ is the assignment } \alpha_i[x \mapsto a](y) = \begin{cases} \alpha_i(y), & y \neq x, \\ a, & y = x \end{cases}$$

# Satisfaction

---

We define the relation  $\mathcal{A} \models_{\alpha} \varphi$  (read as  $\varphi$  is true in  $\mathcal{A}$  under the assignment  $\alpha$ ) inductively:

- ▶  $\mathcal{A} \not\models_{\alpha} \perp$
- ▶  $\mathcal{A} \models_{\alpha} t_1 = t_2$  iff  $\alpha_1(t_1) = \alpha_1(t_2)$
- ▶  $\mathcal{A} \models_{\alpha} R(t_1, \dots, t_k)$  iff  $(\alpha_1(t_1), \dots, \alpha_1(t_k)) \in R^{\mathcal{A}}$
- ▶  $\mathcal{A} \models_{\alpha} X(t)$  iff  $\alpha_1(t) \in \alpha_2(X)$  (**new**)
- ▶  $\mathcal{A} \models_{\alpha} (\varphi \rightarrow \psi)$  iff  $\mathcal{A} \not\models_{\alpha} \varphi$  or  $\mathcal{A} \models_{\alpha} \psi$
- ▶  $\mathcal{A} \models_{\alpha} (\forall x)\varphi$  iff for every  $a \in u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[x \mapsto a]} \varphi$
- ▶  $\mathcal{A} \models_{\alpha} (\forall X)\varphi$  iff for every  $S \subseteq u(\mathcal{A})$ ,  $\mathcal{A} \models_{\alpha[X \mapsto S]} \varphi$  (**new**)

# Examples

---

Recall the signature for the graph structure,  $\tau = \{E\}$

- ▶ The graph is 3-colorable

# Examples

---

Recall the signature for the graph structure,  $\tau = \{E\}$

- ▶ The graph is 3-colorable

$$\exists X \exists Y \exists Z (\forall x [X(x) \vee Y(x) \vee Z(x)] \wedge \\ \forall x \forall y [E(x, y) \rightarrow \{\neg(X(x) \wedge X(y)) \wedge \neg(Y(x) \wedge Y(y)) \wedge \neg(Z(x) \wedge Z(y))\}])$$

# Examples

---

Recall the signature for the graph structure,  $\tau = \{E\}$

- ▶ The graph has an independent set of size  $\geq k$

# Examples

---

Recall the signature for the graph structure,  $\tau = \{E\}$

- ▶ The graph has an independent set of size  $\geq k$

$$\exists I \{ \forall x \forall y [(\neg(x = y) \wedge I(x) \wedge I(y)) \rightarrow \neg E(x, y)] \wedge$$

$$\exists x_1 \dots x_k [\bigwedge_{i \neq j} \neg(x_i = x_j) \wedge \bigwedge_i I(x_i)] \}$$

# Examples

---

Recall the signature  $\tau$  for the word structure,  $\tau = \{Q_a, Q_b, <, S\}$  for  $\Sigma = \{a, b\}$

- ▶ Words of even length

# Examples

---

Recall the signature  $\tau$  for the word structure,  $\tau = \{Q_a, Q_b, <, S\}$  for  $\Sigma = \{a, b\}$

- ▶ Words of even length

$$\exists E \exists O \{ \forall x [(\textit{first}(x) \rightarrow E(x)) \wedge (\textit{last}(x) \rightarrow O(x))] \}$$

# Examples

---

Recall the signature  $\tau$  for the word structure,  $\tau = \{Q_a, Q_b, <, S\}$  for  $\Sigma = \{a, b\}$

- ▶ Words of even length

$$\exists E \exists O \{ \forall x [(\text{first}(x) \rightarrow E(x)) \wedge (\text{last}(x) \rightarrow O(x))] \wedge \forall x [((E(x) \vee O(x)) \wedge \neg(E(x) \wedge O(x)))]$$

# Examples

---

Recall the signature  $\tau$  for the word structure,  $\tau = \{Q_a, Q_b, <, S\}$  for  $\Sigma = \{a, b\}$

- ▶ Words of even length

$$\exists E \exists O \{ \forall x [(\text{first}(x) \rightarrow E(x)) \wedge (\text{last}(x) \rightarrow O(x))] \wedge$$

$$\wedge \forall x [(\text{E}(x) \vee \text{O}(x)) \wedge \neg(\text{E}(x) \wedge \text{O}(x))] \wedge$$

$$\wedge \forall x \forall y [S(x, y) \wedge \text{O}(x) \rightarrow \text{E}(y)] \wedge$$

$$\wedge \forall x \forall y [S(x, y) \wedge \text{E}(x) \rightarrow \text{O}(y)] \}$$

## MSO on Words : Satisfiability

# MSO on Words

---

- ▶ Signature  $\tau = (Q_\Sigma, <, S)$ , domain or universe = set of positions of a word
- ▶ MSO over words: Atomic formulae

$$X(x)|Q_\Sigma(x)|x = y|x < y|S(x, y)$$

- ▶ Given a MSO sentence  $\varphi$ ,  $L(\varphi)$  defined as usual
- ▶ A language  $L \subseteq \Sigma^*$  is MSO definable iff there is an MSO formula  $\varphi$  such that  $L = L(\varphi)$
- ▶ Given an MSO sentence  $\varphi$ , is it satisfiable/valid?

# MSO Expressiveness

---

- ▶ Clearly,  $FO \subseteq MSO$
- ▶  $MSO = \text{Regular}$

# Lecture 23 MSO Regular Languages.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Regular Languages to MSO

---

Given a regular language  $L$ , and a DFA such that  $L = L(A)$ ,

- ▶ Run of a word : at every position of the word, we are in some unique state

# Regular Languages to MSO

---

Given a regular language  $L$ , and a DFA such that  $L = L(A)$ ,

- ▶ Run of a word : at every position of the word, we are in some unique state

Position $x$	:	0	1	2	3
$a$		$a$	$b$	$a$	
$q_0$		$q_1$	$q_0$	$q_2$	$q_2$

# Regular Languages to MSO

---

Given a regular language  $L$ , and a DFA such that  $L = L(A)$ ,

- ▶ Run of a word : at every position of the word, we are in some unique state

Position x :	0	1	2	3
	a	a	b	a
	$q_0$	$q_1$	$q_0$	$q_2$

- ▶ For a state  $q \in Q$ , let  $X_q$ =the set of positions of the word where the state is  $q$  in the run

# Regular Languages to MSO

---

Given a regular language  $L$ , and a DFA such that  $L = L(A)$ ,

- ▶ Run of a word : at every position of the word, we are in some unique state

Position $x$	:	0	1	2	3
$a$		$a$	$b$	$a$	
$q_0$		$q_1$	$q_0$	$q_2$	$q_2$

- ▶ For a state  $q \in Q$ , let  $X_q$ =the set of positions of the word where the state is  $q$  in the run
- ▶  $X_{q_0} = \{0, 2\}, X_{q_1} = \{1\}, X_{q_2} = \{3\}$

# Regular Languages to MSO

---

Given a regular language  $L$ , and a DFA such that  $L = L(A)$ ,

- ▶ Run of a word : at every position of the word, we are in some unique state

Position $x$	:	0	1	2	3
$a$		$a$	$b$	$a$	
$q_0$		$q_1$	$q_0$	$q_2$	$q_2$

- ▶ For a state  $q \in Q$ , let  $X_q$ =the set of positions of the word where the state is  $q$  in the run
- ▶  $X_{q_0} = \{0, 2\}, X_{q_1} = \{1\}, X_{q_2} = \{3\}$
- ▶ The initial position of any word must belong to  $X_{q_0} : 0 \in X_{q_0}$

# Regular Languages to MSO

---

- ▶ If a word  $wa$  is accepted, then
  - ▶ The last position  $x$  of the word satisfies  $Q_a(x)$
  - ▶ For some state  $q$ , we have  $X_q(x)$  and there is a transition  $\delta(q, a) = q_f \in F$

# Regular Languages to MSO

---

- ▶ If a word  $wa$  is accepted, then
  - ▶ The last position  $x$  of the word satisfies  $Q_a(x)$
  - ▶ For some state  $q$ , we have  $X_q(x)$  and there is a transition  $\delta(q, a) = q_f \in F$

Position x :	0	1	2	3
	$a$	$a$	$b$	$a$
	$q_0$	$q_1$	$q_0$	$q_2$

# Regular Languages to MSO

---

- ▶ If a word  $wa$  is accepted, then
  - ▶ The last position  $x$  of the word satisfies  $Q_a(x)$
  - ▶ For some state  $q$ , we have  $X_q(x)$  and there is a transition  $\delta(q, a) = q_f \in F$

Position $x$	:	0	1	2	3
		$a$	$a$	$b$	$a$
		$q_0$	$q_1$	$q_0$	$q_2$

- ▶  $Q_a(3)$  and  $3 \in X_{q_2}$ .  $\delta(q_2, a) = q_2 \notin F$
- ▶ If  $x, y$  are consecutive positions in the word, and if  $X_q(x) \wedge Q_a(x)$ , then it must be that  $X_t(y)$  such that  $\delta(q, a) = t$

# Regular Languages to MSO

---

- ▶ If a word  $wa$  is accepted, then
  - ▶ The last position  $x$  of the word satisfies  $Q_a(x)$
  - ▶ For some state  $q$ , we have  $X_q(x)$  and there is a transition  $\delta(q, a) = q_f \in F$

Position $x$	:	0	1	2	3
		$a$	$a$	$b$	$a$
		$q_0$	$q_1$	$q_0$	$q_2$

- ▶  $Q_a(3)$  and  $3 \in X_{q_2}$ .  $\delta(q_2, a) = q_2 \notin F$
- ▶ If  $x, y$  are consecutive positions in the word, and if  $X_q(x) \wedge Q_a(x)$ , then it must be that  $X_t(y)$  such that  $\delta(q, a) = t$
- ▶  $X_{q_0}(0)$ ,  $X_{q_1}(1)$  and  $Q_a(0)$ .  $\delta(q_0, a) = q_1$ .
- ▶  $X_{q_1}(1)$ ,  $X_{q_0}(2)$  and  $Q_a(1)$ .  $\delta(q_1, a) = q_0$ .

# Regular Languages to MSO

---

Given a DFA  $A = (Q, \Sigma, \delta, q_0, F)$ , a word  $w$  is accepted iff it satisfies

$$\exists X_0 \exists X_1 \dots \exists X_n \{ [\forall x (X_0(x) \vee \dots \vee X_n(x)) \wedge \forall x \bigwedge_{i \neq j} \neg(X_i(x) \wedge X_j(x))] \wedge$$

# Regular Languages to MSO

---

Given a DFA  $A = (Q, \Sigma, \delta, q_0, F)$ , a word  $w$  is accepted iff it satisfies

$$\exists X_0 \exists X_1 \dots X_n \{ [\forall x (X_0(x) \vee \dots \vee X_n(x)) \wedge \forall x \bigwedge_{i \neq j} \neg(X_i(x) \wedge X_j(x))] \wedge \\ [\exists x (\text{first}(x) \wedge X_0(x))] \wedge$$

# Regular Languages to MSO

---

Given a DFA  $A = (Q, \Sigma, \delta, q_0, F)$ , a word  $w$  is accepted iff it satisfies

$$\exists X_0 \exists X_1 \dots X_n \{ [\forall x (X_0(x) \vee \dots \vee X_n(x)) \wedge \forall x \bigwedge_{i \neq j} \neg(X_i(x) \wedge X_j(x))] \wedge$$

$$[\exists x (\text{first}(x) \wedge X_0(x))] \wedge$$

$$\forall x \forall y [S(x, y) \rightarrow \bigvee_{\delta(i, a)=j} [X_i(x) \wedge Q_a(x) \wedge X_j(y)]] \wedge$$

# Regular Languages to MSO

---

Given a DFA  $A = (Q, \Sigma, \delta, q_0, F)$ , a word  $w$  is accepted iff it satisfies

$$\exists X_0 \exists X_1 \dots X_n \{ [\forall x (X_0(x) \vee \dots \vee X_n(x)) \wedge \forall x \bigwedge_{i \neq j} \neg(X_i(x) \wedge X_j(x))] \wedge$$

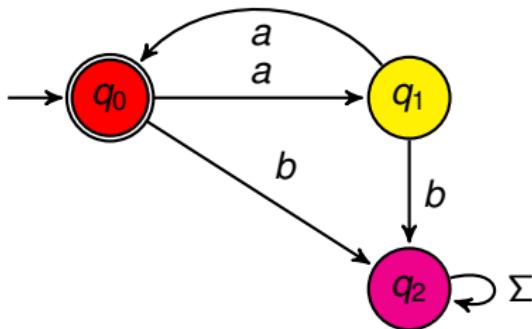
$$[\exists x (\text{first}(x) \wedge X_0(x))] \wedge$$

$$\forall x \forall y [S(x, y) \rightarrow \bigvee_{\delta(i, a)=j} [X_i(x) \wedge Q_a(x) \wedge X_j(y)]] \wedge$$

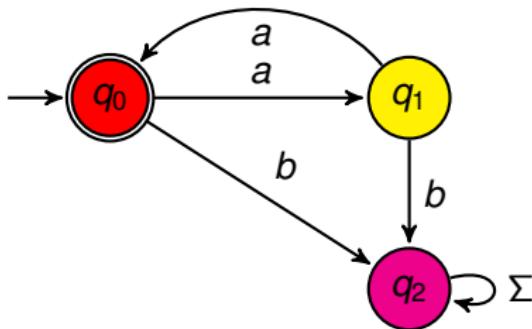
$$\exists x [\text{last}(x) \wedge \bigvee_{\delta(i, a)=j \in F} [X_i(x) \wedge Q_a(x)]] \}$$

- $w \in L(A)$  iff  $w \models \varphi$

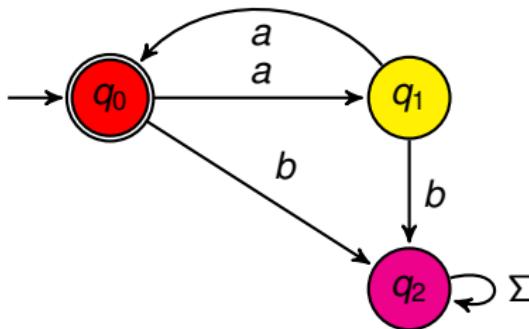
# Example : Regular to MSO


$$\exists X_0 \exists X_1 \exists X_2 \{ [\forall x (X_0(x) \vee X_1(x) \vee X_2(x)) \wedge \forall x [\neg(X_0(x) \wedge X_1(x)) \wedge \neg(X_0(x) \wedge X_2(x)) \wedge \neg(X_1(x) \wedge X_2(x))] \wedge$$

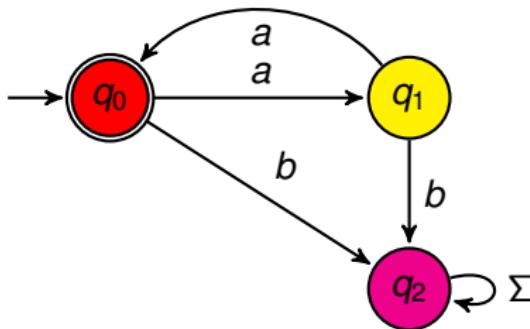
# Example : Regular to MSO


$$\exists X_0 \exists X_1 \exists X_2 \{ [\forall x (X_0(x) \vee X_1(x) \vee X_2(x)) \wedge \forall x [\neg(X_0(x) \wedge X_1(x)) \wedge \neg(X_0(x) \wedge X_2(x)) \wedge \neg(X_1(x) \wedge X_2(x))] \wedge [\exists x (\text{first}(x) \wedge X_0(x))] \wedge$$

# Example : Regular to MSO


$$\exists X_0 \exists X_1 \exists X_2 \{ \forall x (X_0(x) \vee X_1(x) \vee X_2(x)) \wedge \forall x [\neg(X_0(x) \wedge X_1(x)) \wedge \neg(X_0(x) \wedge X_2(x)) \wedge \neg(X_1(x) \wedge X_2(x))] \wedge [\exists x (first(x) \wedge X_0(x))] \wedge$$
$$\forall x \forall y [S(x, y) \rightarrow [(X_0(x) \wedge Q_a(x) \wedge X_1(y)) \vee (X_0(x) \wedge Q_b(x) \wedge X_2(y)) \vee (X_1(x) \wedge Q_a(x) \wedge X_0(y)) \vee (X_1(x) \wedge Q_b(x) \wedge X_2(y)) \vee (X_2(x) \wedge Q_a(x) \wedge X_1(y)) \vee (X_2(x) \wedge Q_b(x) \wedge X_0(y))]]$$

# Example : Regular to MSO


$$\exists X_0 \exists X_1 \exists X_2 \{ \forall x (X_0(x) \vee X_1(x) \vee X_2(x)) \wedge \forall x [\neg(X_0(x) \wedge X_1(x)) \wedge \neg(X_0(x) \wedge X_2(x)) \wedge \neg(X_1(x) \wedge X_2(x))] \wedge [\exists x (first(x) \wedge X_0(x))] \wedge$$
$$\forall x \forall y [S(x, y) \rightarrow [(X_0(x) \wedge Q_a(x) \wedge X_1(y)) \vee (X_0(x) \wedge Q_b(x) \wedge X_2(y)) \vee (X_1(x) \wedge Q_a(x) \wedge X_0(y)) \vee (X_1(x) \wedge Q_b(x) \wedge X_2(y)) \vee (X_2(x) \wedge Q_a(x) \wedge X_1(y)) \vee (X_2(x) \wedge Q_b(x) \wedge X_0(y))]]$$
$$\wedge \exists x [last(x) \wedge (X_1(x) \wedge Q_a(x))]$$

# MSO to Regular Languages

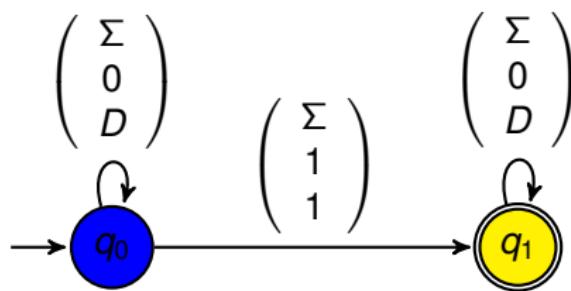
---

- ▶ Every MSO sentence  $\varphi$  over words can be converted into a DFA  $A_\varphi$  such that  $L(\varphi) = L(A_\varphi)$ .
- ▶ Start with atomic formulae, construct DFA for each of them.
- ▶ We already know how to handle  $Q_a(x)$ ,  $x < y$ ,  $S(x, y)$ ,  $x = y$
- ▶ Only  $X(x)$  remains

# Simple Formulae to DFA

---

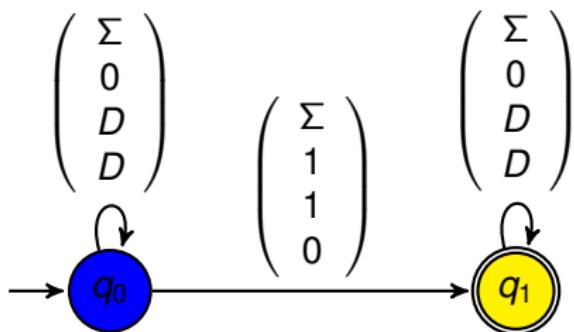
►  $X(x)$



# Simple Formulae to DFA

---

- ▶  $X(x) \wedge \neg Y(x)$
- ▶  $\Sigma' = \Sigma \times \{0, 1\} \times \{0, 1\} \times \{0, 1\}$



# Formulae to DFA

---

- Given  $\varphi(x_1, \dots, x_n, X_1, \dots, X_m)$ , an MSO formula over  $\Sigma$ , consider the extended alphabet

$$\Sigma' = \Sigma \times \{0, 1\}^{m+n}$$

- Assign values to  $x_i, X_j$  at every position as seen in the cases of atomic formulae
- Keep in mind that every  $x_i$  can be assigned 1 at a unique position

# Handling Quantifiers

---

$\exists X \exists Y \forall x [X(x) \rightarrow Y(x)]$  On the board

# Points to Remember

---

- ▶ Given  $\varphi(x_1, \dots, x_n, X_1, \dots, X_m)$ , construct automaton for atomic MSO formulae over the extended alphabet  $\Sigma \times \{0, 1\}^{m+n}$
- ▶ Intersect with the regular language where every  $x_i$  is assigned 1 exactly at one position
- ▶ Given a sentence  $Q_{x_1} \dots Q_{x_n} Q_{X_1} \dots Q_{X_m} \varphi$ , first construct the automaton for the formula  $\varphi(x_1, \dots, x_n, X_1, \dots, X_m)$
- ▶ Replace  $\forall$  in terms of  $\exists$

# Points to Remember

---

- ▶ Given the automaton for  $\varphi(x_1, \dots, x_n, X_1, \dots, X_n)$ , the automaton for  $\exists X_i \varphi(x_1, \dots, x_n, X_1, \dots, X_n)$  is obtained by **projecting out** the row of  $X_i$
- ▶ This may result in an NFA
- ▶ Determinize it and complement it to get a DFA for  $\neg \exists X_i \varphi(x_1, \dots, x_n, X_1, \dots, X_n)$
- ▶ Intersect with the regular language where each of  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$  are assigned 1 exactly at one position

# The Automaton-Logic Connection

---

## Büchi-Elgot-Trakhtenbrot Theorem (1960-1962)

Given any MSO sentence  $\varphi$ , one can construct a DFA  $A_\varphi$  such that  $L(\varphi) = L(A_\varphi)$ . If a language  $L$  is regular, one can construct an MSO sentence  $\varphi$  such that  $L = L(\varphi)$ .

# Lecture 24 Herbrand Theory.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Satisfiability of FOL

---

Given a formula in FOL over some signature  $\tau$ , is it satisfiable?

# Herbrand Theory

---

- ▶ Named after the French mathematician Jacques Herbrand
- ▶ Famous for Herbrand's Theorem, which allows a certain reduction from FOL to propositional logic
- ▶ Herbrand's theorem allows reducing a FOL formula  $\varphi$  in Skolem Normal Form to an infinite set  $E(\varphi)$  of propositional formulae s.t.  $\varphi$  is satisfiable iff  $E(\varphi)$  is satisfiable
- ▶ If  $E(\varphi)$  is not satisfiable, then  $\emptyset \in \text{Res}^*(E(\varphi))$ , and we can derive this in finite number of steps
- ▶ As  $E(\varphi)$  may be infinite, there is no way to say  $\emptyset \notin \text{Res}^*(E(\varphi))$ .

# Herbrand Universe

---

## Herbrand Universe

Let  $\tau$  be a signature. The Herbrand universe for  $\tau$  is the set of all ground terms.

- ▶ If  $\tau$  contains a constant  $c$  and unary function  $f$ , then the Herbrand universe contains  $c, f(c), f(f(c)), f(f(f(c))), \dots$
- ▶ If  $\tau$  contains a constant  $c$  and unary function  $f$  and binary function  $g$ , then the Herbrand universe contains  $c, g(c, c), f(c), g(c, f(c)), g(g(f(f(c)), c), f(c)), \dots$
- ▶ If  $\tau$  has no constants, then the Herbrand universe is empty
- ▶ If  $\tau$  has no functions, then the Herbrand universe consists of the constants of  $\tau$

# Herbrand Structures

## Herbrand Structures

A structure  $\mathcal{A}$  over a signature  $\tau$  is a Herbrand structure if its underlying universe is a Herbrand universe.

- ▶ If the signature  $\tau$  has no relations, then there is a unique Herbrand structure for  $\tau$
- ▶ If the signature  $\tau$  has relations, then there are many Herbrand structures over  $\tau$  depending on how you interpret them.
  - ▶ If  $\tau$  contains a constant  $c$ , function  $f$  and a unary relation  $R$ , then
  - ▶  $\mathcal{A} = (U^{\mathcal{A}} = \{c, f(c), f(f(c)), \dots\}, R^{\mathcal{A}} = \{c, f(c)\})$  is a Herbrand structure for  $\tau$ .
  - ▶  $\mathcal{A} = (U^{\mathcal{A}} = \{c, f(c), f(f(c)), \dots\}, R^{\mathcal{A}} = \{f(c), f(f(f(c))))\})$  is a Herbrand structure, and so on.

# Herbrand Signature

---

Let  $\Gamma$  be a set of sentences over a signature  $\tau$ .

- ▶ The Herbrand signature for  $\Gamma$  denoted  $\tau_H = \tau \cup \{c\}$  if  $\tau$  contains no constants, else it is  $\tau$ .
- ▶ The Herbrand universe for  $\Gamma$  denoted  $H(\Gamma)$  is  $\tau_H$ .

## Herbrand Model

A Herbrand model for  $\Gamma$  is a Herbrand structure  $M$  over  $\tau_H$  such that  $M \models \varphi$  for all  $\varphi \in \Gamma$ .

# FO without equality

---

Let us focus on FO without “ $=$ ”. Recall that “ $=$ ” is always interpreted as equality.

## Herbrand Theorem

Let  $\Gamma = \{\varphi_1, \varphi_2, \dots\}$  be a set of equality-free sentences in Skolem Normal Form. Then  $\Gamma$  is satisfiable iff  $\Gamma$  has a Herbrand model.

If  $\Gamma$  has a Herbrand model, clearly  $\Gamma$  is satisfiable. The converse needs a proof.

# The converse

---

Assume  $\Gamma$  is satisfiable. Let  $\tau_H$  be the Herbrand signature for  $\Gamma$ .

- ▶ Let  $\mathcal{A}$  be a  $\tau_H$  structure such that  $\mathcal{A} \models \Gamma$ . ( $U^{\mathcal{A}}$  need not be the Herbrand universe)
- ▶ Let  $\mathcal{B}$  be a Herbrand structure over  $\tau_H$ . ( $U^{\mathcal{B}}$  is the Herbrand universe)
- ▶ Try “merging”  $\mathcal{A}$  and  $\mathcal{B}$  to obtain a Herbrand model  $M$  for  $\Gamma$ .
  - ▶ Define  $M$  so that its universe is the Herbrand universe over  $\tau_H$ .
  - ▶ Let  $M$  interpret functions and constants like  $\mathcal{B}$  (both have the same Herbrand universe)
  - ▶ Let  $M$  interpret relations like  $\mathcal{A}$  (not obvious, their universes are not the same.)

# Building the Herbrand Model $M$

---

- ▶ Let  $R$  be an  $n$ -ary relation in  $\tau_H$  (hence in  $\tau$ ).
- ▶ For each  $n$ -tuple  $(t_1, \dots, t_n)$  with  $t_i$  coming from the Herbrand universe  $H(\Gamma)$ , we must say whether  $(t_1, \dots, t_n) \in R^M$  or not
- ▶ Each  $t_i \in H(\Gamma)$  is a ground term in  $\tau_H$  (so variable free).
- ▶ Since  $\mathcal{A}$  is a structure over  $\tau_H$ , we know for each  $n$ -tuple  $(t_1, \dots, t_n)$ , whether  $(t_1, \dots, t_n) \in R^{\mathcal{A}}$  or not
- ▶ Define  $R^M = R^{\mathcal{A}}$ .
- ▶ Now we prove that if  $\mathcal{A} \models \varphi$  for any  $\varphi \in \Gamma$ , then  $M \models \varphi$ .
- ▶ The proof is by induction on the number of quantifiers in  $\varphi$ .  
Recall that each  $\varphi$  is in Skolem Normal Form.

## Base case : $\varphi$ has 0 quantifiers

---

$\mathcal{A} \models \varphi$  iff  $M \models \varphi$ . Do structural induction on  $\varphi$ .

- ▶ Assume  $\varphi$  is an atomic formula. Then  $\varphi$  is  $R(t_1, \dots, t_n)$  where  $R$  is an  $n$ -ary relation from  $\tau_H$ , and  $t_1, \dots, t_n$  are all terms from  $H(\Gamma)$ .
- ▶ By the construction of  $M$ ,  $R^M = R^{\mathcal{A}}$ .
- ▶ Hence  $M \models \varphi$  iff  $\mathcal{A} \models \varphi$ .
- ▶ Same reasoning holds for  $\varphi_1 \wedge \varphi_2$ ,  $\varphi_1 \wedge \varphi_2$  and  $\neg\varphi$ .
- ▶ Hence,  $\mathcal{A} \models \varphi$  iff  $M \models \varphi$ .

# Post Inductive Hypothesis

---

Assume that for any  $\psi \in \Gamma$  with  $\leq k - 1$  quantifiers, if  $\mathcal{A} \models \psi$ , then  $M \models \psi$ . Let  $\varphi \in \Gamma$  have  $k$  quantifiers,  $\forall x_1 \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$  where  $\zeta$  is quantifier free.

- ▶ Let  $\kappa(x_1) = \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$ , and  $\varphi = \forall x_1 \kappa(x_1)$ .
- ▶  $\mathcal{A} \models \varphi$  implies  $\mathcal{A} \models \forall x_1 \kappa(x_1)$ . That is,  $\mathcal{A} \models \kappa(a)$  for any  $a \in U^{\mathcal{A}}$ .
- ▶ Since  $\mathcal{A}$  is a structure over  $\tau_H$ , if  $t \in H(\Gamma)$  is a ground term from  $\tau_H$ ,  $\mathcal{A}$  interprets  $t$  as an element of  $U^{\mathcal{A}}$ .
- ▶ Thus,  $\mathcal{A} \models \kappa(t)$  for any  $t \in H(\Gamma)$ .
- ▶ By induction hypothesis,  $M \models \kappa(t)$  for any  $t \in H(\Gamma)$ .
- ▶ Since  $H(\Gamma)$  is the universe of  $M$ ,  $M \models \forall x_1 \kappa(x_1)$ . That is,  $M \models \varphi$ .

# Dealing with Equality

---

Assume  $\varphi$  is in Skolem Normal Form and uses “ $=$ ”. We define a equisatisfiable formula  $\varphi_E$  which does not use “ $=$ ”.

- ▶ Let  $\tau$  be the signature of  $\varphi$ . Let  $E$  be a binary relation not in  $\tau$ .
  - ▶ Let  $\varphi_{\neq}$  be the sentence obtained by replacing all occurrences of  $t_1 = t_2$  in  $\varphi$  with  $E(t_1, t_2)$ .
  - ▶ Define  $\varphi_{ER}$  to be the sentence
- $$\forall x \forall y \forall z (E(x, x) \wedge ((E(x, y) \leftrightarrow E(y, x)) \wedge (E(x, y) \wedge E(y, z) \rightarrow E(x, z))))$$
- ▶ For each relation  $R$  in  $\tau$ , define  $\varphi_R$  as

$$\forall x_1 \dots \forall x_n \forall y_1 \dots \forall y_n ((\bigwedge_{i=1}^n E(x_i, y_i) \wedge R(x_1, \dots, x_n)) \rightarrow R(y_1, \dots, y_n))$$

- ▶ Let  $\varphi_1 = \bigwedge_{R \in \tau} \varphi_R$

# Dealing with Equality

- ▶ For each function  $f$  in  $\tau$ , define  $\varphi_f$  as

$$\forall x_1 \dots \forall x_n \forall y_1 \dots \forall y_n ((\bigwedge_{i=1}^n E(x_i, y_i) \rightarrow E(f(x_1, \dots, x_n), f(y_1, \dots, y_n)))$$

- ▶ Let  $\varphi_2 = \bigwedge_{f \in \tau} \varphi_f$
- ▶ Let  $\psi_E = \varphi_{\neq} \wedge \varphi_{ER} \wedge \varphi_1 \wedge \varphi_2$
- ▶ Convert  $\psi_E$  to Prenex normal form to obtain  $\varphi_E$  in Skolem normal form

For any formula  $\varphi$  in Skolem normal form,  $\varphi$  is satisfiable iff  $\varphi_E$  is satisfiable

# An Example

---

$$\varphi = \forall x[(f(x) \neq x) \wedge (f(f(x)) = x)].$$

- ▶  $\varphi$  is satisfiable :  $\mathcal{A} = (\{c, d\}, f^{\mathcal{A}}(c) = d, f^{\mathcal{A}}(d) = c)$  and  $\mathcal{A} \models \varphi$ .
- ▶  $\varphi_{\neq} = \forall x[\neg E(f(x), x) \wedge E(f(f(x)), x)]$
- ▶  $\varphi_2 = \forall x \forall y [E(x, y) \rightarrow E(f(x), f(y))]$
- ▶ Conjunct  $\varphi_{\neq}$ ,  $\varphi_2$  and  $\varphi_{ER}$  and convert to Prenex normal form
- ▶  $\varphi_E = \forall x \forall y \forall z [(\neg E(f(x), x) \wedge E(f(f(x)), x)) \wedge (E(x, y) \rightarrow E(f(x), f(y))) \wedge E(x, x) \wedge (E(x, y) \wedge E(y, z) \rightarrow E(x, z))]$
- ▶ By Herbrand's Theorem,  $\varphi_E$  has a Herbrand model  
 $M = (\{c, f(c), f(f(c)), \dots\}, E^M = \{(t, t') \in H(\varphi_E) \mid \text{the number of } f\text{'s in both } t, t' \text{ have the same parity}\})$
- ▶  $M \models \varphi_E$

# Herbrand's Method

---

Given a FO sentence  $\varphi$ , is it satisfiable? Wlg, assume that  $\varphi$  is equality-free and is in Skolem normal form.

- ▶ Let  $\varphi = \forall x_1 \dots \forall x_n \psi(x_1, \dots, x_n)$
- ▶ Let  $H(\varphi)$  be the Herbrand universe of  $\varphi$
- ▶ Let  $E(\varphi) = \{\psi(t_1, \dots, t_n) \mid t_1, \dots, t_n \in H(\varphi)\}$  be the set obtained by substituting terms from  $H(\varphi)$  for the variables  $x_1, \dots, x_n$  in  $\varphi$
- ▶  $\varphi$  is satisfiable iff  $E(\varphi)$  is satisfiable

# Herbrand's Method

---

- ▶ Assume  $\varphi$  is satisfiable. Then  $\mathcal{A} \models \forall x_1, \dots, x_n \psi(x_1, \dots, x_n)$
- ▶ Then  $\mathcal{A} \models \psi(t_1, \dots, t_n)$  where  $t_1, \dots, t_n \in H(\varphi)$
- ▶ Then  $\mathcal{A} \models \varphi_i$  for all  $\varphi_i \in E(\varphi)$
- ▶ Hence,  $E(\varphi)$  is satisfiable.

# Herbrand's Method

---

- ▶ Assume  $E(\varphi)$  is satisfiable. Then by Herbrand's Theorem, there is a Herbrand model  $M$  for  $E(\varphi)$ .
- ▶ The Herbrand signature for  $E(\varphi)$  is the same as the Herbrand signature of  $\varphi$ .
- ▶ The universe of  $M$  is  $H(\varphi)$
- ▶ For  $t_1, \dots, t_n \in H(\varphi)$ ,  $M \models \psi(t_1, \dots, t_n)$
- ▶ Then  $M \models \forall x_1 \dots x_n \psi(x_1, \dots, x_n)$
- ▶ Then  $M \models \varphi$  and  $\varphi$  is satisfiable.
- ▶  $\varphi$  is unsatisfiable iff  $E(\varphi)$  is unsatisfiable.

# Checking Unsatisfiability of $\varphi$

---

- ▶  $E(\varphi) = \{\varphi_1, \varphi_2, \dots\}$  is a set of quantifier free sentences, so it can be seen as a set of propositional logic formulae
- ▶ Since  $\varphi$  is in Skolem normal form, each formula  $\varphi_i \in E(\varphi)$  is in CNF
- ▶ We know that  $E(\varphi)$  is unsatisfiable iff  $\emptyset \in \text{Res}^*(E(\varphi))$
- ▶ That is, there is some finite subset  $F = \{\varphi_1, \dots, \varphi_m\} \subseteq E(\varphi)$  such that  $\emptyset \in \text{Res}^*(F)$
- ▶ So if  $\emptyset \in \text{Res}^*(\{\varphi_1, \dots, \varphi_m\})$  for some finite  $m$ , we conclude  $\varphi$  is unsatisfiable

# Checking Satisfiability of $\varphi$

---

- ▶ If  $\emptyset \notin \text{Res}^*(\{\varphi_1, \dots, \varphi_m\})$ , then we look at  
 $\text{Res}^*(\{\varphi_1, \dots, \varphi_m, \varphi_{m+1}\})$
- ▶ If  $\emptyset \notin \text{Res}^*(\{\varphi_1, \dots, \varphi_{m+1}\})$ , then we look at  
 $\text{Res}^*(\{\varphi_1, \dots, \varphi_{m+1}, \varphi_{m+2}\})$
- ⋮
- ▶ If  $\varphi$  is satisfiable, then this procedure will continue.

# Wrapping Up

---

- ▶ We have a method to show that a FOL formula  $\varphi$  is unsatisfiable
- ▶ First, write  $\varphi$  in equality free Skolem normal form
- ▶ Check if  $\emptyset \in \text{Res}^*(E(\varphi))$ , this may take some time
- ▶ There is a more systematic resolution for FOL which we do not cover (this also uses Herbrand Theory)
- ▶ We also do not cover a direct undecidability proof for the satisfiability of FOL (at least now)

# Lecture 25 Herbrand Theorem.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# FO without equality

---

Let us focus on FO without “ $=$ ”. Recall that “ $=$ ” is always interpreted as equality.

## Herbrand Theorem

Let  $\Gamma = \{\varphi_1, \varphi_2, \dots\}$  be a set of equality-free sentences in Skolem Normal Form. Then  $\Gamma$  is satisfiable iff  $\Gamma$  has a Herbrand model.

If  $\Gamma$  has a Herbrand model, clearly  $\Gamma$  is satisfiable. The converse needs a proof.

# The converse

---

Assume  $\Gamma$  is satisfiable. Let  $\tau_H$  be the Herbrand signature for  $\Gamma$ .

- ▶ Let  $\mathcal{A}$  be a  $\tau_H$  structure such that  $\mathcal{A} \models \Gamma$ . ( $U^{\mathcal{A}}$  need not be the Herbrand universe)
- ▶ Let  $\mathcal{B}$  be a Herbrand structure over  $\tau_H$ . ( $U^{\mathcal{B}}$  is the Herbrand universe)

# The converse

---

Assume  $\Gamma$  is satisfiable. Let  $\tau_H$  be the Herbrand signature for  $\Gamma$ .

- ▶ Let  $\mathcal{A}$  be a  $\tau_H$  structure such that  $\mathcal{A} \models \Gamma$ . ( $U^{\mathcal{A}}$  need not be the Herbrand universe)
- ▶ Let  $\mathcal{B}$  be a Herbrand structure over  $\tau_H$ . ( $U^{\mathcal{B}}$  is the Herbrand universe)
- ▶ We need to create a Herbrand model for  $\Gamma$

# The converse

---

Assume  $\Gamma$  is satisfiable. Let  $\tau_H$  be the Herbrand signature for  $\Gamma$ .

- ▶ Let  $\mathcal{A}$  be a  $\tau_H$  structure such that  $\mathcal{A} \models \Gamma$ . ( $U^{\mathcal{A}}$  need not be the Herbrand universe)
- ▶ Let  $\mathcal{B}$  be a Herbrand structure over  $\tau_H$ . ( $U^{\mathcal{B}}$  is the Herbrand universe)
- ▶ We need to create a Herbrand model for  $\Gamma$
- ▶ Try “merging”  $\mathcal{A}$  and  $\mathcal{B}$  to obtain a Herbrand model  $M$  for  $\Gamma$ .

# The converse

---

Assume  $\Gamma$  is satisfiable. Let  $\tau_H$  be the Herbrand signature for  $\Gamma$ .

- ▶ Let  $\mathcal{A}$  be a  $\tau_H$  structure such that  $\mathcal{A} \models \Gamma$ . ( $U^{\mathcal{A}}$  need not be the Herbrand universe)
- ▶ Let  $\mathcal{B}$  be a Herbrand structure over  $\tau_H$ . ( $U^{\mathcal{B}}$  is the Herbrand universe)
- ▶ We need to create a Herbrand model for  $\Gamma$
- ▶ Try “merging”  $\mathcal{A}$  and  $\mathcal{B}$  to obtain a Herbrand model  $M$  for  $\Gamma$ .
  - ▶ Define  $M$  so that its universe is the Herbrand universe over  $\tau_H$ .
  - ▶ Let  $M$  interpret functions and constants like  $\mathcal{B}$  (both have the same Herbrand universe)
  - ▶ How to interpret the relations from  $\tau_H$  (same as those of  $\tau$ )?

# The converse

---

Assume  $\Gamma$  is satisfiable. Let  $\tau_H$  be the Herbrand signature for  $\Gamma$ .

- ▶ Let  $\mathcal{A}$  be a  $\tau_H$  structure such that  $\mathcal{A} \models \Gamma$ . ( $U^{\mathcal{A}}$  need not be the Herbrand universe)
- ▶ Let  $\mathcal{B}$  be a Herbrand structure over  $\tau_H$ . ( $U^{\mathcal{B}}$  is the Herbrand universe)
- ▶ We need to create a Herbrand model for  $\Gamma$
- ▶ Try “merging”  $\mathcal{A}$  and  $\mathcal{B}$  to obtain a Herbrand model  $M$  for  $\Gamma$ .
  - ▶ Define  $M$  so that its universe is the Herbrand universe over  $\tau_H$ .
  - ▶ Let  $M$  interpret functions and constants like  $\mathcal{B}$  (both have the same Herbrand universe)
  - ▶ How to interpret the relations from  $\tau_H$  (same as those of  $\tau$ )?
  - ▶ Let  $M$  interpret relations like  $\mathcal{A}$  (not obvious, their universes are not the same.)

# Building the Herbrand Model $M$

---

- ▶ Let  $R$  be an  $n$ -ary relation in  $\tau_H$  (hence in  $\tau$ ).
- ▶ For each  $n$ -tuple  $(t_1, \dots, t_n)$  with  $t_i$  coming from the Herbrand universe  $H(\Gamma)$ , we must say whether  $(t_1, \dots, t_n) \in R^M$  or not
- ▶ Each  $t_i \in H(\Gamma)$  is a ground term in  $\tau_H$  (so variable free).
- ▶ Since  $\mathcal{A}$  is a structure over  $\tau_H$ , if  $t \in H(\Gamma)$  is a ground term from  $\tau_H$ ,  $\mathcal{A}$  interprets  $t$  as an element of  $U^{\mathcal{A}}$ .
- ▶ For each  $n$ -tuple  $(t_1, \dots, t_n)$ ,  $t_i \in H(\Gamma)$ , we know whether  $(t_1, \dots, t_n) \in R^{\mathcal{A}}$  or not
- ▶ Define  $R^M = R^{\mathcal{A}}$ .
- ▶ Prove that if  $\mathcal{A} \models \varphi$  for any  $\varphi \in \Gamma$ , then  $M \models \varphi$ .
- ▶ The proof is by induction on the number of quantifiers in  $\varphi$ .  
Recall that each  $\varphi$  is in Skolem Normal Form.

## Base case : $\varphi$ has 0 quantifiers

---

$\mathcal{A} \models \varphi$  iff  $M \models \varphi$ . Do structural induction on  $\varphi$ .

- ▶ Assume  $\varphi$  is an atomic formula. Then  $\varphi$  is  $R(t_1, \dots, t_n)$  where  $R$  is an  $n$ -ary relation from  $\tau_H$ , and  $t_1, \dots, t_n$  are all terms from  $H(\Gamma)$ .
- ▶ By the construction of  $M$ ,  $R^M = R^{\mathcal{A}}$ .
- ▶ Hence  $M \models \varphi$  iff  $\mathcal{A} \models \varphi$ .
- ▶ Same reasoning holds for  $\varphi_1 \wedge \varphi_2$ ,  $\varphi_1 \vee \varphi_2$  and  $\neg\varphi$ .
- ▶ Hence,  $\mathcal{A} \models \varphi$  iff  $M \models \varphi$ .

# Post Inductive Hypothesis

---

Assume that for any  $\psi \in \Gamma$  with  $\leq k - 1$  quantifiers, if  $\mathcal{A} \models \psi$ , then  $M \models \psi$ . Let  $\varphi \in \Gamma$  have  $k$  quantifiers,  $\varphi = \forall x_1 \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$  where  $\zeta$  is quantifier free.

# Post Inductive Hypothesis

---

Assume that for any  $\psi \in \Gamma$  with  $\leq k - 1$  quantifiers, if  $\mathcal{A} \models \psi$ , then  $M \models \psi$ . Let  $\varphi \in \Gamma$  have  $k$  quantifiers,  $\varphi = \forall x_1 \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$  where  $\zeta$  is quantifier free.

- ▶ Let  $\kappa(x_1) = \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$ , and  $\varphi = \forall x_1 \kappa(x_1)$ .

# Post Inductive Hypothesis

---

Assume that for any  $\psi \in \Gamma$  with  $\leq k - 1$  quantifiers, if  $\mathcal{A} \models \psi$ , then  $M \models \psi$ . Let  $\varphi \in \Gamma$  have  $k$  quantifiers,  $\varphi = \forall x_1 \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$  where  $\zeta$  is quantifier free.

- ▶ Let  $\kappa(x_1) = \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$ , and  $\varphi = \forall x_1 \kappa(x_1)$ .
- ▶  $\mathcal{A} \models \varphi$  implies  $\mathcal{A} \models \forall x_1 \kappa(x_1)$ . That is,  $\mathcal{A} \models \kappa(a)$  for any  $a \in U^{\mathcal{A}}$ .

# Post Inductive Hypothesis

---

Assume that for any  $\psi \in \Gamma$  with  $\leq k - 1$  quantifiers, if  $\mathcal{A} \models \psi$ , then  $M \models \psi$ . Let  $\varphi \in \Gamma$  have  $k$  quantifiers,  $\varphi = \forall x_1 \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$  where  $\zeta$  is quantifier free.

- ▶ Let  $\kappa(x_1) = \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$ , and  $\varphi = \forall x_1 \kappa(x_1)$ .
- ▶  $\mathcal{A} \models \varphi$  implies  $\mathcal{A} \models \forall x_1 \kappa(x_1)$ . That is,  $\mathcal{A} \models \kappa(a)$  for any  $a \in U^{\mathcal{A}}$ .
- ▶ Since  $\mathcal{A}$  is a structure over  $\tau_H$ , if  $t \in H(\Gamma)$  is a ground term from  $\tau_H$ ,  $\mathcal{A}$  interprets  $t$  as an element of  $U^{\mathcal{A}}$ .
- ▶ Thus,  $\mathcal{A} \models \kappa(t)$  for any  $t \in H(\Gamma)$ .

# Post Inductive Hypothesis

---

Assume that for any  $\psi \in \Gamma$  with  $\leq k - 1$  quantifiers, if  $\mathcal{A} \models \psi$ , then  $M \models \psi$ . Let  $\varphi \in \Gamma$  have  $k$  quantifiers,  $\varphi = \forall x_1 \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$  where  $\zeta$  is quantifier free.

- ▶ Let  $\kappa(x_1) = \forall x_2 \dots \forall x_k \zeta(x_1, \dots, x_k)$ , and  $\varphi = \forall x_1 \kappa(x_1)$ .
- ▶  $\mathcal{A} \models \varphi$  implies  $\mathcal{A} \models \forall x_1 \kappa(x_1)$ . That is,  $\mathcal{A} \models \kappa(a)$  for any  $a \in U^{\mathcal{A}}$ .
- ▶ Since  $\mathcal{A}$  is a structure over  $\tau_H$ , if  $t \in H(\Gamma)$  is a ground term from  $\tau_H$ ,  $\mathcal{A}$  interprets  $t$  as an element of  $U^{\mathcal{A}}$ .
- ▶ Thus,  $\mathcal{A} \models \kappa(t)$  for any  $t \in H(\Gamma)$ .
- ▶ By induction hypothesis,  $M \models \kappa(t)$  for any  $t \in H(\Gamma)$ .
- ▶ Since  $H(\Gamma)$  is the universe of  $M$ ,  $M \models \forall x_1 \kappa(x_1)$ . That is,  $M \models \varphi$ .

# Equality

---

$$\varphi = \forall x[(f(x) \neq x) \wedge (f(f(x)) = x)].$$

# Equality

---

$$\varphi = \forall x[(f(x) \neq x) \wedge (f(f(x)) = x)].$$

- ▶ Herbrand universe over  $\tau_H = \{c, f\}$  is  $\{c, f(c), f(f(c)), \dots\}$  all distinct
- ▶ Then  $f(f(c)) \neq c$ , and a Herbrand structure cannot satisfy  $\varphi$

# Equality

---

$$\varphi = \forall x[(f(x) \neq x) \wedge (f(f(x)) = x)].$$

- ▶ Herbrand universe over  $\tau_H = \{c, f\}$  is  $\{c, f(c), f(f(c)), \dots\}$  all distinct
- ▶ Then  $f(f(c)) \neq c$ , and a Herbrand structure cannot satisfy  $\varphi$
- ▶ However,  $\varphi$  is satisfiable. Define a structure  $\mathcal{A} = (\{0, 1\}, f^{\mathcal{A}}(0) = 1, f^{\mathcal{A}}(1) = 0)$ ,  $\mathcal{A} \models \varphi$
- ▶ For formulae which have equality, Herbrand's Theorem does not apply directly
- ▶ If  $\varphi$  has equality, convert it to an equisatisfiable sentence without equality and apply Herbrand

# Equality

$$\varphi = \forall x[(f(x) \neq x) \wedge (f(f(x)) = x)].$$

- ▶ Herbrand universe over  $\tau_H = \{c, f\}$  is  $\{c, f(c), f(f(c)), \dots\}$  all distinct
- ▶ Then  $f(f(c)) \neq c$ , and a Herbrand structure cannot satisfy  $\varphi$
- ▶ However,  $\varphi$  is satisfiable. Define a structure  $\mathcal{A} = (\{0, 1\}, f^{\mathcal{A}}(0) = 1, f^{\mathcal{A}}(1) = 0)$ ,  $\mathcal{A} \models \varphi$
- ▶ For formulae which have equality, Herbrand's Theorem does not apply directly
- ▶ If  $\varphi$  has equality, convert it to an equisatisfiable sentence without equality and apply Herbrand

Let  $\varphi$  be in Skolem normal form with equality. Then  $\varphi$  is satisfiable iff there is an equisatisfiable formula  $\psi$  in Skolem normal form without equality which has a Herbrand model.

# Lecture 26 Herbrand Method.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Dealing with Equality

Assume  $\varphi$  is in Skolem Normal Form and uses “ $=$ ”. We define a equisatisfiable formula  $\varphi_E$  which does not use “ $=$ ”.

- ▶ Let  $\tau$  be the signature of  $\varphi$ . Let  $E$  be a binary relation not in  $\tau$ .
  - ▶ Let  $\varphi_{\neq}$  be the sentence obtained by replacing all occurrences of  $t_1 = t_2$  in  $\varphi$  with  $E(t_1, t_2)$ .
  - ▶ Define  $\varphi_{ER}$  to be the sentence
- $$\forall x \forall y \forall z (E(x, x) \wedge ((E(x, y) \leftrightarrow E(y, x)) \wedge (E(x, y) \wedge E(y, z) \rightarrow E(x, z))))$$
- ▶ For each relation  $R$  in  $\tau$ , define  $\varphi_R$  as

$$\forall x_1 \dots \forall x_n \forall y_1 \dots \forall y_n ((\bigwedge_{i=1}^n E(x_i, y_i) \wedge R(x_1, \dots, x_n)) \rightarrow R(y_1, \dots, y_n))$$

- ▶ Let  $\varphi_1 = \bigwedge_{R \in \tau} \varphi_R$

# Dealing with Equality

- ▶ For each function  $f$  in  $\tau$ , define  $\varphi_f$  as

$$\forall x_1 \dots \forall x_n \forall y_1 \dots \forall y_n ((\bigwedge_{i=1}^n E(x_i, y_i) \rightarrow E(f(x_1, \dots, x_n), f(y_1, \dots, y_n)))$$

- ▶ Let  $\varphi_2 = \bigwedge_{f \in \tau} \varphi_f$
- ▶ Let  $\psi_E = \varphi_{\neq} \wedge \varphi_{ER} \wedge \varphi_1 \wedge \varphi_2$
- ▶ Convert  $\psi_E$  to Prenex normal form to obtain  $\varphi_E$  in Skolem normal form

For any formula  $\varphi$  in Skolem normal form,  $\varphi$  is satisfiable iff  $\varphi_E$  is satisfiable

# An Example

---

$$\varphi = \forall x[(f(x) \neq x) \wedge (f(f(x)) = x)].$$

- ▶  $\varphi$  is satisfiable :  $\mathcal{A} = (\{0, 1\}, f^{\mathcal{A}}(0) = 1, f^{\mathcal{A}}(1) = 0)$  and  $\mathcal{A} \models \varphi$ .
- ▶  $\varphi_{\neq} = \forall x[\neg E(f(x), x) \wedge E(f(f(x)), x)]$
- ▶  $\varphi_2 = \forall x \forall y [E(x, y) \rightarrow E(f(x), f(y))]$
- ▶ Conjunct  $\varphi_{\neq}$ ,  $\varphi_2$  and  $\varphi_{ER}$  and convert to Prenex normal form
- ▶  $\varphi_E = \forall x \forall y \forall z [(\neg E(f(x), x) \wedge E(f(f(x)), x)) \wedge (E(x, y) \rightarrow E(f(x), f(y))) \wedge E(x, x) \wedge (E(x, y) \wedge E(y, z) \rightarrow E(x, z))]$
- ▶ By Herbrand's Theorem,  $\varphi_E$  has a Herbrand model  
 $M = (\{c, f(c), f(f(c)), \dots, \}, E^M = \{(t, t') \in H(\varphi_E) \mid \text{the number of } f\text{'s in both } t, t' \text{ have the same parity}\})$
- ▶  $M \models \varphi_E$

# Herbrand's Method

---

Given a FO sentence  $\varphi$ , is it satisfiable? Wlg, assume that  $\varphi$  is equality-free and is in Skolem normal form.

- ▶ Let  $\varphi = \forall x_1 \dots \forall x_n \psi(x_1, \dots, x_n)$
- ▶ Let  $H(\varphi)$  be the Herbrand universe of  $\varphi$
- ▶ Let  $E(\varphi) = \{\psi(t_1, \dots, t_n) \mid t_1, \dots, t_n \in H(\varphi)\}$  be the set obtained by substituting terms from  $H(\varphi)$  for the variables  $x_1, \dots, x_n$  in  $\varphi$
- ▶  $\varphi$  is satisfiable iff  $E(\varphi)$  is satisfiable

# Herbrand's Method

---

- ▶ Assume  $\varphi$  is satisfiable. Then  $\mathcal{A} \models \forall x_1, \dots, x_n \psi(x_1, \dots, x_n)$
- ▶ Then  $\mathcal{A} \models \psi(t_1, \dots, t_n)$  where  $t_1, \dots, t_n \in H(\varphi)$
- ▶ Then  $\mathcal{A} \models \varphi_i$  for all  $\varphi_i \in E(\varphi)$
- ▶ Hence,  $E(\varphi)$  is satisfiable.

# Herbrand's Method

---

- ▶ Assume  $E(\varphi)$  is satisfiable.  $E(\varphi)$  is a set of equality-free sentences.
- ▶ By Herbrand's Theorem, there is a Herbrand model  $M$  for  $E(\varphi)$ .
- ▶ The Herbrand signature for  $E(\varphi)$  is the same as the Herbrand signature of  $\varphi$ .
- ▶ The universe of  $M$  is  $H(\varphi)$ . For  $t_1, \dots, t_n \in H(\varphi)$ ,  
 $M \models \psi(t_1, \dots, t_n)$
- ▶ Then  $M \models \forall x_1 \dots x_n \psi(x_1, \dots, x_n)$
- ▶ Then  $M \models \varphi$  and  $\varphi$  is satisfiable.
- ▶  $\varphi$  is unsatisfiable iff  $E(\varphi)$  is unsatisfiable.

# Checking Unsatisfiability of $\varphi$

---

- ▶  $E(\varphi) = \{\varphi_1, \varphi_2, \dots\}$  is a set of quantifier free sentences, so it can be seen as a set of propositional logic formulae

# Checking Unsatisfiability of $\varphi$

---

- ▶  $E(\varphi) = \{\varphi_1, \varphi_2, \dots\}$  is a set of quantifier free sentences, so it can be seen as a set of propositional logic formulae
- ▶ Since  $\varphi$  is in Skolem normal form, each formula  $\varphi_i \in E(\varphi)$  is in CNF
- ▶ We know that  $E(\varphi)$  is unsatisfiable iff  $\emptyset \in \text{Res}^*(E(\varphi))$
- ▶ By Compactness Theorem of propositional logic, there is some finite subset  $F = \{\varphi_1, \dots, \varphi_m\} \subseteq E(\varphi)$  such that  $\emptyset \in \text{Res}^*(F)$
- ▶ So if  $\emptyset \in \text{Res}^*(\{\varphi_1, \dots, \varphi_m\})$  for some finite  $m$ , we conclude  $\varphi$  is unsatisfiable

# Checking Satisfiability of $\varphi$

---

- ▶ If  $\emptyset \notin \text{Res}^*(\{\varphi_1, \dots, \varphi_m\})$ , then we look at  
 $\text{Res}^*(\{\varphi_1, \dots, \varphi_m, \varphi_{m+1}\})$
- ▶ If  $\emptyset \notin \text{Res}^*(\{\varphi_1, \dots, \varphi_{m+1}\})$ , then we look at  
 $\text{Res}^*(\{\varphi_1, \dots, \varphi_{m+1}, \varphi_{m+2}\})$
- ⋮
- ▶ If  $\varphi$  is satisfiable, then this procedure will continue.

# Wrapping Up

---

- ▶ We have a method to show that a FOL formula  $\varphi$  is unsatisfiable
- ▶ First, write  $\varphi$  in equality free Skolem normal form
- ▶ Check if  $\emptyset \in \text{Res}^*(E(\varphi))$ , this may take some time
- ▶ There is a more systematic resolution for FOL which we do not cover (this also uses Herbrand Theory)
- ▶ We also do not cover a direct undecidability proof for the satisfiability of FOL (at least now)

# Lecture 27-29 LTL Transitions DBA and NBA.pdf

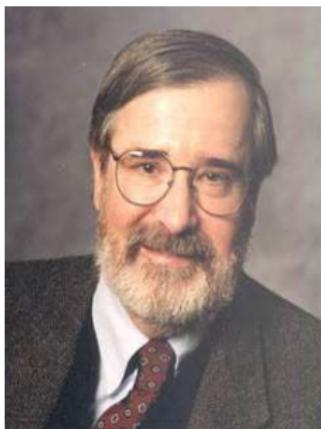
# CS 228 : Logic in Computer Science

Krishna. S

## Linear Temporal Logic

# Model Checking

---



- ▶ Year 2007 : ACM confers the **Turing Award** to the pioneers of Model Checking: **Ed Clarke, Allen Emerson, and Joseph Sifakis**
- ▶ [https://amturing.acm.org/award\\_winners/clarke\\_1167964](https://amturing.acm.org/award_winners/clarke_1167964)

# Model checking

---

- ▶ Model checking has evolved in last 25 years into a widely used verification and debugging technique for software and hardware.
- ▶ 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



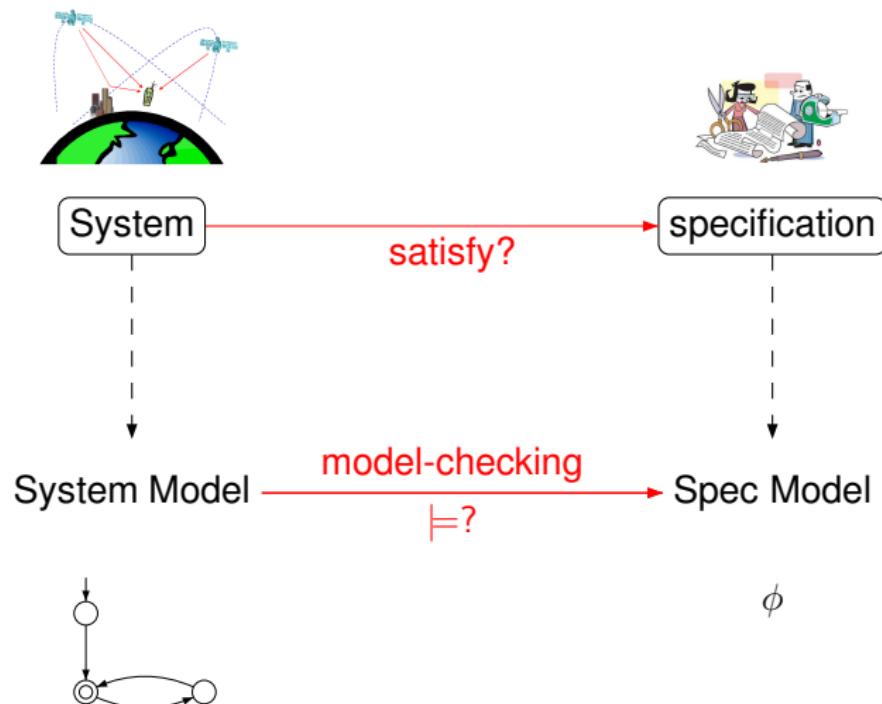
satisfy?



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 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?

# Notations for Infinite Words

---

- ▶  $\Sigma$  is a finite alphabet
- ▶  $\Sigma^*$  set of finite words over  $\Sigma$
- ▶ An infinite word is written as  $\alpha = \alpha(0)\alpha(1)\alpha(2)\dots$ , where  $\alpha(i) \in \Sigma$
- ▶ Such words are called  $\omega$ -words
- ▶  $a^\omega, a^7.b^\omega$

# Transition Systems

---

A **Transition System** is a tuple  $(S, Act, \rightarrow, I, AP, L)$  where

- ▶  $S$  is a set of **states**
- ▶  $Act$  is a set of **actions**
- ▶  $s \xrightarrow{\alpha} s'$  in  $S \times Act \times S$  is the **transition relation**
- ▶  $I \subseteq S$  is the **set of initial states**
- ▶  $AP$  is the set of **atomic propositions**
- ▶  $L : S \rightarrow 2^{AP}$  is the **labeling function**

# Traces of Transition Systems

---

- ▶ Labels of the locations represent values of all observable propositions  $\in AP$
- ▶ Captures system state
- ▶ Focus on sequences  $L(s_0)L(s_1)\dots$  of labels of locations
- ▶ Such sequences are called **traces**
- ▶ Assuming transition systems have no terminal states,
  - ▶ Traces are infinite words over  $2^{AP}$
  - ▶ Traces  $\in (2^{AP})^\omega$
  - ▶ Go to the example slide and define traces

# Traces of Transition Systems

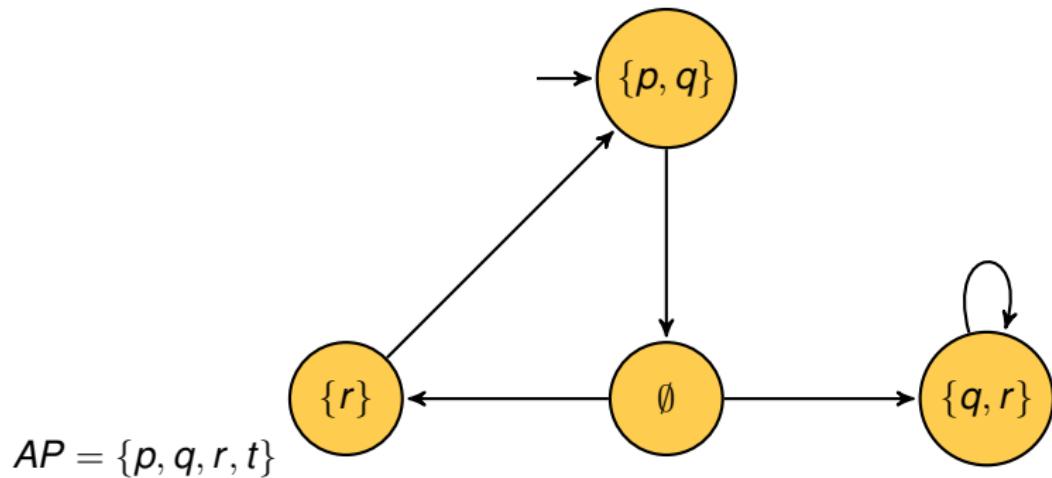
---

Given a transition system  $TS = (S, Act, \rightarrow, I, AP, L)$  **without terminal states**,

- ▶ All maximal executions/paths are infinite
- ▶ Path  $\pi = s_0 s_1 s_2 \dots$ ,  $trace(\pi) = L(s_0)L(s_1)\dots$
- ▶ For a set  $\Pi$  of paths,  $Trace(\Pi) = \{trace(\pi) \mid \pi \in \Pi\}$
- ▶ For a location  $s$ ,  $Traces(s) = Trace(Paths(s))$
- ▶  $Traces(TS) = \bigcup_{s \in I} Traces(s)$

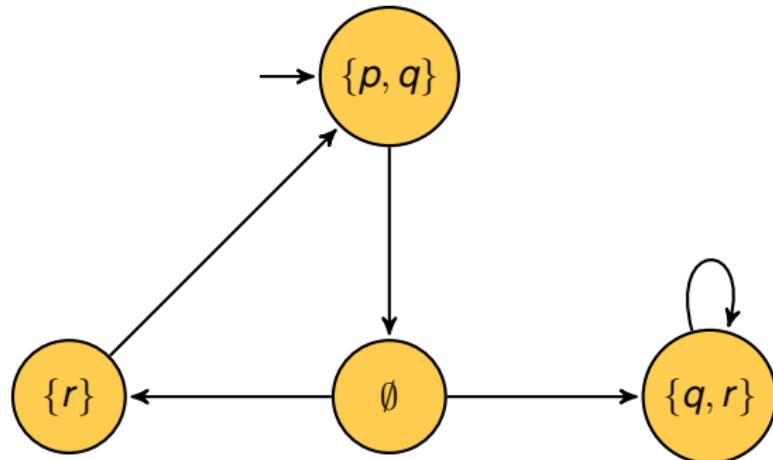
# Example Traces

---



# Example Traces

---

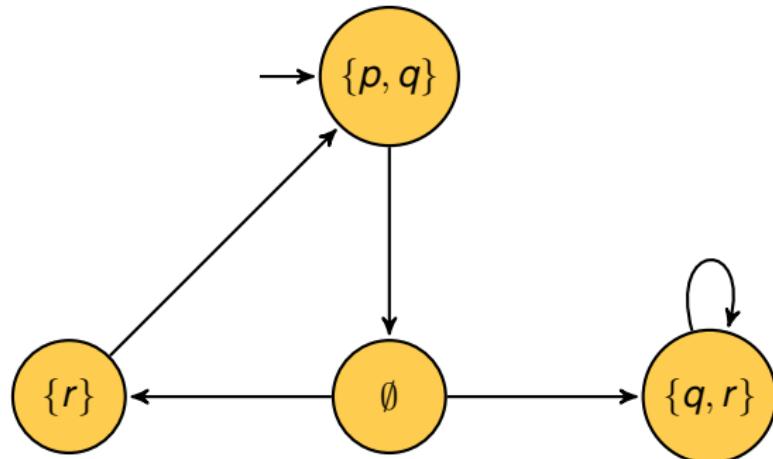


$$AP = \{p, q, r, t\}$$

►  $\{p, q\} \emptyset \{q, r\}^\omega$

# Example Traces

---

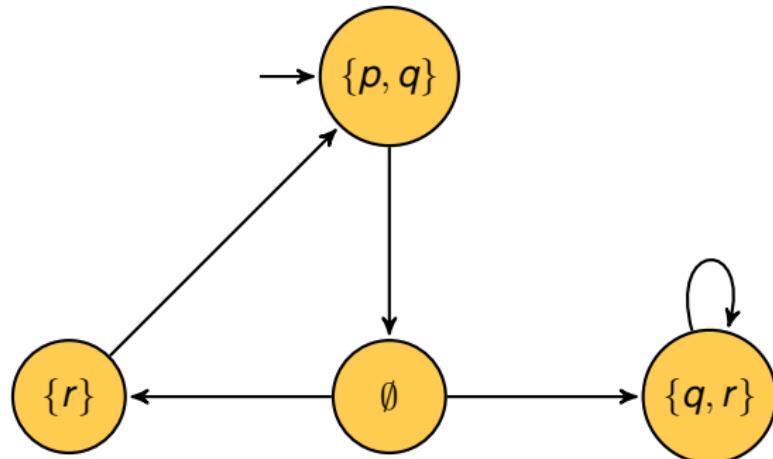


$$AP = \{p, q, r, t\}$$

- ▶  $\{p, q\} \emptyset \{q, r\}^\omega$
- ▶  $(\{p, q\} \emptyset \{r\})^\omega$

# Example Traces

---



$AP = \{p, q, r, t\}$

- ▶  $\{p, q\} \emptyset \{q, r\}^\omega$
- ▶  $(\{p, q\} \emptyset \{r\})^\omega$
- ▶  $(\{p, q\} \emptyset \{r\})^* \{p, q\} \emptyset \{q, r\}^\omega$

# Linear Time Properties

---

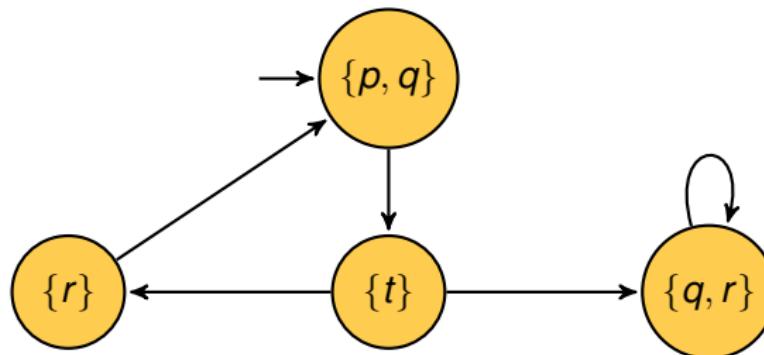
- ▶ Linear-time properties specify traces that a  $TS$  must have
- ▶ A LT property  $P$  over  $AP$  is a subset of  $(2^{AP})^\omega$
- ▶  $TS$  over  $AP$  satisfies a LT property  $P$  over  $AP$

$$TS \models P \text{ iff } \text{Traces}(TS) \subseteq P$$

- ▶  $s \in S$  satisfies LT property  $P$  (denoted  $s \models P$ ) iff  $\text{Traces}(s) \subseteq P$

# Specifying Traces

---



- ▶ Whenever  $p$  is true,  $r$  will eventually become true
  - ▶  $\{A_0 A_1 A_2 \dots \mid \forall i \geq 0, p \in A_i \rightarrow \exists j \geq i, r \in A_j\}$
- ▶  $q$  is true infinitely often
  - ▶  $\{A_0 A_1 A_2 \dots \mid \forall i \geq 0, \exists j \geq i, q \in A_j\}$
- ▶ Whenever  $r$  is true, so is  $q$ 
  - ▶  $\{A_0 A_1 \dots \mid \forall i \geq 0, r \in A_i \rightarrow q \in A_i\}$

# Syntax of Linear Temporal Logic

---

Given  $AP$ , a set of propositions,

# Syntax of Linear Temporal Logic

---

Given  $AP$ , a set of propositions,

- ▶ Propositional logic formulae over  $AP$ 
  - ▶  $a \in AP$  (atomic propositions)
  - ▶  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi$

# Syntax of Linear Temporal Logic

---

Given  $AP$ , a set of propositions,

- ▶ Propositional logic formulae over  $AP$ 
  - ▶  $a \in AP$  (atomic propositions)
  - ▶  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi$
- ▶ Temporal Operators
  - ▶  $\bigcirc\varphi$  (Next  $\varphi$ )
  - ▶  $\varphi U\psi$  ( $\varphi$  holds until a  $\psi$ -state is reached)
- ▶ LTL : Logic for describing LT properties

# Semantics (On the board)

---

LTL formulae  $\varphi$  over  $AP$  interpreted over words  $w \in \Sigma^\omega$ ,  $\Sigma = 2^{AP}$ ,  
 $w \models \varphi$

# Derived Operators

---

- ▶  $\text{true} = \varphi \vee \neg\varphi$
- ▶  $\text{false} = \neg\text{true}$
- ▶  $\Diamond\varphi = \text{true} \mathbf{U} \varphi$  (Eventually  $\varphi$ )
- ▶  $\Box\varphi = \neg\Diamond\neg\varphi$  (Forever  $\varphi$ )

## Precedence

- ▶ Unary Operators bind stronger than Binary
- ▶  $\bigcirc$  and  $\neg$  equally strong
- ▶  $\mathbf{U}$  takes precedence over  $\wedge, \vee, \rightarrow$ 
  - ▶  $a \vee b \mathbf{U} c \equiv a \vee (b \mathbf{U} c)$
  - ▶  $\bigcirc a \mathbf{U} \neg b \equiv (\bigcirc a) \mathbf{U} (\neg b)$

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* →  $\neg \bigcirc$  *green*)

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* →  $\neg \bigcirc$  *green*)
- ▶ Eventually the traffic light will become yellow

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* → ¬○*green*)
- ▶ Eventually the traffic light will become yellow  
◊*yellow*

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* → ¬○*green*)
- ▶ Eventually the traffic light will become yellow  
◊*yellow*
- ▶ Once the traffic light becomes yellow, it will eventually become green

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* →  $\neg \bigcirc$  *green*)
- ▶ Eventually the traffic light will become yellow  
◊*yellow*
- ▶ Once the traffic light becomes yellow, it will eventually become green  
□(*yellow* → ◊*green*)

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0 A_1 A_2 \dots$ .

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0 A_1 A_2 \dots$ .

- ▶  $\sigma \models a$  iff  $a \in A_0$

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0 A_1 A_2 \dots$ .

- ▶  $\sigma \models a$  iff  $a \in A_0$
- ▶  $\sigma \models \varphi_1 \wedge \varphi_2$  iff  $\sigma \models \varphi_1$  and  $\sigma \models \varphi_2$

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0 A_1 A_2 \dots$ .

- ▶  $\sigma \models a$  iff  $a \in A_0$
- ▶  $\sigma \models \varphi_1 \wedge \varphi_2$  iff  $\sigma \models \varphi_1$  and  $\sigma \models \varphi_2$
- ▶  $\sigma \models \neg \varphi$  iff  $\sigma \not\models \varphi$

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0 A_1 A_2 \dots$ .

- ▶  $\sigma \models a$  iff  $a \in A_0$
- ▶  $\sigma \models \varphi_1 \wedge \varphi_2$  iff  $\sigma \models \varphi_1$  and  $\sigma \models \varphi_2$
- ▶  $\sigma \models \neg \varphi$  iff  $\sigma \not\models \varphi$
- ▶  $\sigma \models \bigcirc \varphi$  iff  $A_1 A_2 \dots \models \varphi$

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0 A_1 A_2 \dots$ .

- ▶  $\sigma \models a$  iff  $a \in A_0$
- ▶  $\sigma \models \varphi_1 \wedge \varphi_2$  iff  $\sigma \models \varphi_1$  and  $\sigma \models \varphi_2$
- ▶  $\sigma \models \neg \varphi$  iff  $\sigma \not\models \varphi$
- ▶  $\sigma \models \bigcirc \varphi$  iff  $A_1 A_2 \dots \models \varphi$
- ▶  $\sigma \models \varphi \bigcup \psi$  iff  
 $\exists j \geq 0$  such that  $A_j A_{j+1} \dots \models \psi \wedge \forall 0 \leq i < j, A_i A_{i+1} \dots \models \varphi$

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

- $\sigma \models \Diamond \varphi$  iff  $\exists j \geq 0, A_j A_{j+1} \dots \models \varphi$

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

- ▶  $\sigma \models \Diamond \varphi$  iff  $\exists j \geq 0, A_j A_{j+1} \dots \models \varphi$
- ▶  $\sigma \models \Box \varphi$  iff  $\forall j \geq 0, A_j A_{j+1} \dots \models \varphi$

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

- ▶  $\sigma \models \Diamond \varphi$  iff  $\exists j \geq 0, A_j A_{j+1} \dots \models \varphi$
- ▶  $\sigma \models \Box \varphi$  iff  $\forall j \geq 0, A_j A_{j+1} \dots \models \varphi$
- ▶  $\sigma \models \Box \Diamond \varphi$  iff  $\forall j \geq 0, \exists i \geq j, A_i A_{i+1} \dots \models \varphi$

# Semantics over Infinite Words

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

- ▶  $\sigma \models \Diamond \varphi$  iff  $\exists j \geq 0, A_j A_{j+1} \dots \models \varphi$
- ▶  $\sigma \models \Box \varphi$  iff  $\forall j \geq 0, A_j A_{j+1} \dots \models \varphi$
- ▶  $\sigma \models \Box \Diamond \varphi$  iff  $\forall j \geq 0, \exists i \geq j, A_i A_{i+1} \dots \models \varphi$
- ▶  $\sigma \models \Diamond \Box \varphi$  iff  $\exists j \geq 0, \forall i \geq j, A_i A_{i+1} \dots \models \varphi$

If  $\sigma = A_0 A_1 A_2 \dots$ ,  $\sigma \models \varphi$  is also written as  $\sigma, 0 \models \varphi$ . This simply means  $A_0 A_1 A_2 \dots \models \varphi$ . One can also define  $\sigma, i \models \varphi$  to mean  $A_i A_{i+1} A_{i+2} \dots \models \varphi$  to talk about a suffix of the word  $\sigma$  satisfying a property.

# Transition System Semantics $TS \models \varphi$

---

Let  $TS = (S, S_0, \rightarrow, AP, L)$  be a transition system, and  $\varphi$  an LTL formula over  $AP$

- ▶ For an infinite path fragment  $\pi$  of  $TS$ ,

$$\pi \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi$$

# Transition System Semantics $TS \models \varphi$

---

Let  $TS = (S, S_0, \rightarrow, AP, L)$  be a transition system, and  $\varphi$  an LTL formula over  $AP$

- ▶ For an infinite path fragment  $\pi$  of  $TS$ ,

$$\pi \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi$$

- ▶ For  $s \in S$ ,

$$s \models \varphi \text{ iff } \forall \pi \in \text{Paths}(s), \pi \models \varphi$$

# Transition System Semantics $TS \models \varphi$

---

Let  $TS = (S, S_0, \rightarrow, AP, L)$  be a transition system, and  $\varphi$  an LTL formula over  $AP$

- ▶ For an infinite path fragment  $\pi$  of  $TS$ ,

$$\pi \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi$$

- ▶ For  $s \in S$ ,

$$s \models \varphi \text{ iff } \forall \pi \in \text{Paths}(s), \pi \models \varphi$$

- ▶  $TS \models \varphi$  iff  $\text{Traces}(TS) \subseteq L(\varphi)$

# Transition System Semantics $TS \models \varphi$

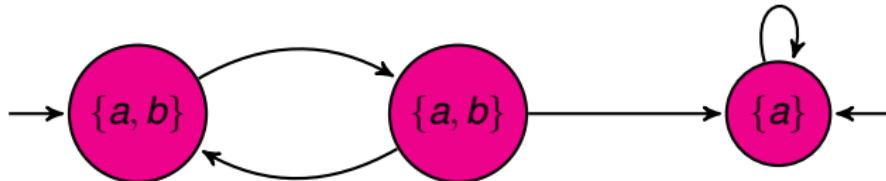
---

Assume all states in  $TS$  are reachable from  $S_0$ .

- ▶  $TS \models \varphi$  iff  $TS \models L(\varphi)$  iff  $Traces(TS) \subseteq L(\varphi)$
- ▶  $TS \models L(\varphi)$  iff  $\pi \models \varphi \forall \pi \in Paths(TS)$
- ▶  $\pi \models \varphi \forall \pi \in Paths(TS)$  iff  $s_0 \models \varphi \forall s_0 \in S_0$

# Example

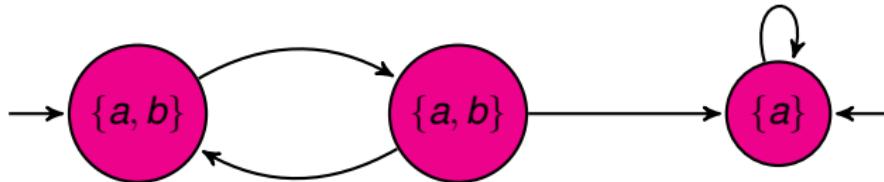
---



- ▶  $TS \models \Box a,$

# Example

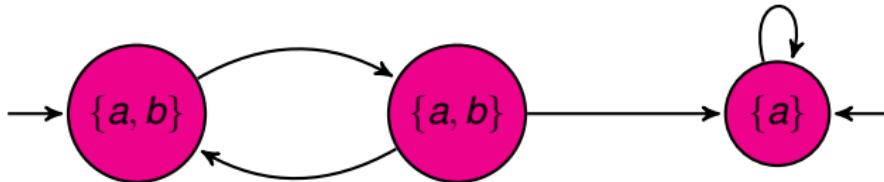
---



- ▶  $TS \models \Box a$ ,
- ▶  $TS \not\models \Diamond(a \wedge b)$

# Example

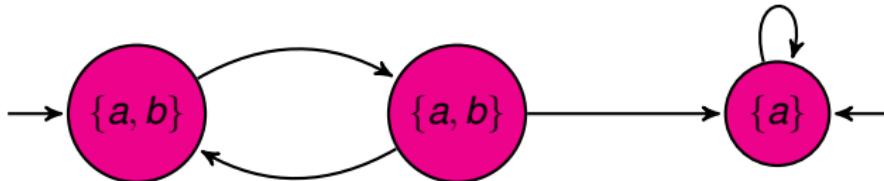
---



- ▶  $TS \models \Box a$ ,
- ▶  $TS \not\models \bigcirc(a \wedge b)$
- ▶  $TS \not\models (b \cup(a \wedge \neg b))$

# Example

---



- ▶  $TS \models \Box a,$
- ▶  $TS \not\models \bigcirc(a \wedge b)$
- ▶  $TS \not\models (b \cup(a \wedge \neg b))$
- ▶  $TS \models \Box(\neg b \rightarrow \Box(a \wedge \neg b))$

# More Semantics

---

- ▶ For paths  $\pi$ ,  $\pi \models \varphi$  iff  $\pi \not\models \neg\varphi$

# More Semantics

---

- ▶ For paths  $\pi$ ,  $\pi \models \varphi$  iff  $\pi \not\models \neg\varphi$   
 $trace(\pi) \in L(\varphi)$  iff  $trace(\pi) \notin L(\neg\varphi) = \overline{L(\varphi)}$
- ▶  $TS \not\models \varphi$  iff  $TS \models \neg\varphi$ ?
  - ▶  $TS \models \neg\varphi \rightarrow \forall$  paths  $\pi$  of  $TS$ ,  $\pi \models \neg\varphi$
  - ▶ Thus,  $\forall \pi$ ,  $\pi \not\models \varphi$ . Hence,  $TS \not\models \varphi$

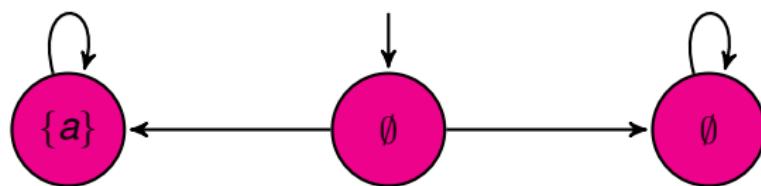
# More Semantics

---

- ▶ For paths  $\pi$ ,  $\pi \models \varphi$  iff  $\pi \not\models \neg\varphi$   
 $trace(\pi) \in L(\varphi)$  iff  $trace(\pi) \notin L(\neg\varphi) = \overline{L(\varphi)}$
- ▶  $TS \not\models \varphi$  iff  $TS \models \neg\varphi$ ?
  - ▶  $TS \models \neg\varphi \rightarrow \forall$  paths  $\pi$  of  $TS$ ,  $\pi \models \neg\varphi$
  - ▶ Thus,  $\forall \pi$ ,  $\pi \not\models \varphi$ . Hence,  $TS \not\models \varphi$
  - ▶ Now assume  $TS \not\models \varphi$
  - ▶ Then  $\exists$  some path  $\pi$  in  $TS$  such that  $\pi \models \neg\varphi$
  - ▶ However, there could be another path  $\pi'$  such that  $\pi' \models \varphi$
  - ▶ Then  $TS \not\models \neg\varphi$  as well
- ▶ Thus,  $TS \not\models \varphi \not\equiv TS \models \neg\varphi$ .

# An Example

---



$TS \not\models \diamond a$  and  $TS \not\models \square \neg a$

# Equivalence of LTL Formulae

---

## Equivalence

$\varphi$  and  $\psi$  are equivalent ( $\varphi \equiv \psi$ ) iff  $L(\varphi) = L(\psi)$ .

## Expansion Laws

- ▶  $\varphi \mathsf{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathsf{U} \psi))$
- ▶  $\Diamond \varphi \equiv \varphi \vee \bigcirc \Diamond \varphi$
- ▶  $\Box \varphi \equiv \varphi \wedge \bigcirc \Box \varphi$

# Equivalence of LTL Formulae

---

$\varphi$  and  $\psi$  are equivalent iff  $L(\varphi) = L(\psi)$ .

# Equivalence of LTL Formulae

---

$\varphi$  and  $\psi$  are equivalent iff  $L(\varphi) = L(\psi)$ .

## Distribution

$$\bigcirc(\varphi \vee \psi) \equiv \bigcirc\varphi \vee \bigcirc\psi,$$

$$\bigcirc(\varphi \wedge \psi) \equiv \bigcirc\varphi \wedge \bigcirc\psi,$$

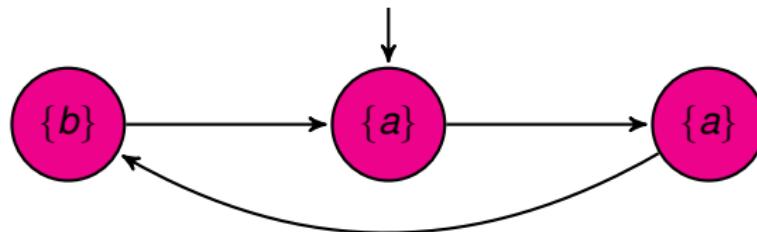
$$\bigcirc(\varphi \mathbf{U} \psi) \equiv (\bigcirc\varphi) \mathbf{U} (\bigcirc\psi),$$

$$\diamond(\varphi \vee \psi) \equiv \diamond\varphi \vee \diamond\psi,$$

$$\square(\varphi \wedge \psi) \equiv \square\varphi \wedge \square\psi$$

# Equivalence of LTL Formulae

---



$TS \models \diamond a \wedge \diamond b, TS \not\models \diamond(a \wedge b)$

$TS \models \square(a \vee b), TS \not\models \square a \vee \square b$

# Satisfiability, Model Checking of LTL

---

## Two Questions

Given transition system  $TS$ , and an LTL formula  $\varphi$ . Does  $TS \models \varphi$ ?  
Given an LTL formula  $\varphi$ , is  $L(\varphi) = \emptyset$ ?

How we go about this:

- ▶ Translate  $\varphi$  into an automaton  $A_\varphi$  that accepts infinite words such that  $L(A_\varphi) = L(\varphi)$ .
- ▶ Check for emptiness of  $A_\varphi$  to check satisfiability of  $\varphi$ .
- ▶ Check if  $TS \cap \overline{A_\varphi}$  is empty, to answer the model-checking problem.

## $\omega$ -automata

An  $\omega$ -automaton is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$  where

- ▶  $Q$  is a finite set of states
- ▶  $\Sigma$  is a finite alphabet
- ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$  is a state transition function (if non-deterministic,  
otherwise,  $\delta : Q \times \Sigma \rightarrow Q$ )
- ▶  $q_0 \in Q$  is an initial state and  $Acc$  is an acceptance condition

# $\omega$ -automata

An  $\omega$ -automaton is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$  where

- ▶  $Q$  is a finite set of states
- ▶  $\Sigma$  is a finite alphabet
- ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$  is a state transition function (if non-deterministic, otherwise,  $\delta : Q \times \Sigma \rightarrow Q$ )
- ▶  $q_0 \in Q$  is an initial state and  $Acc$  is an acceptance condition

## Run

A run  $\rho$  of  $\mathcal{A}$  on an  $\omega$ -word  $\alpha = a_1 a_2 \dots \in \Sigma^\omega$  is an infinite state sequence  $\rho(0) \rho(1) \rho(2) \dots$  such that

- ▶  $\rho(0) = q_0$ ,
- ▶  $\rho(i) = \delta(\rho(i-1), a_i)$  if  $\mathcal{A}$  is deterministic,
- ▶  $\rho(i) \in \delta(\rho(i-1), a_i)$  if  $\mathcal{A}$  is non-deterministic,

# $\omega$ -automata

An  $\omega$ -automaton is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$  where

- ▶  $Q$  is a finite set of states
- ▶  $\Sigma$  is a finite alphabet
- ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$  is a state transition function (if non-deterministic, otherwise,  $\delta : Q \times \Sigma \rightarrow Q$ )
- ▶  $q_0 \in Q$  is an initial state and  $Acc$  is an acceptance condition

## Run

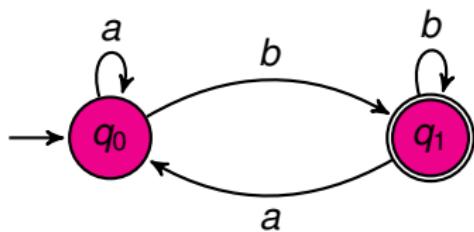
A run  $\rho$  of  $\mathcal{A}$  on an  $\omega$ -word  $\alpha = a_1 a_2 \dots \in \Sigma^\omega$  is an infinite state sequence  $\rho(0) \rho(1) \rho(2) \dots$  such that

- ▶  $\rho(0) = q_0$ ,
- ▶  $\rho(i) = \delta(\rho(i-1), a_i)$  if  $\mathcal{A}$  is deterministic,
- ▶  $\rho(i) \in \delta(\rho(i-1), a_i)$  if  $\mathcal{A}$  is non-deterministic,

## Büchi Acceptance

For Büchi Acceptance,  $Acc$  is specified as a set of states,  $G \subseteq Q$ . The  $\omega$ -word  $\alpha$  is accepted if there is a run  $\rho$  of  $\alpha$  such that  $Inf(\rho) \cap G \neq \emptyset$ .

# $\omega$ -Automata with Büchi Acceptance

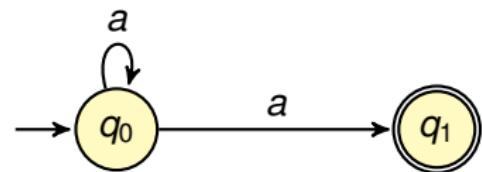
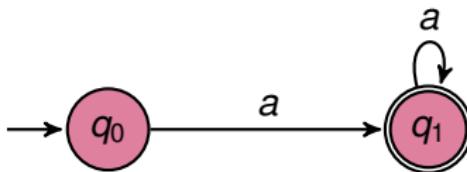


$$L(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha \text{ has a run } \rho \text{ such that } \text{Inf}(\rho) \cap G \neq \emptyset\}$$

Language accepted=Infinitely many b's.

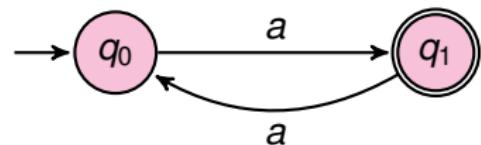
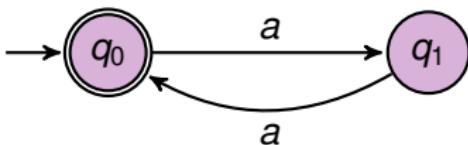
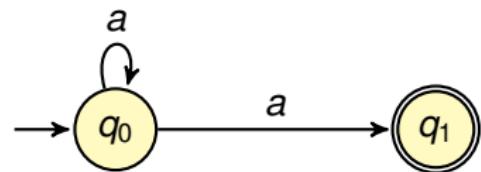
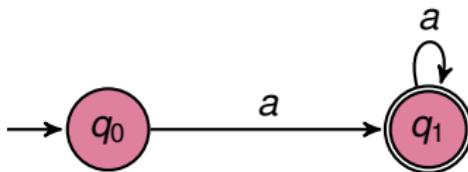
# Comparing NFA and NBA

---

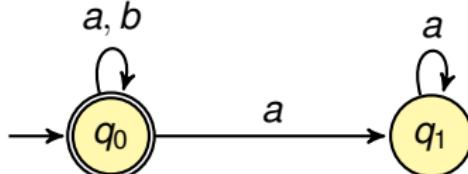
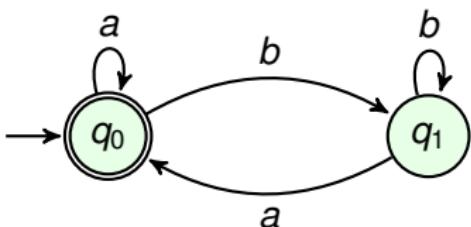
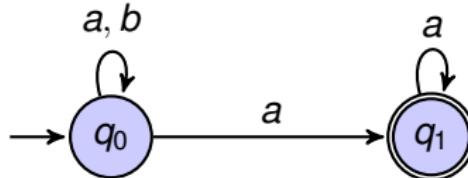
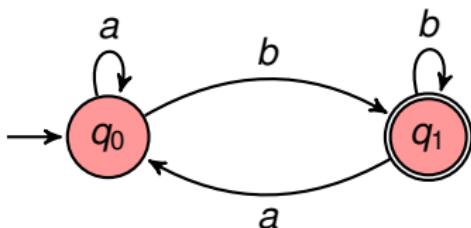


# Comparing NFA and NBA

---



# $\omega$ -Automata with Büchi Acceptance

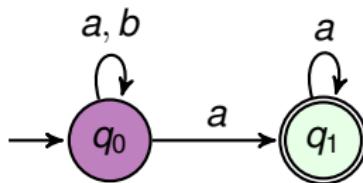


- ▶ Left (T-B): Inf many  $b$ 's, Inf many  $a$ 's
- ▶ Right (T-B): Finitely many  $b$ 's,  $(a + b)^\omega$

# NBA and DBA

---

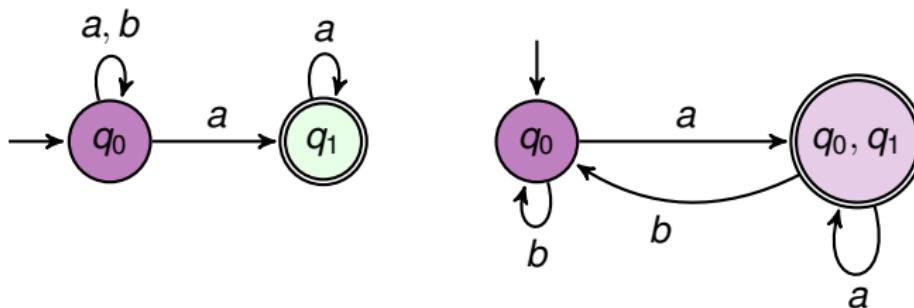
- ▶ Is every DBA as expressible as a NBA, like in the case of DFA and NFA?
- ▶ Can we do subset construction on NBA and obtain DBA?



# NBA and DBA

---

- ▶ Is every DBA as expressible as a NBA, like in the case of DFA and NFA?
- ▶ Can we do subset construction on NBA and obtain DBA?



# NBA and DBA

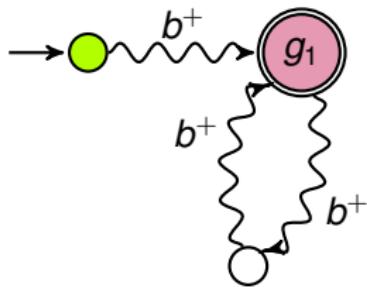
---

There does not exist a deterministic Büchi automata capturing the language finitely many a's.

# NBA and DBA

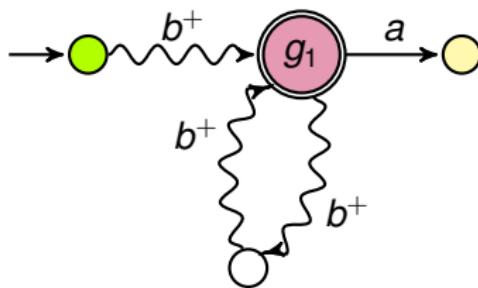
---

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



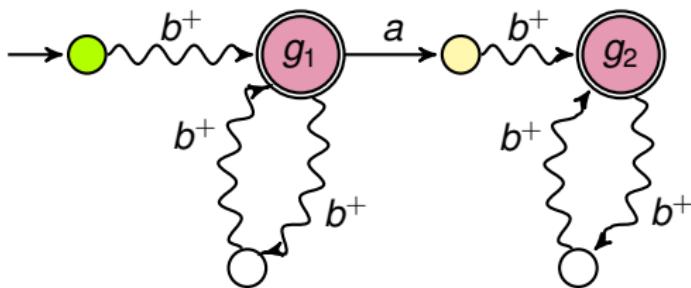
# NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



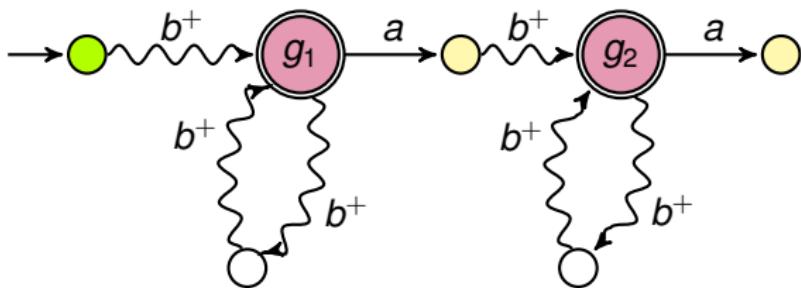
# NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



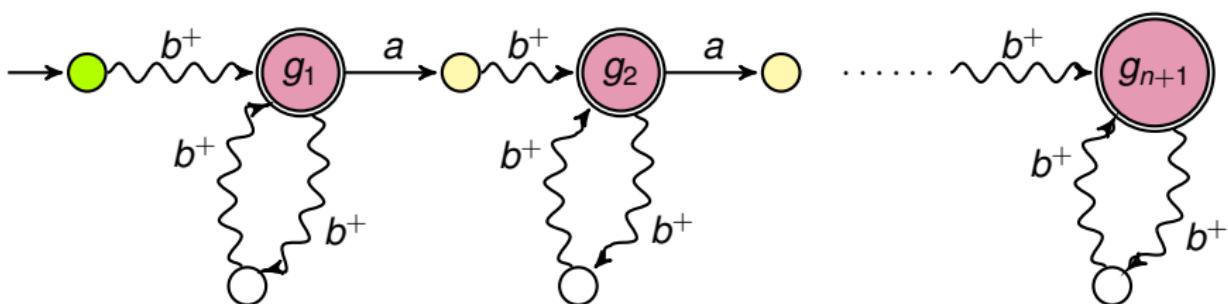
# NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



# NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



# Lecture 27 Model Checking LTL.pdf

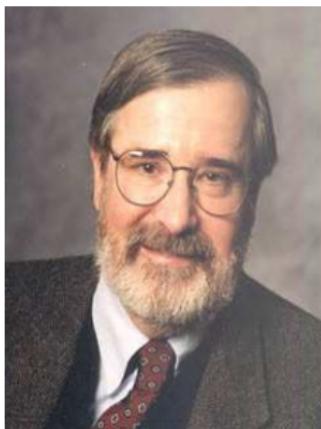
# CS 228 : Logic in Computer Science

Krishna. S

## Linear Temporal Logic

# Model Checking

---



- ▶ Year 2007 : ACM confers the **Turing Award** to the pioneers of Model Checking: **Ed Clarke, Allen Emerson, and Joseph Sifakis**
- ▶  
[https://amturing.acm.org/award\\_winners/clarke\\_1167964](https://amturing.acm.org/award_winners/clarke_1167964)

# Model checking

---

- ▶ Model checking has evolved in last 25 years into a widely used verification and debugging technique for software and hardware.
- ▶ 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



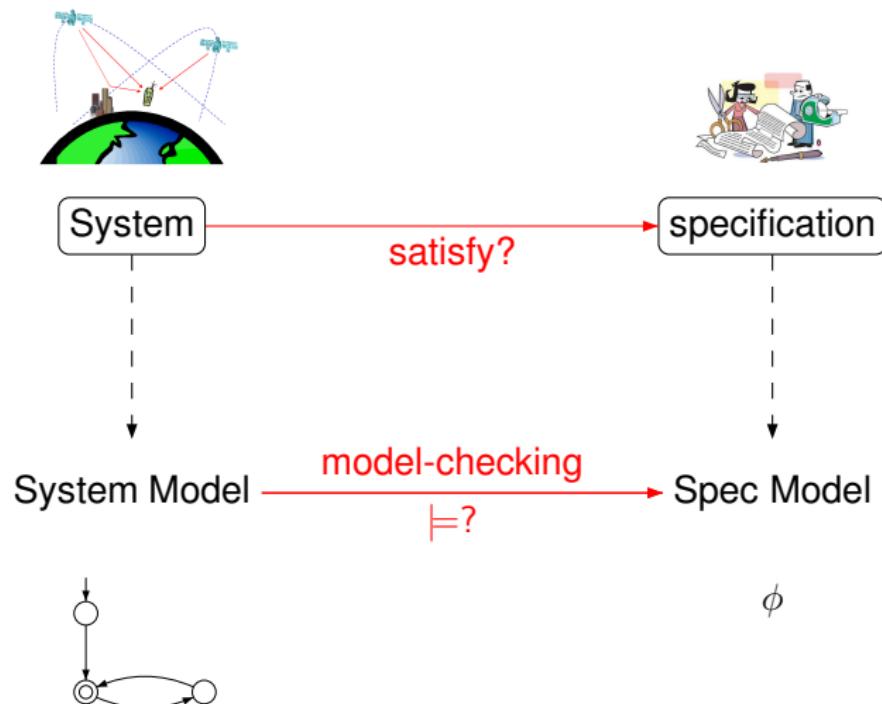
satisfy?



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 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?

# Notations for Infinite Words

---

- ▶  $\Sigma$  is a finite alphabet
- ▶  $\Sigma^*$  set of finite words over  $\Sigma$
- ▶ An infinite word is written as  $\alpha = \alpha(0)\alpha(1)\alpha(2)\dots$ , where  $\alpha(i) \in \Sigma$
- ▶ Such words are called  $\omega$ -words
- ▶  $a^\omega, a^7.b^\omega$

# Transition Systems

---

A **Transition System** is a tuple  $(S, Act, \rightarrow, I, AP, L)$  where

- ▶  $S$  is a set of **states**
- ▶  $Act$  is a set of **actions**
- ▶  $s \xrightarrow{\alpha} s'$  in  $S \times Act \times S$  is the **transition relation**
- ▶  $I \subseteq S$  is the **set of initial states**
- ▶  $AP$  is the set of **atomic propositions**
- ▶  $L : S \rightarrow 2^{AP}$  is the **labeling function**

# Traces of Transition Systems

---

- ▶ Labels of the locations represent values of all observable propositions  $\in AP$
- ▶ Captures system state
- ▶ Focus on sequences  $L(s_0)L(s_1)\dots$  of labels of locations
- ▶ Such sequences are called **traces**
- ▶ Assuming transition systems have no terminal states,
  - ▶ Traces are infinite words over  $2^{AP}$
  - ▶ Traces  $\in (2^{AP})^\omega$
  - ▶ Go to the example slide and define traces

# Traces of Transition Systems

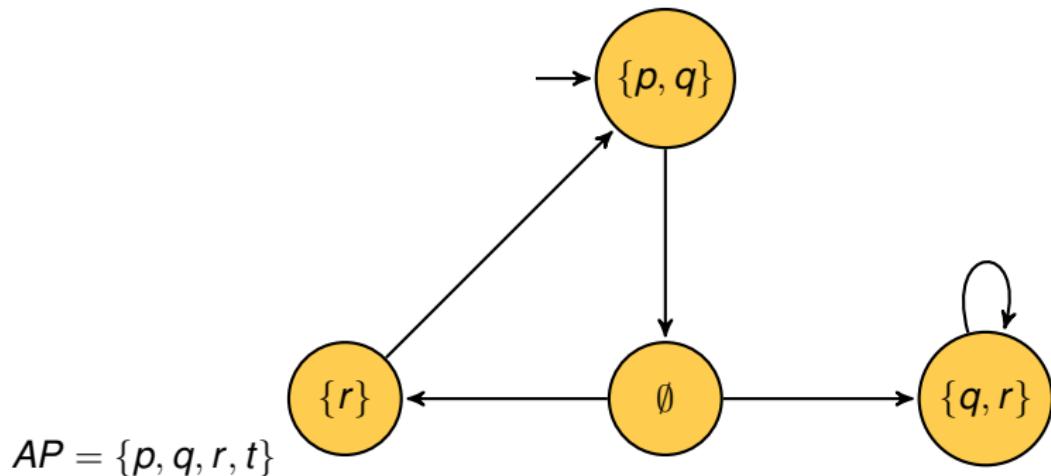
---

Given a transition system  $TS = (S, Act, \rightarrow, I, AP, L)$  **without terminal states**,

- ▶ All maximal executions/paths are infinite
- ▶ Path  $\pi = s_0 s_1 s_2 \dots$ ,  $trace(\pi) = L(s_0)L(s_1)\dots$
- ▶ For a set  $\Pi$  of paths,  $Trace(\Pi) = \{trace(\pi) \mid \pi \in \Pi\}$
- ▶ For a location  $s$ ,  $Traces(s) = Trace(Paths(s))$
- ▶  $Traces(TS) = \bigcup_{s \in I} Traces(s)$

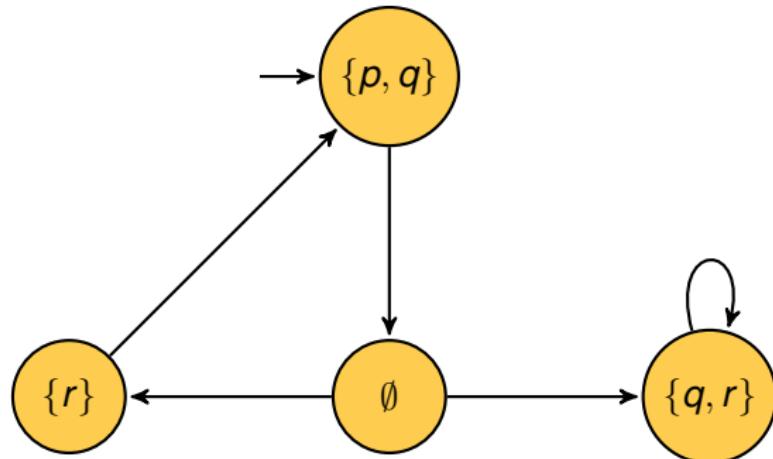
# Example Traces

---



# Example Traces

---

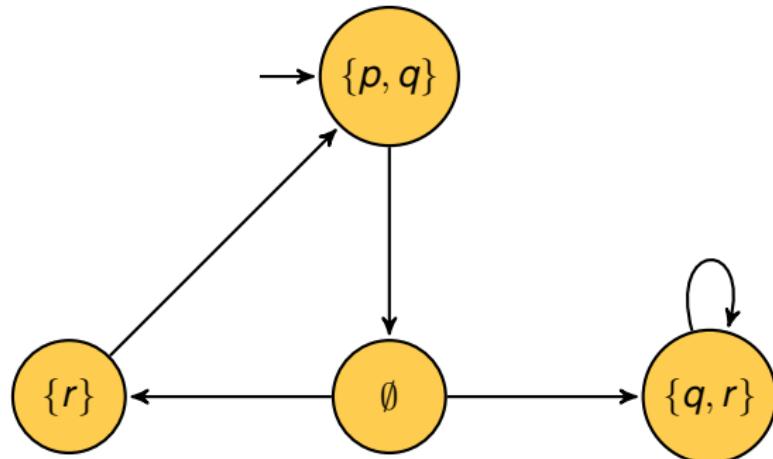


$$AP = \{p, q, r, t\}$$

►  $\{p, q\} \emptyset \{q, r\}^\omega$

# Example Traces

---

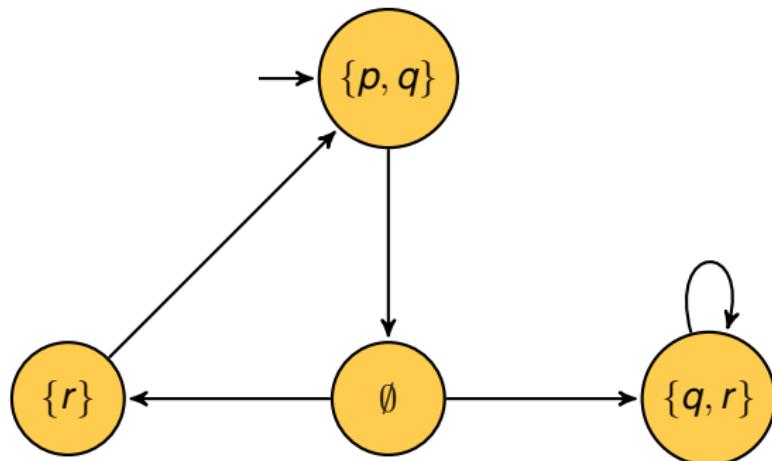


$$AP = \{p, q, r, t\}$$

- ▶  $\{p, q\} \emptyset \{q, r\}^\omega$
- ▶  $(\{p, q\} \emptyset \{r\})^\omega$

# Example Traces

---



$$AP = \{p, q, r, t\}$$

- ▶  $\{p, q\} \emptyset \{q, r\}^\omega$
- ▶  $(\{p, q\} \emptyset \{r\})^\omega$
- ▶  $(\{p, q\} \emptyset \{r\})^* \{p, q\} \emptyset \{q, r\}^\omega$

# Linear Time Properties

---

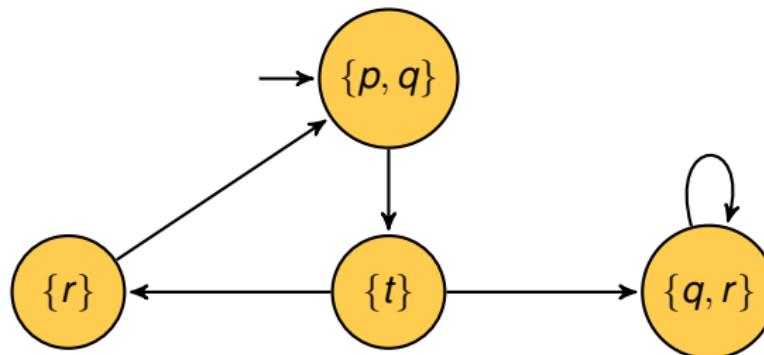
- ▶ Linear-time properties specify traces that a  $TS$  must have
- ▶ A LT property  $P$  over  $AP$  is a subset of  $(2^{AP})^\omega$
- ▶  $TS$  over  $AP$  satisfies a LT property  $P$  over  $AP$

$$TS \models P \text{ iff } \text{Traces}(TS) \subseteq P$$

- ▶  $s \in S$  satisfies LT property  $P$  (denoted  $s \models P$ ) iff  $\text{Traces}(s) \subseteq P$

# Specifying Traces

---



- ▶ Whenever  $p$  is true,  $r$  will eventually become true
  - ▶  $\{A_0A_1A_2 \dots \mid \forall i \geq 0, p \in A_i \rightarrow \exists j \geq i, r \in A_j\}$
- ▶  $q$  is true infinitely often
  - ▶  $\{A_0A_1A_2 \dots \mid \forall i \geq 0, \exists j \geq i, q \in A_j\}$
- ▶ Whenever  $r$  is true, so is  $q$ 
  - ▶  $\{A_0A_1 \dots \mid \forall i \geq 0, r \in A_i \rightarrow q \in A_i\}$

# Syntax of Linear Temporal Logic

---

Given  $AP$ , a set of propositions,

# Syntax of Linear Temporal Logic

---

Given  $AP$ , a set of propositions,

- ▶ Propositional logic formulae over  $AP$ 
  - ▶  $a \in AP$  (atomic propositions)
  - ▶  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi$

# Syntax of Linear Temporal Logic

---

Given  $AP$ , a set of propositions,

- ▶ Propositional logic formulae over  $AP$ 
  - ▶  $a \in AP$  (atomic propositions)
  - ▶  $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi$
- ▶ Temporal Operators
  - ▶  $\bigcirc\varphi$  (Next  $\varphi$ )
  - ▶  $\varphi U\psi$  ( $\varphi$  holds until a  $\psi$ -state is reached)
- ▶ LTL : Logic for describing LT properties

# Semantics (On the board)

---

LTL formulae  $\varphi$  over  $AP$  interpreted over words  $w \in \Sigma^\omega$ ,  $\Sigma = 2^{AP}$ ,  
 $w \models \varphi$

# Lecture 28-29 LTL.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Derived Operators

---

- ▶  $\text{true} = \varphi \vee \neg\varphi$
- ▶  $\text{false} = \neg\text{true}$
- ▶  $\Diamond\varphi = \text{true} \mathbf{U} \varphi$  (Eventually  $\varphi$ )
- ▶  $\Box\varphi = \neg\Diamond\neg\varphi$  (Forever  $\varphi$ )

## Precedence

- ▶ Unary Operators bind stronger than Binary
- ▶  $\bigcirc$  and  $\neg$  equally strong
- ▶  $\mathbf{U}$  takes precedence over  $\wedge, \vee, \rightarrow$ 
  - ▶  $a \vee b \mathbf{U} c \equiv a \vee (b \mathbf{U} c)$
  - ▶  $\bigcirc a \mathbf{U} \neg b \equiv (\bigcirc a) \mathbf{U} (\neg b)$

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* →  $\neg \bigcirc$  *green*)

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
 $\Box(\text{red} \rightarrow \neg \bigcirc \text{green})$
- ▶ Eventually the traffic light will become yellow

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* → ¬○*green*)
- ▶ Eventually the traffic light will become yellow  
◊*yellow*

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* →  $\neg \bigcirc$  *green*)
- ▶ Eventually the traffic light will become yellow  
◊*yellow*
- ▶ Once the traffic light becomes yellow, it will eventually become green

# Examples

---

- ▶ Whenever the traffic light is red, it cannot become green immediately:  
□(*red* →  $\neg \bigcirc$  *green*)
- ▶ Eventually the traffic light will become yellow  
◊*yellow*
- ▶ Once the traffic light becomes yellow, it will eventually become green  
□(*yellow* → ◊*green*)

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0 A_1 A_2 \dots$ , with  $A_i \subseteq AP$ .

# Semantics over Infinite Words

---

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0A_1A_2\dots$ , with  $A_i \subseteq AP$ .

- ▶  $\sigma \models a$  iff  $a \in A_0$
- ▶  $\sigma \models \varphi_1 \wedge \varphi_2$  iff  $\sigma \models \varphi_1$  and  $\sigma \models \varphi_2$
- ▶  $\sigma \models \neg\varphi$  iff  $\sigma \not\models \varphi$
- ▶  $\sigma \models \bigcirc\varphi$  iff  $A_1A_2\dots \models \varphi$
- ▶  $\sigma \models \varphi \mathbf{U} \psi$  iff  
 $\exists j \geq 0$  such that  $A_jA_{j+1}\dots \models \psi \wedge \forall 0 \leq i < j, A_iA_{i+1}\dots \models \varphi$

# Semantics over Infinite Words

Given LTL formula  $\varphi$  over  $AP$ ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let  $\sigma = A_0A_1A_2\dots$ , with  $A_i \subseteq AP$ .

- ▶  $\sigma \models \diamond \varphi$  iff  $\exists j \geq 0, A_jA_{j+1}\dots \models \varphi$
- ▶  $\sigma \models \square \varphi$  iff  $\forall j \geq 0, A_jA_{j+1}\dots \models \varphi$
- ▶  $\sigma \models \square \diamond \varphi$  iff  $\forall j \geq 0, \exists i \geq j, A_iA_{i+1}\dots \models \varphi$
- ▶  $\sigma \models \diamond \square \varphi$  iff  $\exists j \geq 0, \forall i \geq j, A_iA_{i+1}\dots \models \varphi$

If  $\sigma = A_0A_1A_2\dots$ ,  $\sigma \models \varphi$  is also written as  $\sigma, 0 \models \varphi$ . This simply means  $A_0A_1A_2\dots \models \varphi$ . One can also define  $\sigma, i \models \varphi$  to mean  $A_iA_{i+1}A_{i+2}\dots \models \varphi$  to talk about a suffix of the word  $\sigma$  satisfying a property.

# Transition System Semantics $TS \models \varphi$

---

Let  $TS = (S, S_0, \rightarrow, AP, L)$  be a transition system, and  $\varphi$  an LTL formula over  $AP$

- ▶ For an infinite path fragment  $\pi$  of  $TS$ ,

$$\pi \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi$$

# Transition System Semantics $TS \models \varphi$

---

Let  $TS = (S, S_0, \rightarrow, AP, L)$  be a transition system, and  $\varphi$  an LTL formula over  $AP$

- ▶ For an infinite path fragment  $\pi$  of  $TS$ ,

$$\pi \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi$$

- ▶ For  $s \in S$ ,

$$s \models \varphi \text{ iff } \forall \pi \in \text{Paths}(s), \pi \models \varphi$$

# Transition System Semantics $TS \models \varphi$

---

Let  $TS = (S, S_0, \rightarrow, AP, L)$  be a transition system, and  $\varphi$  an LTL formula over  $AP$

- ▶ For an infinite path fragment  $\pi$  of  $TS$ ,

$$\pi \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi$$

- ▶ For  $s \in S$ ,

$$s \models \varphi \text{ iff } \forall \pi \in \text{Paths}(s), \pi \models \varphi$$

- ▶  $TS \models \varphi$  iff  $\text{Traces}(TS) \subseteq L(\varphi)$

# Transition System Semantics $TS \models \varphi$

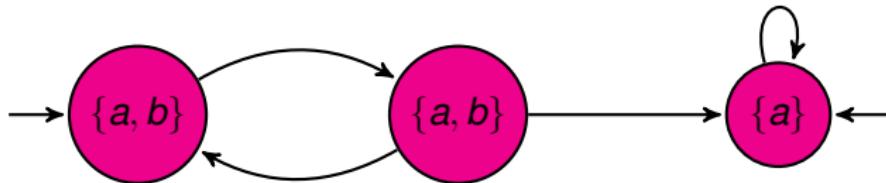
---

Assume all states in  $TS$  are reachable from  $S_0$ .

- ▶  $TS \models \varphi$  iff  $TS \models L(\varphi)$  iff  $Traces(TS) \subseteq L(\varphi)$
- ▶  $TS \models L(\varphi)$  iff  $\pi \models \varphi \forall \pi \in Paths(TS)$
- ▶  $\pi \models \varphi \forall \pi \in Paths(TS)$  iff  $s_0 \models \varphi \forall s_0 \in S_0$

# Example

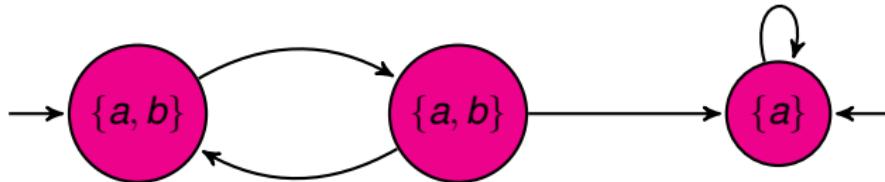
---



- ▶  $TS \models \Box a,$

# Example

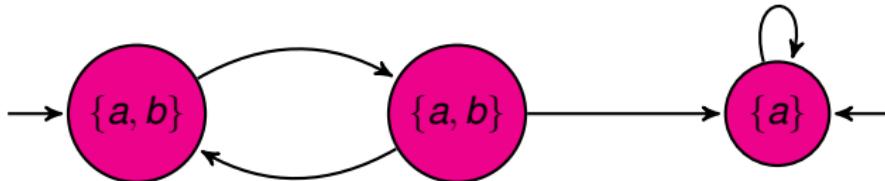
---



- ▶  $TS \models \Box a$ ,
- ▶  $TS \not\models \Diamond(a \wedge b)$

# Example

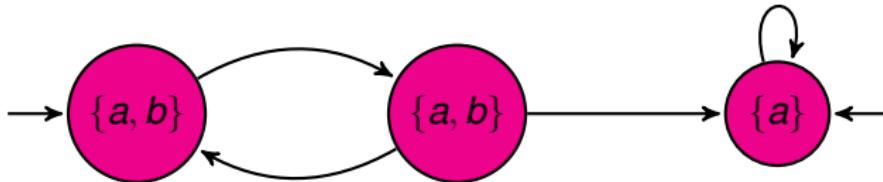
---



- ▶  $TS \models \Box a$ ,
- ▶  $TS \not\models \bigcirc(a \wedge b)$
- ▶  $TS \not\models (b \cup(a \wedge \neg b))$

# Example

---



- ▶  $TS \models \Box a,$
- ▶  $TS \not\models \bigcirc(a \wedge b)$
- ▶  $TS \not\models (b \cup (a \wedge \neg b))$
- ▶  $TS \models \Box(\neg b \rightarrow \Box(a \wedge \neg b))$

# More Semantics

---

- ▶ For paths  $\pi$ ,  $\pi \models \varphi$  iff  $\pi \not\models \neg\varphi$

# More Semantics

---

- ▶ For paths  $\pi$ ,  $\pi \models \varphi$  iff  $\pi \not\models \neg\varphi$   
 $trace(\pi) \in L(\varphi)$  iff  $trace(\pi) \notin L(\neg\varphi) = \overline{L(\varphi)}$
- ▶  $TS \not\models \varphi$  iff  $TS \models \neg\varphi$ ?

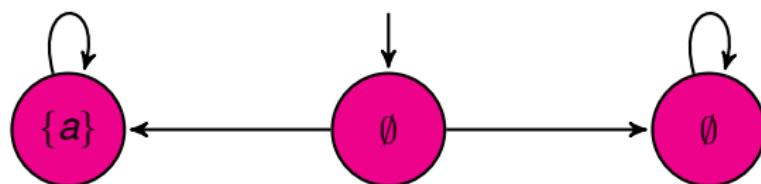
# More Semantics

---

- ▶ For paths  $\pi$ ,  $\pi \models \varphi$  iff  $\pi \not\models \neg\varphi$   
 $trace(\pi) \in L(\varphi)$  iff  $trace(\pi) \notin L(\neg\varphi) = \overline{L(\varphi)}$
- ▶  $TS \not\models \varphi$  iff  $TS \models \neg\varphi$ ?
  - ▶  $TS \models \neg\varphi \rightarrow \forall$  paths  $\pi$  of  $TS$ ,  $\pi \models \neg\varphi$
  - ▶ Thus,  $\forall \pi$ ,  $\pi \not\models \varphi$ . Hence,  $TS \not\models \varphi$
  - ▶ Now assume  $TS \not\models \varphi$
  - ▶ Then  $\exists$  some path  $\pi$  in  $TS$  such that  $\pi \models \neg\varphi$
  - ▶ However, there could be another path  $\pi'$  such that  $\pi' \models \varphi$
  - ▶ Then  $TS \not\models \neg\varphi$  as well
- ▶ Thus,  $TS \not\models \varphi \not\equiv TS \models \neg\varphi$ .

# An Example

---



$TS \not\models \diamond a$  and  $TS \not\models \square \neg a$

# Equivalence of LTL Formulae

---

## Equivalence

$\varphi$  and  $\psi$  are equivalent ( $\varphi \equiv \psi$ ) iff  $L(\varphi) = L(\psi)$ .

## Expansion Laws

- ▶  $\varphi \mathsf{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathsf{U} \psi))$
- ▶  $\Diamond \varphi \equiv \varphi \vee \bigcirc \Diamond \varphi$
- ▶  $\Box \varphi \equiv \varphi \wedge \bigcirc \Box \varphi$

# Equivalence of LTL Formulae

---

$\varphi$  and  $\psi$  are equivalent iff  $L(\varphi) = L(\psi)$ .

# Equivalence of LTL Formulae

---

$\varphi$  and  $\psi$  are equivalent iff  $L(\varphi) = L(\psi)$ .

## Distribution

$$\bigcirc(\varphi \vee \psi) \equiv \bigcirc\varphi \vee \bigcirc\psi,$$

$$\bigcirc(\varphi \wedge \psi) \equiv \bigcirc\varphi \wedge \bigcirc\psi,$$

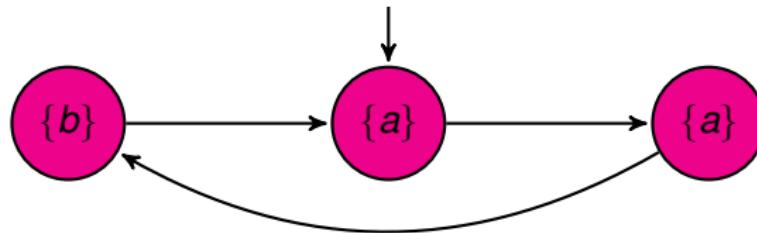
$$\bigcirc(\varphi \mathbf{U} \psi) \equiv (\bigcirc\varphi) \mathbf{U} (\bigcirc\psi),$$

$$\diamond(\varphi \vee \psi) \equiv \diamond\varphi \vee \diamond\psi,$$

$$\square(\varphi \wedge \psi) \equiv \square\varphi \wedge \square\psi$$

# Equivalence of LTL Formulae

---



$TS \models \diamond a \wedge \diamond b, TS \not\models \diamond(a \wedge b)$

$TS \models \square(a \vee b), TS \not\models \square a \vee \square b$

# Satisfiability, Model Checking of LTL

---

## Two Questions

Given transition system  $TS$ , and an LTL formula  $\varphi$ . Does  $TS \models \varphi$ ?

Given an LTL formula  $\varphi$ , is  $L(\varphi) = \emptyset$ ?

## $\omega$ -automata

An  $\omega$ -automaton is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$  where

- ▶  $Q$  is a finite set of states
- ▶  $\Sigma$  is a finite alphabet
- ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$  is a state transition function (if non-deterministic,  
otherwise,  $\delta : Q \times \Sigma \rightarrow Q$ )
- ▶  $q_0 \in Q$  is an initial state and  $Acc$  is an acceptance condition

# $\omega$ -automata

An  $\omega$ -automaton is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$  where

- ▶  $Q$  is a finite set of states
- ▶  $\Sigma$  is a finite alphabet
- ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$  is a state transition function (if non-deterministic, otherwise,  $\delta : Q \times \Sigma \rightarrow Q$ )
- ▶  $q_0 \in Q$  is an initial state and  $Acc$  is an acceptance condition

## Run

A run  $\rho$  of  $\mathcal{A}$  on an  $\omega$ -word  $\alpha = a_1 a_2 \dots \in \Sigma^\omega$  is an infinite state sequence  $\rho(0) \rho(1) \rho(2) \dots$  such that

- ▶  $\rho(0) = q_0$ ,
- ▶  $\rho(i) = \delta(\rho(i-1), a_i)$  if  $\mathcal{A}$  is deterministic,
- ▶  $\rho(i) \in \delta(\rho(i-1), a_i)$  if  $\mathcal{A}$  is non-deterministic,

# $\omega$ -automata

An  $\omega$ -automaton is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$  where

- ▶  $Q$  is a finite set of states
- ▶  $\Sigma$  is a finite alphabet
- ▶  $\delta : Q \times \Sigma \rightarrow 2^Q$  is a state transition function (if non-deterministic, otherwise,  $\delta : Q \times \Sigma \rightarrow Q$ )
- ▶  $q_0 \in Q$  is an initial state and  $Acc$  is an acceptance condition

## Run

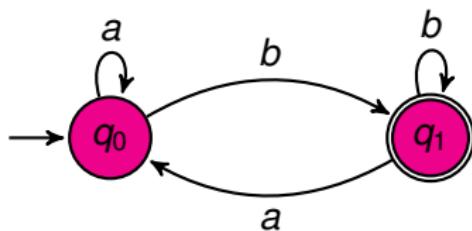
A run  $\rho$  of  $\mathcal{A}$  on an  $\omega$ -word  $\alpha = a_1 a_2 \dots \in \Sigma^\omega$  is an infinite state sequence  $\rho(0) \rho(1) \rho(2) \dots$  such that

- ▶  $\rho(0) = q_0$ ,
- ▶  $\rho(i) = \delta(\rho(i-1), a_i)$  if  $\mathcal{A}$  is deterministic,
- ▶  $\rho(i) \in \delta(\rho(i-1), a_i)$  if  $\mathcal{A}$  is non-deterministic,

## Büchi Acceptance

For Büchi Acceptance,  $Acc$  is specified as a set of states,  $G \subseteq Q$ . The  $\omega$ -word  $\alpha$  is accepted if there is a run  $\rho$  of  $\alpha$  such that  $Inf(\rho) \cap G \neq \emptyset$ .

# $\omega$ -Automata with Büchi Acceptance

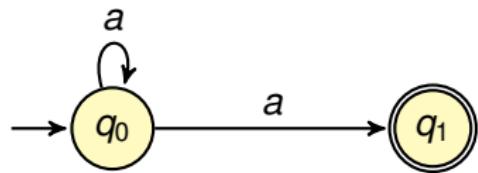
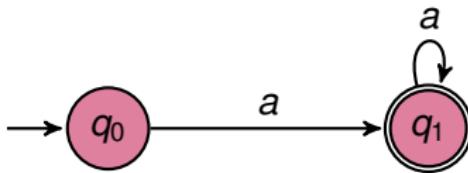


$$L(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha \text{ has a run } \rho \text{ such that } \text{Inf}(\rho) \cap G \neq \emptyset\}$$

Language accepted=Infinitely many b's.

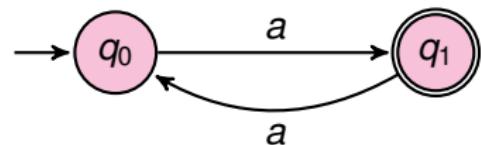
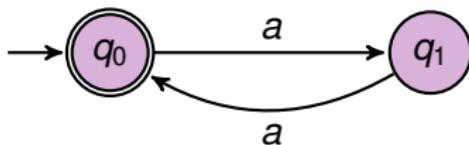
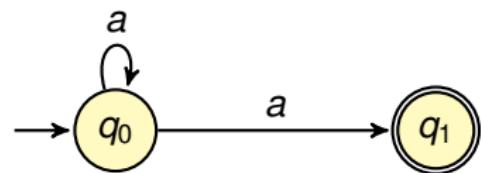
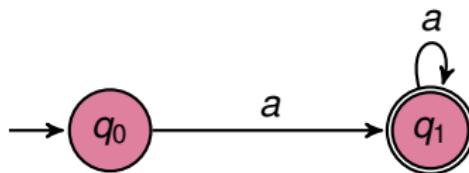
# Comparing NFA and NBA

---

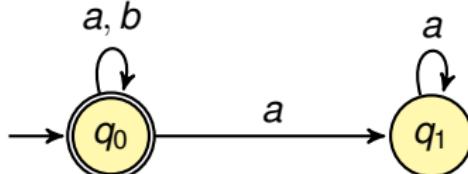
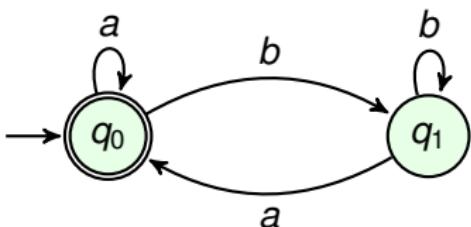
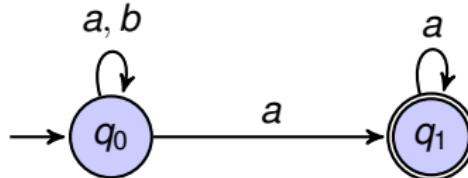
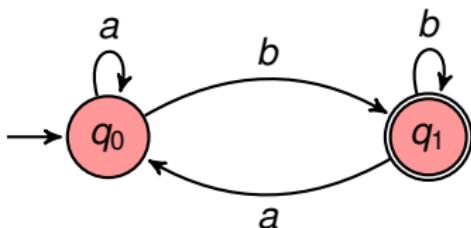


# Comparing NFA and NBA

---



# $\omega$ -Automata with Büchi Acceptance

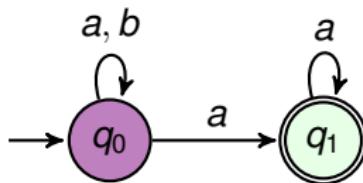


- ▶ Left (T-B): Inf many  $b$ 's, Inf many  $a$ 's
- ▶ Right (T-B): Finitely many  $b$ 's,  $(a + b)^\omega$

# NBA and DBA

---

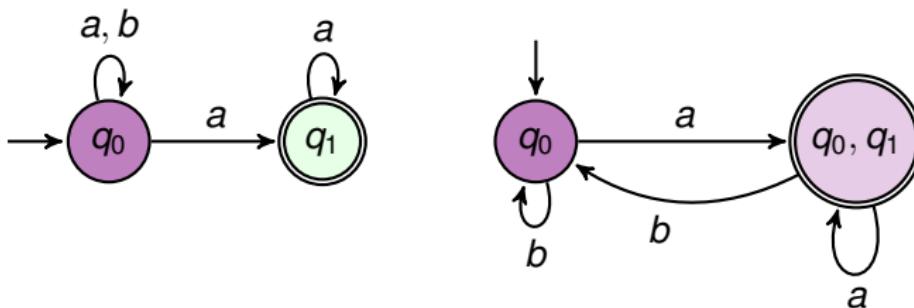
- ▶ Is every DBA as expressible as a NBA, like in the case of DFA and NFA?
- ▶ Can we do subset construction on NBA and obtain DBA?



# NBA and DBA

---

- ▶ Is every DBA as expressible as a NBA, like in the case of DFA and NFA?
- ▶ Can we do subset construction on NBA and obtain DBA?



# NBA and DBA

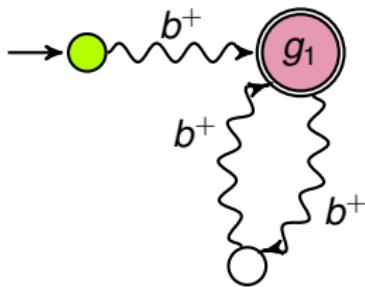
---

There does not exist a deterministic Büchi automata capturing the language finitely many a's.

# NBA and DBA

---

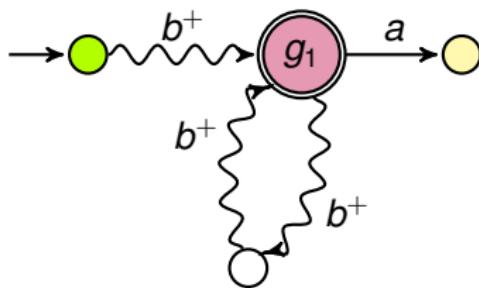
There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



# NBA and DBA

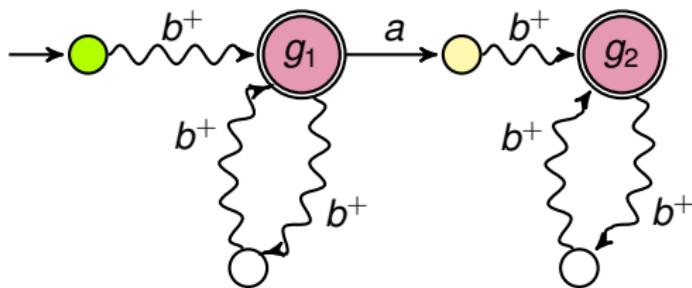
---

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



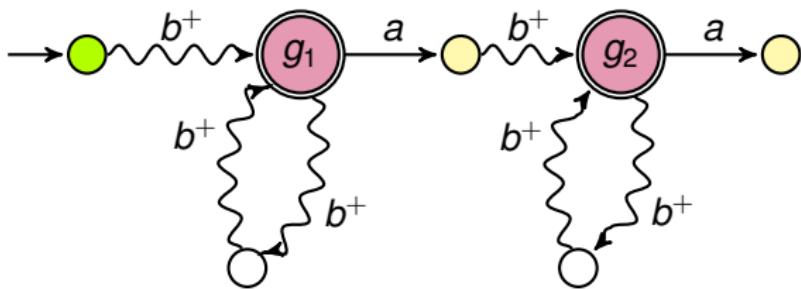
# NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



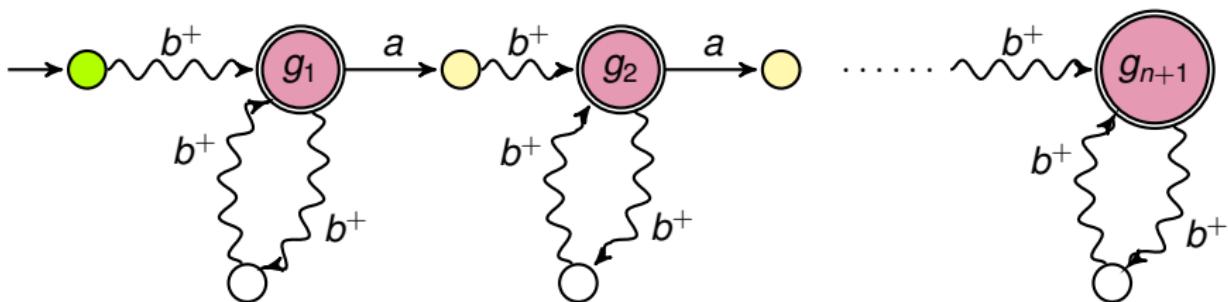
# NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



# NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many  $a$ 's.



# Lecture 30.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# So Far

---

- ▶  $\omega$ -automata with Büchi acceptance, also called Büchi automata
- ▶ Non-determinism versus determinism

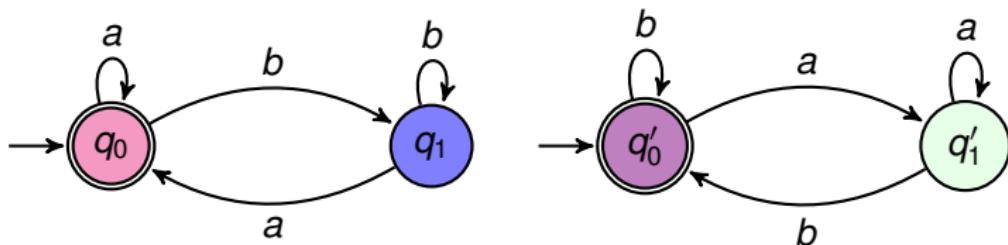
# Büchi Acceptance

---

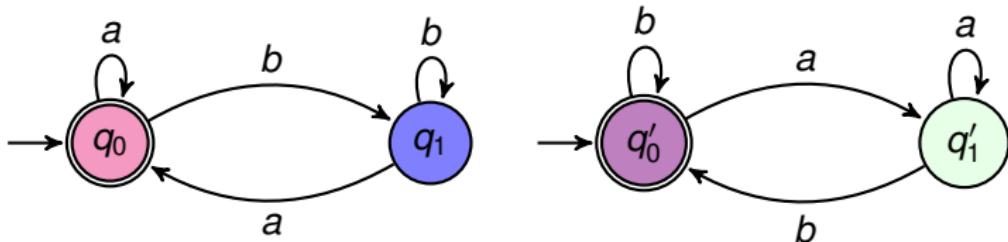
A language  $L \subseteq \Sigma^\omega$  is called  $\omega$ -regular if there exists a NBA  $\mathcal{A}$  such that  $L = L(\mathcal{A})$ .

# Union and Intersection of NBA

---

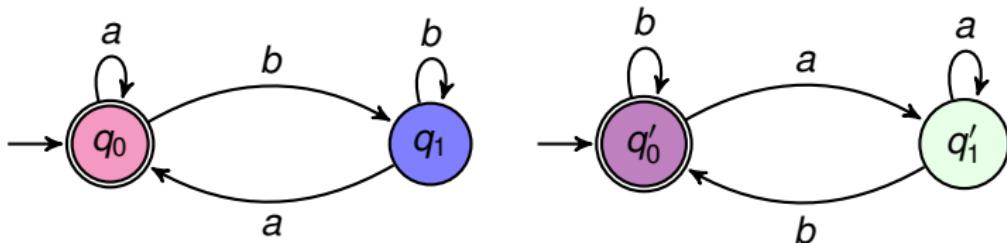


# Union and Intersection of NBA



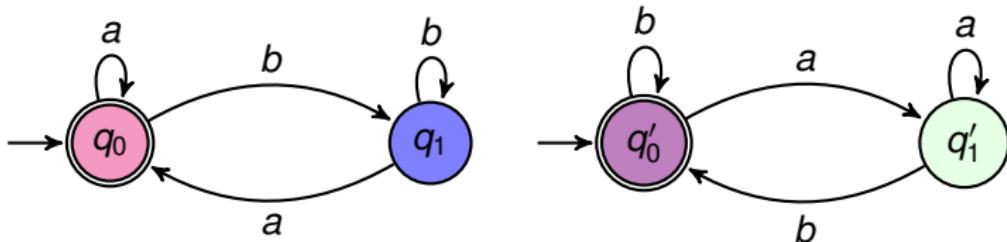
- ▶ States as  $Q_1 \times Q_2 \times \{1, 2\}$ , start state  $(q_0, q'_0, 1)$

# Union and Intersection of NBA



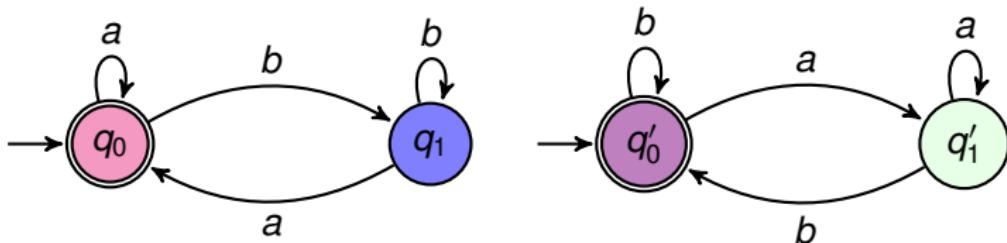
- ▶ States as  $Q_1 \times Q_2 \times \{1, 2\}$ , start state  $(q_0, q'_0, 1)$
- ▶  $(q_1, q_2, 1) \xrightarrow{a} (q'_1, q'_2, 1)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_1 \notin G_1$
- ▶  $(q_1, q_2, 1) \xrightarrow{a} (q'_1, q'_2, 2)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_1 \in G_1$

# Union and Intersection of NBA



- ▶ States as  $Q_1 \times Q_2 \times \{1, 2\}$ , start state  $(q_0, q'_0, 1)$
- ▶  $(q_1, q_2, 1) \xrightarrow{a} (q'_1, q'_2, 1)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_1 \notin G_1$
- ▶  $(q_1, q_2, 1) \xrightarrow{a} (q'_1, q'_2, 2)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_1 \in G_1$
- ▶  $(q_1, q_2, 2) \xrightarrow{a} (q'_1, q'_2, 2)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_2 \notin G_2$
- ▶  $(q_1, q_2, 2) \xrightarrow{a} (q'_1, q'_2, 1)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_2 \in G_2$

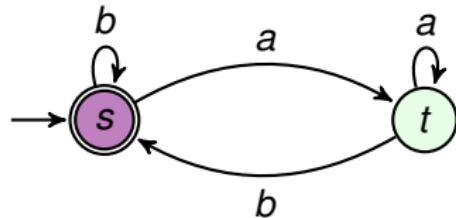
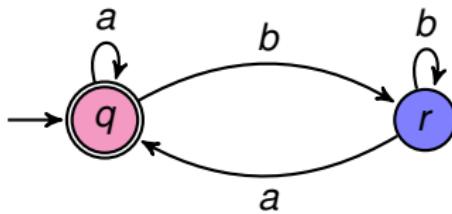
# Union and Intersection of NBA



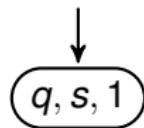
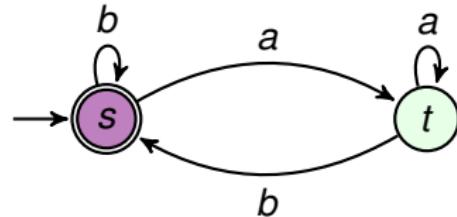
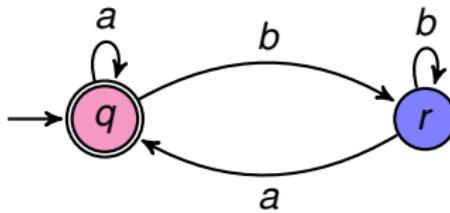
- ▶ States as  $Q_1 \times Q_2 \times \{1, 2\}$ , start state  $(q_0, q'_0, 1)$
- ▶  $(q_1, q_2, 1) \xrightarrow{a} (q'_1, q'_2, 1)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_1 \notin G_1$
- ▶  $(q_1, q_2, 1) \xrightarrow{a} (q'_1, q'_2, 2)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_1 \in G_1$
- ▶  $(q_1, q_2, 2) \xrightarrow{a} (q'_1, q'_2, 2)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_2 \notin G_2$
- ▶  $(q_1, q_2, 2) \xrightarrow{a} (q'_1, q'_2, 1)$  if  $q_1 \xrightarrow{a} q'_1$  and  $q_2 \xrightarrow{a} q'_2$  and  $q_2 \in G_2$
- ▶ Good states =  $Q_1 \times G_2 \times \{2\}$  or  $G_1 \times Q_2 \times \{1\}$

# Union and Intersection of NBA

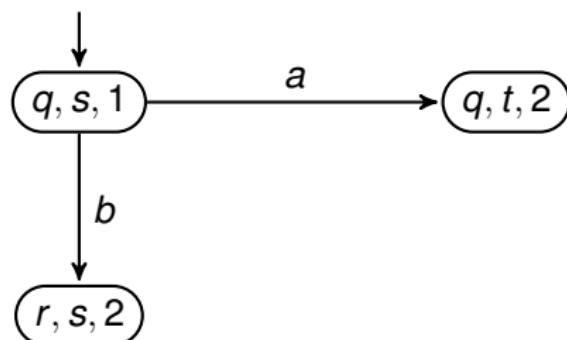
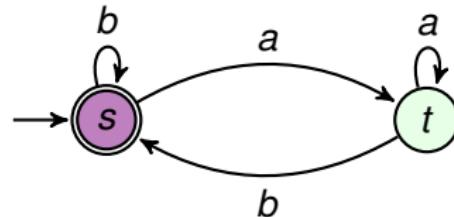
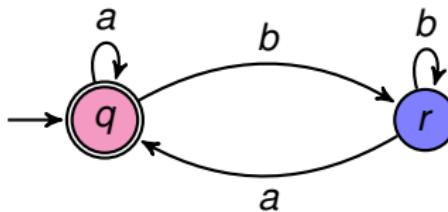
---



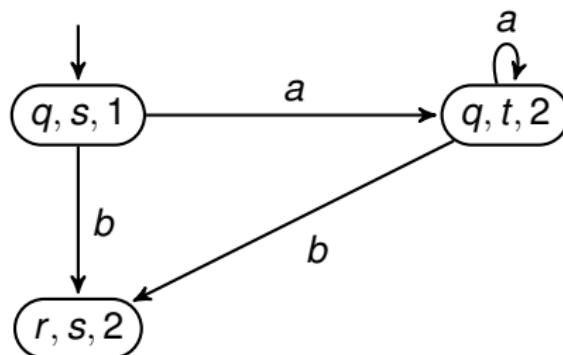
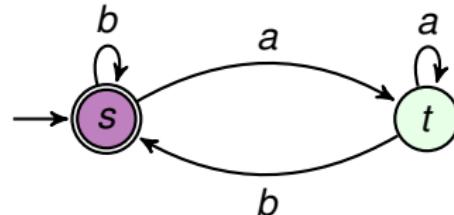
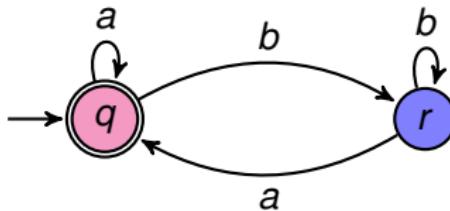
# Union and Intersection of NBA



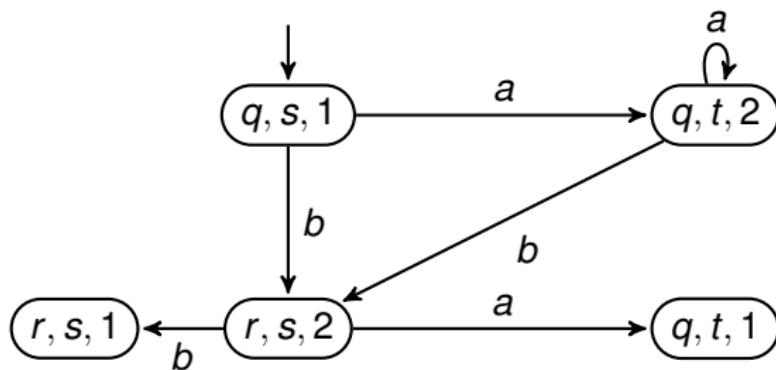
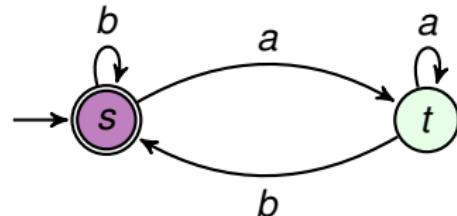
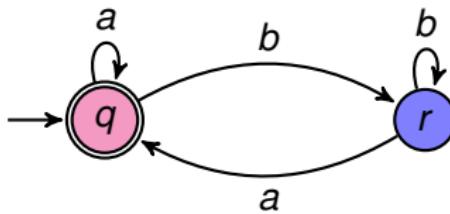
# Union and Intersection of NBA



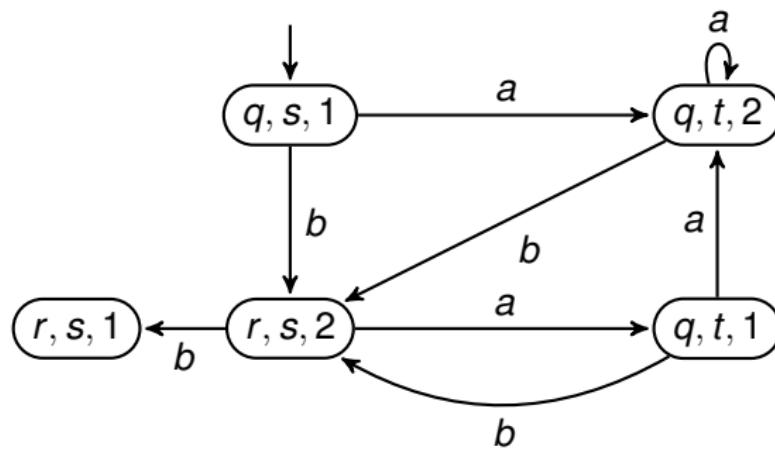
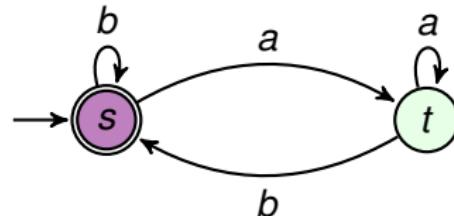
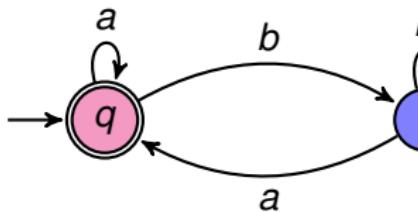
# Union and Intersection of NBA



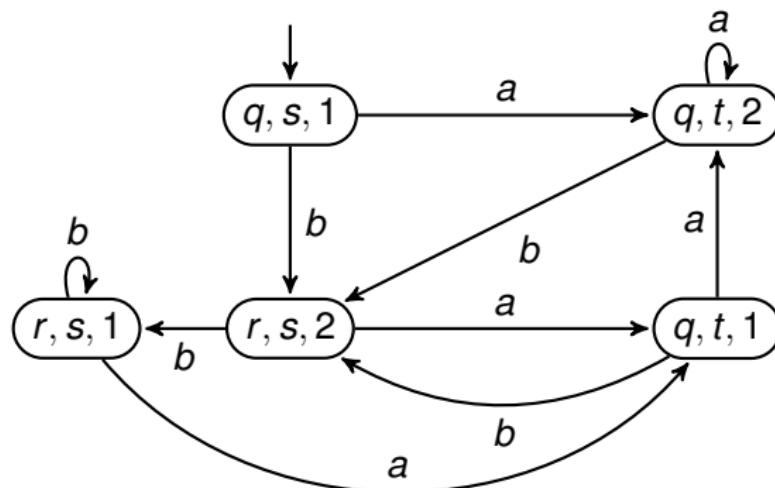
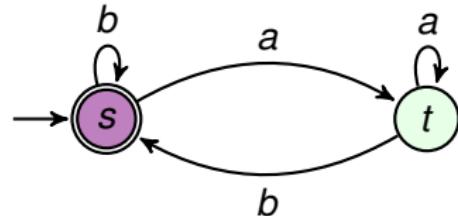
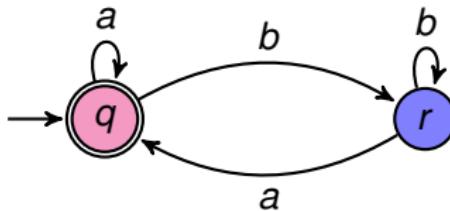
# Union and Intersection of NBA



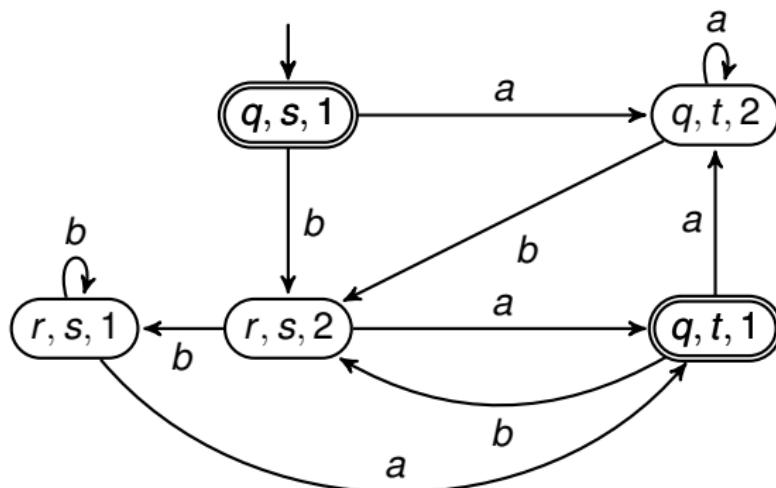
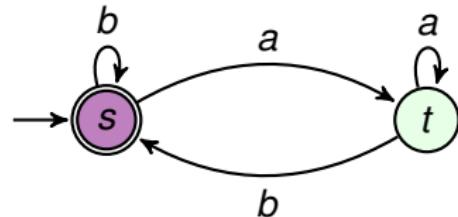
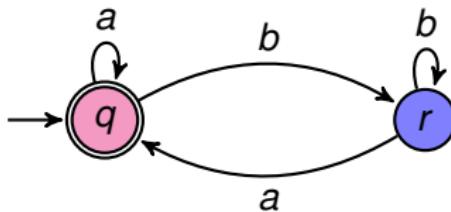
# Union and Intersection of NBA



# Union and Intersection of NBA



# Union and Intersection of NBA



# Emptiness

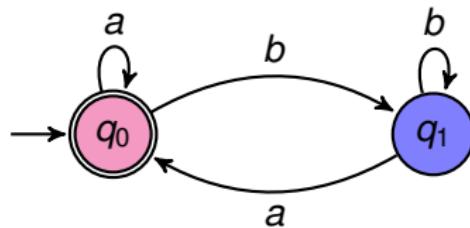
---

Given an NBA/DBA  $\mathcal{A}$ , how do you check if  $L(\mathcal{A}) = \emptyset$ ?

- ▶ Enumerate SCCs
- ▶ Check if there is an SCC containing a good state

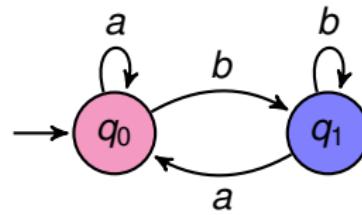
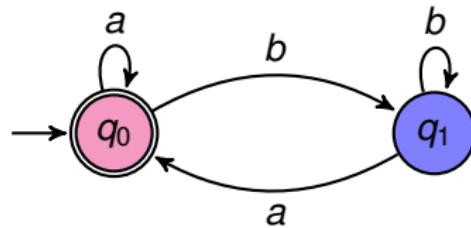
# Complementation of DBA

---



# Complementation of DBA

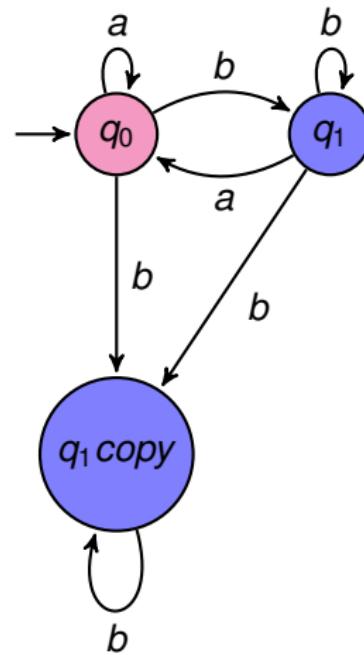
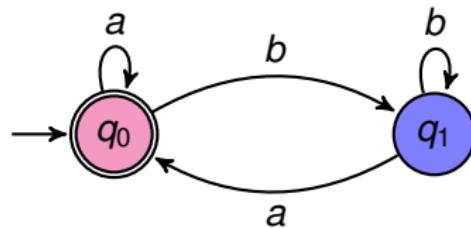
---



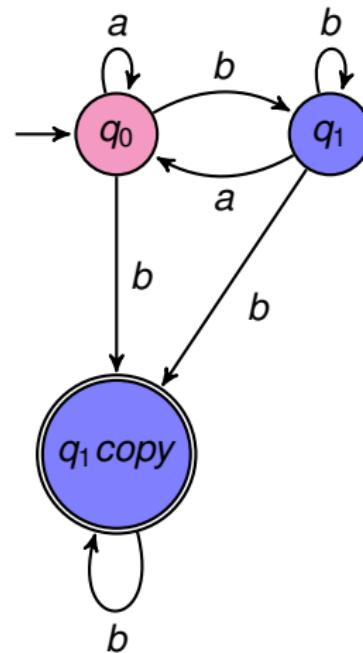
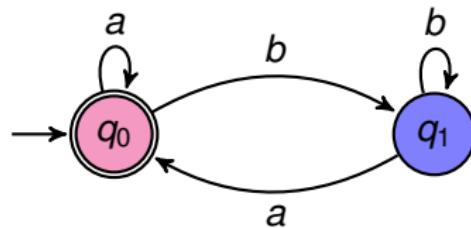
$q_1 \text{ copy}$

# Complementation of DBA

---



# Complementation of DBA



# Complementation of DBA

---

- ▶ Given  $\mathcal{A}$  is a DBA, and  $w \notin L(\mathcal{A})$ , then after some finite prefix, the unique run of  $w$  settles in bad states.
- ▶ Idea for complement: “copy” states of  $Q - G$ , once you enter this block, you stay there.
- ▶ View this as the set of good states, any word  $w$  that was rejected by  $\mathcal{A}$  has two possible runs in this automaton: the original run, and one another, that will settle in the  $Q - G$  copy, and will be accepted.
- ▶ What we get now is an NBA for  $\overline{L(\mathcal{A})}$ , not a DBA.

Complementing NBA non-trivial, can be done.

# Normal Form for $\omega$ -regular languages

An  $\omega$ -regular language  $L \subseteq \Sigma^\omega$  can be written as  $L = \bigcup_{i=1}^n U_i V_i^\omega$ , where  $U_i, V_i$  are regular languages.

One direction : Assume  $L$  is accepted by an NBA/DBA.

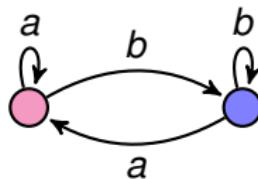
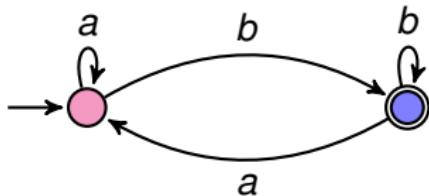
- ▶ Define  $U_g = \{w \in \Sigma^* \mid q_0 \xrightarrow{w} g\}$
- ▶ Define  $V_g = \{w \in \Sigma^* \mid g \xrightarrow{w} g\}$
- ▶ Then  $L = \bigcup_{g \in G} U_g V_g^\omega$ , where  $U_g, V_g$  are regular
- ▶ Show that  $U_g, V_g$  are regular.

# Normal Form for $\omega$ -regular languages

An  $\omega$ -regular language  $L \subseteq \Sigma^\omega$  can be written as  $L = \bigcup_{i=1}^n U_i V_i^\omega$ , where  $U_i, V_i$  are regular languages.

Other direction : Assume  $L = \bigcup_{i=1}^n U_i V_i^\omega$ . Show that  $L$  is accepted by an NBA/DBA.

1. If  $V$  is regular,  $V^\omega$  is  $\omega$ -regular

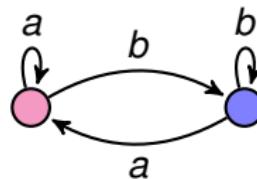
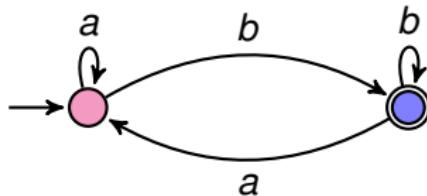


# Normal Form for $\omega$ -regular languages

An  $\omega$ -regular language  $L \subseteq \Sigma^\omega$  can be written as  $L = \bigcup_{i=1}^n U_i V_i^\omega$ , where  $U_i, V_i$  are regular languages.

Other direction : Assume  $L = \bigcup_{i=1}^n U_i V_i^\omega$ . Show that  $L$  is accepted by an NBA/DBA.

1. If  $V$  is regular,  $V^\omega$  is  $\omega$ -regular

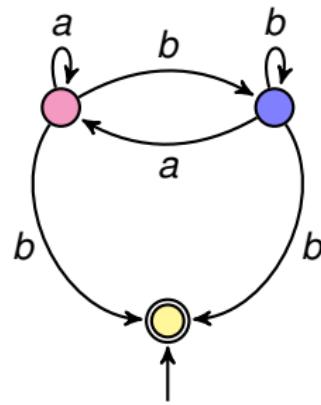
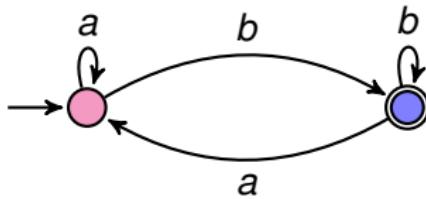


# Normal Form for $\omega$ -regular languages

An  $\omega$ -regular language  $L \subseteq \Sigma^\omega$  can be written as  $L = \bigcup_{i=1}^n U_i V_i^\omega$ , where  $U_i, V_i$  are regular languages.

Other direction : Assume  $L = \bigcup_{i=1}^n U_i V_i^\omega$ . Show that  $L$  is accepted by an NBA/DBA.

1. If  $V$  is regular,  $V^\omega$  is  $\omega$ -regular

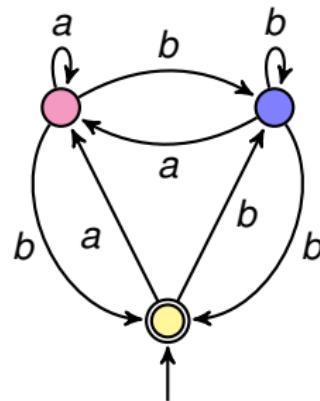
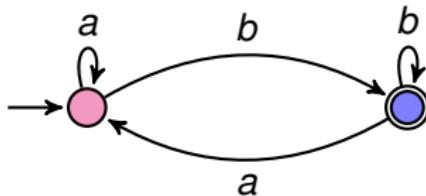


# Normal Form for $\omega$ -regular languages

An  $\omega$ -regular language  $L \subseteq \Sigma^\omega$  can be written as  $L = \bigcup_{i=1}^n U_i V_i^\omega$ , where  $U_i, V_i$  are regular languages.

Other direction : Assume  $L = \bigcup_{i=1}^n U_i V_i^\omega$ . Show that  $L$  is accepted by an NBA/DBA.

1. If  $V$  is regular,  $V^\omega$  is  $\omega$ -regular



# Normal Form for $\omega$ -regular languages

---

1. If  $V$  is regular,  $V^\omega$  is  $\omega$ -regular
  - ▶ Let  $D = (Q, \Sigma, q_0, \delta, F)$  be a DFA accepting  $V$
  - ▶ Construct NBA  $E = (Q \cup \{p_0\}, \Sigma, p_0, \Delta, G)$  such that  $G = \{p_0\}$ ,
  - ▶  $\Delta = \delta \cup \{p_0 \in \Delta(q, a) \mid \delta(q, a) \in F\} \cup \{\Delta(p_0, a) = s \mid \delta(q_0, a) = s\}$
2. Show that if  $U$  is regular and  $V^\omega$  is  $\omega$ -regular, then  $UV^\omega$  is  $\omega$ -regular
  - ▶  $D = (Q_1, \Sigma, q_0, \delta_1, F)$  be a DFA,  $L(D) = U$  and  
 $E = (Q_2, \Sigma, q'_0, \delta_2, G)$  be an NBA,  $L(E) = V^\omega$ .
  - ▶  $A = (Q_1 \cup Q_2, \Sigma, q_0, \delta', G)$  NBA such that  
 $\delta' = \delta_1 \cup \delta_2 \cup \{(q, a, q'_0) \mid \delta_1(q, a) \in F\}$

# Lecture 31.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Union of NBA/DBA

---

# Normal Form for $\omega$ -regular languages

An  $\omega$ -regular language  $L \subseteq \Sigma^\omega$  can be written as  $L = \bigcup_{i=1}^n U_i V_i^\omega$ , where  $U_i, V_i$  are regular languages.

One direction : Assume  $L$  is accepted by an NBA/DBA.

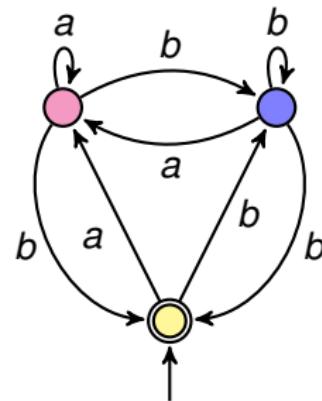
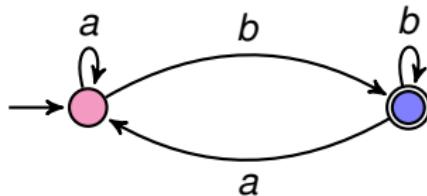
- ▶ Define  $U_g = \{w \in \Sigma^* \mid q_0 \xrightarrow{w} g\}$
- ▶ Define  $V_g = \{w \in \Sigma^* \mid g \xrightarrow{w} g\}$
- ▶ Then  $L = \bigcup_{g \in G} U_g V_g^\omega$ , where  $U_g, V_g$  are regular
- ▶ Show that  $U_g, V_g$  are regular.

# Normal Form for $\omega$ -regular languages

An  $\omega$ -regular language  $L \subseteq \Sigma^\omega$  can be written as  $L = \bigcup_{i=1}^n U_i V_i^\omega$ , where  $U_i, V_i$  are regular languages.

Other direction : Assume  $L = \bigcup_{i=1}^n U_i V_i^\omega$ . Show that  $L$  is accepted by an NBA/DBA.

1. If  $V$  is regular,  $V^\omega$  is  $\omega$ -regular



# Normal Form for $\omega$ -regular languages

---

1. If  $V$  is regular,  $V^\omega$  is  $\omega$ -regular
  - ▶ Let  $D = (Q, \Sigma, q_0, \delta, F)$  be a DFA accepting  $V$
  - ▶ Construct NBA  $E = (Q \cup \{p_0\}, \Sigma, p_0, \Delta, G)$  such that  $G = \{p_0\}$ ,
  - ▶  $\Delta = \delta \cup \{p_0 \in \Delta(q, a) \mid \delta(q, a) \in F\} \cup \{\Delta(p_0, a) = s \mid \delta(q_0, a) = s\}$
2. Show that if  $U$  is regular and  $V^\omega$  is  $\omega$ -regular, then  $UV^\omega$  is  $\omega$ -regular
  - ▶  $D = (Q_1, \Sigma, q_0, \delta_1, F)$  be a DFA,  $L(D) = U$  and  
 $E = (Q_2, \Sigma, q'_0, \delta_2, G)$  be an NBA,  $L(E) = V^\omega$ .
  - ▶  $A = (Q_1 \cup Q_2, \Sigma, q_0, \delta', G)$  NBA such that  
 $\delta' = \delta_1 \cup \delta_2 \cup \{(q, a, q'_0) \mid \delta_1(q, a) \in F\}$

# LTL ModelChecking

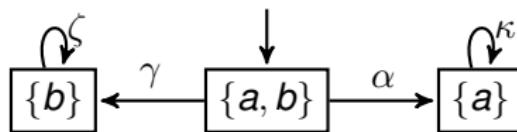
---

- ▶ Given transition system  $TS$ , and LTL formula  $\varphi$ , does  $TS \models \varphi$ ?
- ▶  $Tr(TS) \subseteq L(\varphi)$  iff  $Tr(TS) \cap \overline{L(\varphi)} = \emptyset$
- ▶ First construct NBA  $A_{\neg\varphi}$  for  $\neg\varphi$ .
- ▶ Construct product of  $TS$  and  $A_{\neg\varphi}$ , obtaining a new  $TS$ , say  $TS'$ .
- ▶ Check some very simple property on  $TS'$ , to check  $TS \models \varphi$ .

# An Example $TS \models \varphi$

---

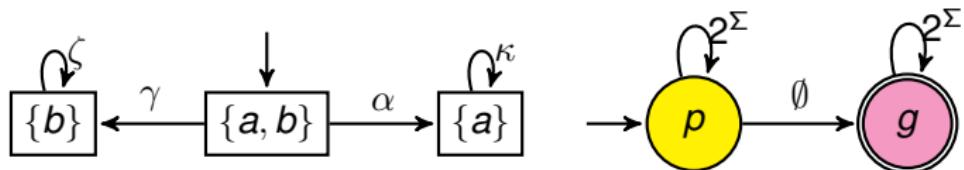
- ▶ Let  $\varphi = \square(a \vee b)$ ,  $\neg\varphi = \diamond(\neg a \wedge \neg b)$
- ▶ Take TS and  $A_{\neg\varphi}$ , and check the intersection.



# An Example $TS \models \varphi$

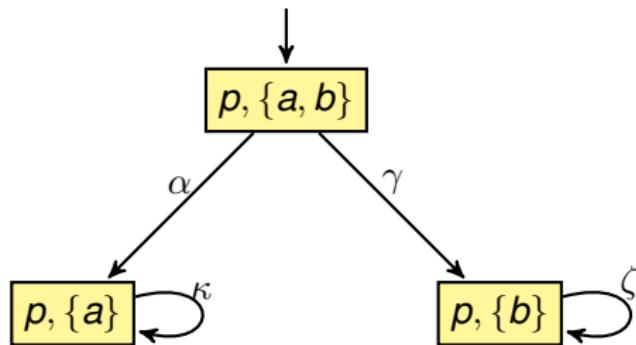
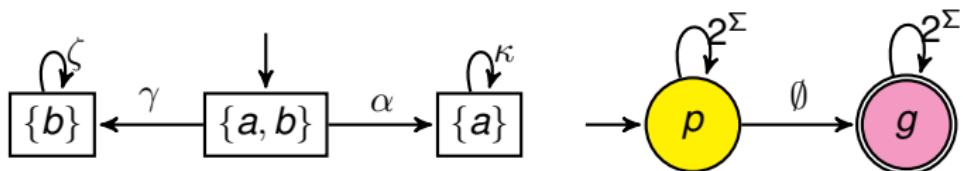
---

- ▶ Let  $\varphi = \square(a \vee b)$ ,  $\neg\varphi = \diamond(\neg a \wedge \neg b)$
- ▶ Take TS and  $A_{\neg\varphi}$ , and check the intersection.



# An Example $TS \models \varphi$

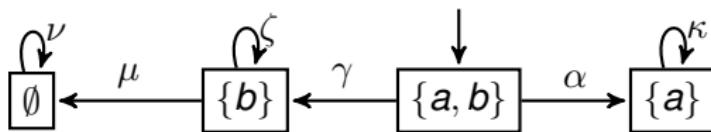
- Let  $\varphi = \square(a \vee b), \neg\varphi = \diamond(\neg a \wedge \neg b)$
- Take TS and  $A_{\neg\varphi}$ , and check the intersection.



# An Example : $TS \not\models \varphi$

---

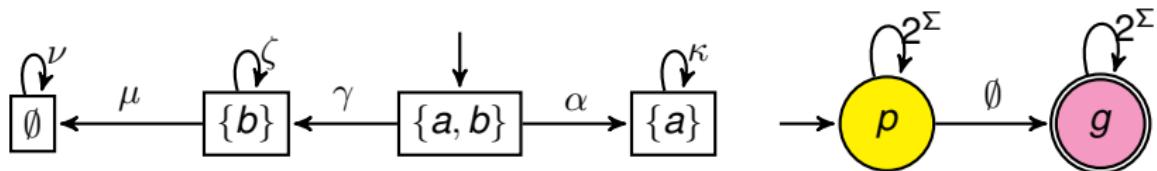
- ▶ Let  $\varphi = \square(a \vee b)$ ,  $\neg\varphi = \diamond(\neg a \wedge \neg b)$
- ▶ Take TS and  $A_{\neg\varphi}$ , and check the intersection.



# An Example : $TS \not\models \varphi$

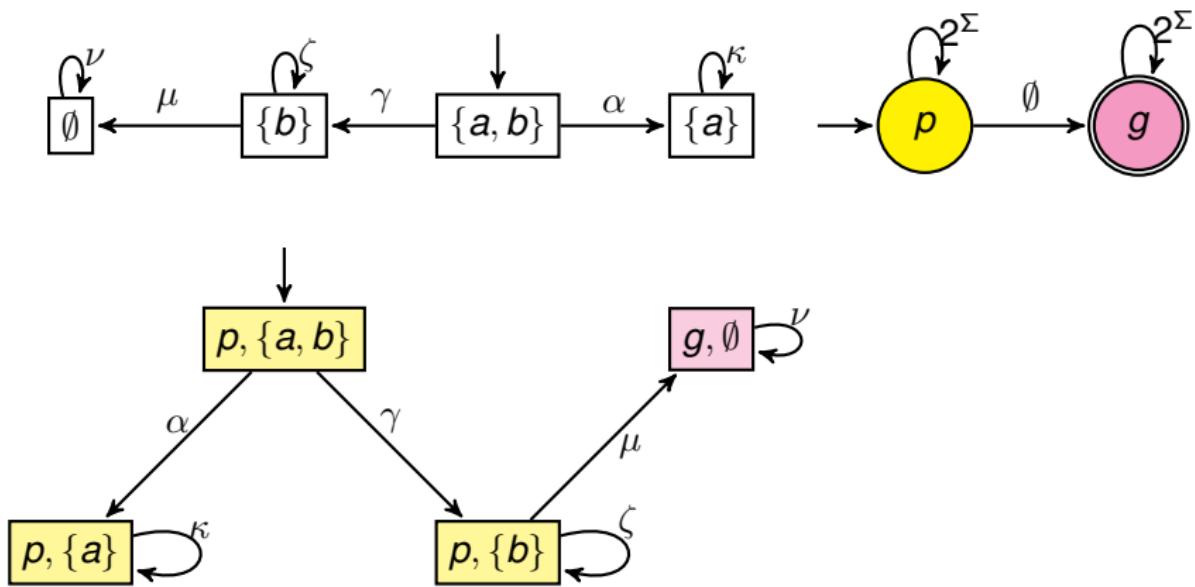
---

- Let  $\varphi = \square(a \vee b)$ ,  $\neg\varphi = \diamond(\neg a \wedge \neg b)$
- Take TS and  $A_{\neg\varphi}$ , and check the intersection.



# An Example : $TS \not\models \varphi$

- Let  $\varphi = \square(a \vee b)$ ,  $\neg\varphi = \diamond(\neg a \wedge \neg b)$
- Take TS and  $A_{\neg\varphi}$ , and check the intersection.



# Product of TS and NBA

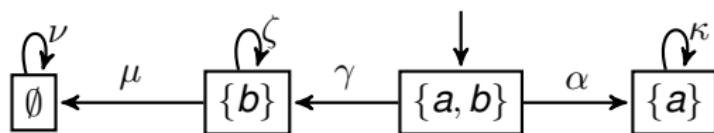
---

Given  $TS = (S, Act, I, AP, L)$  and  $\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, G)$  NBA.  
Define  $TS \otimes \mathcal{A} = (S \times Q, Act, I', AP', L')$  such that

- ▶  $I' = \{(s_0, q) \mid s_0 \in I \text{ and } \exists q_0 \in Q_0, q_0 \xrightarrow{L(s_0)} q\}$
- ▶  $AP' = Q$ ,  $L' : S \times Q \rightarrow 2^Q$  such that  $L'((s, q)) = \{q\}$
- ▶ If  $s \xrightarrow{\alpha} t$  and  $q \xrightarrow{L(t)} p$ , then  $(s, q) \xrightarrow{\alpha} (t, p)$

# Persistence Properties

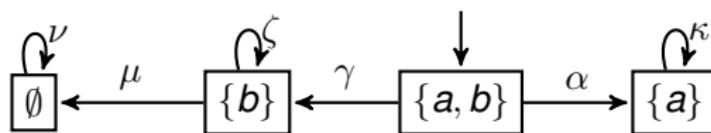
Let  $\eta$  be a propositional logic formula over  $AP$ . A persistence property  $P_{pers}$  has the form  $\diamond \square \eta$ . How will you check a persistence property on a TS?



- ▶ For example,  $TS \not\models \diamond \square(a \vee b)$
- ▶ For example,  $TS \models \diamond \square(a \vee (a \rightarrow b))$

# Persistence Properties

Let  $\eta$  be a propositional logic formula over  $AP$ . A persistence property  $P_{pers}$  has the form  $\diamond\Box\eta$ . How will you check a persistence property on a TS?



- ▶ For example,  $TS \not\models \diamond\Box(a \vee b)$
- ▶ For example,  $TS \models \diamond\Box(a \vee (a \rightarrow b))$
- ▶  $TS \not\models P_{pers}$  iff there is a reachable cycle in the TS containing a state with a label which satisfies  $\neg\eta$ .

# LTL ModelChecking

---

- ▶ Given  $TS$  and LTL formula  $\varphi$ . Does  $TS \models \varphi$ ?
- ▶ Construct  $A_{\neg\varphi}$ , and let  $g_1, \dots, g_n$  be the good states in  $A_{\neg\varphi}$ .
- ▶ Build  $TS' = TS \otimes A_{\neg\varphi}$ .
- ▶ The labels of  $TS'$  are the state names of  $A_{\neg\varphi}$ .
- ▶ Check if  $TS' \models \diamond\Box(\neg g_1 \wedge \dots \wedge \neg g_n)$ .

# LTL ModelChecking

- ▶ Given  $TS$  and LTL formula  $\varphi$ . Does  $TS \models \varphi$ ?
- ▶ Construct  $A_{\neg\varphi}$ , and let  $g_1, \dots, g_n$  be the good states in  $A_{\neg\varphi}$ .
- ▶ Build  $TS' = TS \otimes A_{\neg\varphi}$ .
- ▶ The labels of  $TS'$  are the state names of  $A_{\neg\varphi}$ .
- ▶ Check if  $TS' \models \diamond\Box(\neg g_1 \wedge \dots \wedge \neg g_n)$ .

ModelChecking LTL in  $TS$  = Check Persistence in  $TS'$

The following are equivalent.

- ▶  $TS \models \varphi$
- ▶  $Tr(TS) \cap L(A_{\neg\varphi}) = \emptyset$
- ▶  $TS' \models \diamond\Box(\neg g_1 \wedge \dots \wedge \neg g_n)$ .

# Lecture 32-34.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# GNBA

---

- ▶ Generalized NBA, a variant of NBA
- ▶ Only difference is in acceptance condition
- ▶ Acceptance condition in GNBA is a set  $\mathcal{F} = \{F_1, \dots, F_k\}$ , each  $F_i \subseteq Q$
- ▶ An infinite run  $\rho$  is accepting in a GNBA iff

$$\forall F_i \in \mathcal{F}, \text{Inf}(\rho) \cap F_i \neq \emptyset$$

- ▶ Note that when  $\mathcal{F} = \emptyset$ , all infinite runs are accepting
- ▶ GNBA and NBA are equivalent in expressive power.

# Word View (On the board)

---

- ▶  $w = \{a\}\{a, b\}\{\} \dots,$
- ▶  $\varphi = a \cup (\neg a \wedge b)$
- ▶ Subformulae of  $\varphi = \{a, \neg a, b, \neg a \wedge b, \varphi\}$
- ▶ Parse trees to compute all subformulae

# Closure of $\varphi$ , $cl(\varphi)$

---

- ▶  $cl(\varphi)$ =all subformulae of  $\varphi$  and their negations, identifying  $\neg\neg\psi$  to be  $\psi$ .
- ▶ Example for  $\varphi = a \cup (\neg a \wedge b)$
- ▶  $cl(\varphi) = \{a, \neg a, b, \neg b, \neg a \wedge b, \neg(\neg a \wedge b), \varphi, \neg\varphi\}$

# Elementary Sets

---

Let  $\varphi$  be an LTL formula. Then  $B \subseteq cl(\varphi)$  is elementary provided:

- ▶  $B$  is propositionally and maximally consistent : for all  $\varphi_1 \wedge \varphi_2, \psi \in cl(\varphi)$ ,
  - ▶  $\varphi_1 \wedge \varphi_2 \in B \Leftrightarrow \varphi_1 \in B \wedge \varphi_2 \in B$
  - ▶  $\psi \in B \Leftrightarrow \neg\psi \notin B$
  - ▶  $true \in cl(\varphi) \Rightarrow true \in B$
- ▶  $B$  is locally consistent wrt  $U$ . That is, for all  $\varphi_1 \mathbin{U} \varphi_2 \in cl(\varphi)$ ,
  - ▶  $\varphi_2 \in B \Rightarrow \varphi_1 \mathbin{U} \varphi_2 \in B$
  - ▶  $\varphi_1 \mathbin{U} \varphi_2 \in B, \varphi_2 \notin B \Rightarrow \varphi_1 \in B$
- ▶  $B$  is elementary :  $B$  is propositionally, maximally and locally consistent
- ▶ Given a  $B \subseteq cl(\varphi)$ , how can you check if  $B$  is elementary?

# Check Elementary

---

Let  $\varphi = a \cup (\neg a \wedge b)$

- $B_1 = \{a, b, \neg a \wedge b, \varphi\}$

# Check Elementary

---

Let  $\varphi = a \cup (\neg a \wedge b)$

- ▶  $B_1 = \{a, b, \neg a \wedge b, \varphi\}$  No, propositionally inconsistent
- ▶  $B_2 = \{\neg a, b, \varphi\}$

# Check Elementary

---

Let  $\varphi = a \cup (\neg a \wedge b)$

- ▶  $B_1 = \{a, b, \neg a \wedge b, \varphi\}$  No, propositionally inconsistent
- ▶  $B_2 = \{\neg a, b, \varphi\}$  No, not maximal as  $\neg a \wedge b \notin B_2$ ,  $\neg(\neg a \wedge b) \notin B_2$
- ▶  $B_3 = \{\neg a, b, \neg a \wedge b, \neg \varphi\}$

# Check Elementary

---

Let  $\varphi = a \cup (\neg a \wedge b)$

- ▶  $B_1 = \{a, b, \neg a \wedge b, \varphi\}$  No, propositionally inconsistent
- ▶  $B_2 = \{\neg a, b, \varphi\}$  No, not maximal as  $\neg a \wedge b \notin B_2$ ,  $\neg(\neg a \wedge b) \notin B_2$
- ▶  $B_3 = \{\neg a, b, \neg a \wedge b, \neg \varphi\}$  No, not locally consistent for  $\cup$
- ▶  $B_4 = \{\neg a, \neg b, \neg(\neg a \wedge b), \neg \varphi\}$

# Check Elementary

---

Let  $\varphi = a \cup (\neg a \wedge b)$

- ▶  $B_1 = \{a, b, \neg a \wedge b, \varphi\}$  No, propositionally inconsistent
- ▶  $B_2 = \{\neg a, b, \varphi\}$  No, not maximal as  $\neg a \wedge b \notin B_2$ ,  $\neg(\neg a \wedge b) \notin B_2$
- ▶  $B_3 = \{\neg a, b, \neg a \wedge b, \neg \varphi\}$  No, not locally consistent for  $\cup$
- ▶  $B_4 = \{\neg a, \neg b, \neg(\neg a \wedge b), \neg \varphi\}$  Yes, elementary

# LTL $\varphi$ to GNBA $G_\varphi$

---

- ▶ States of  $G_\varphi$  are elementary sets  $B_i$
- ▶ For a word  $w = A_0A_1A_2\dots$  the sequence of states  $\sigma = B_0B_1B_2\dots$  will be a run for  $w$
- ▶  $\sigma$  will be accepting iff  $w \models \varphi$  iff  $\varphi \in B_0$
- ▶ In general, a run  $B_iB_{i+1}\dots$  for  $A_iA_{i+1}\dots$  is accepting iff  $A_iA_{i+1}\dots \models \psi$  for all  $\psi \in B_i$ .

# LTL to GNBA

---

- ▶ Let  $\varphi = \bigcirc a$ .
- ▶ Subformulae of  $\varphi$  :  $\{a, \bigcirc a\}$ . Let  $A = \{a, \bigcirc a, \neg a, \neg \bigcirc a\}$ .
- ▶ Possibilities at each state
  - ▶  $\{a, \bigcirc a\}$
  - ▶  $\{\neg a, \bigcirc a\}$
  - ▶  $\{a, \neg \bigcirc a\}$
  - ▶  $\{\neg a, \neg \bigcirc a\}$
- ▶ Our initial state(s) must guarantee truth of  $\bigcirc a$ . Thus, initial states:  $\{a, \bigcirc a\}$  and  $\{\neg a, \bigcirc a\}$

# LTL to GNBA

---

$\{a, \bigcirc a\}$

$\{a, \neg \bigcirc a\}$

$\{\neg a, \bigcirc a\}$

$\{\neg a, \neg \bigcirc a\}$

# LTL to GNBA

---

→  $\{a, \bigcirc a\}$

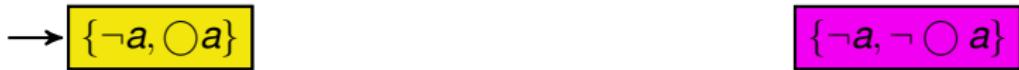
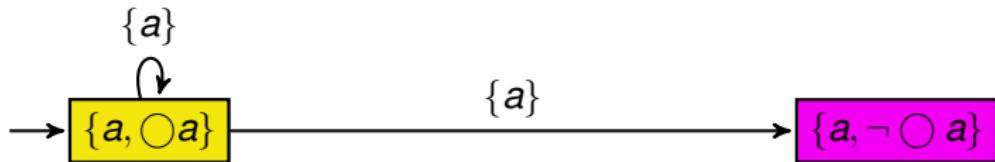
$\{a, \neg \bigcirc a\}$

→  $\{\neg a, \bigcirc a\}$

$\{\neg a, \neg \bigcirc a\}$

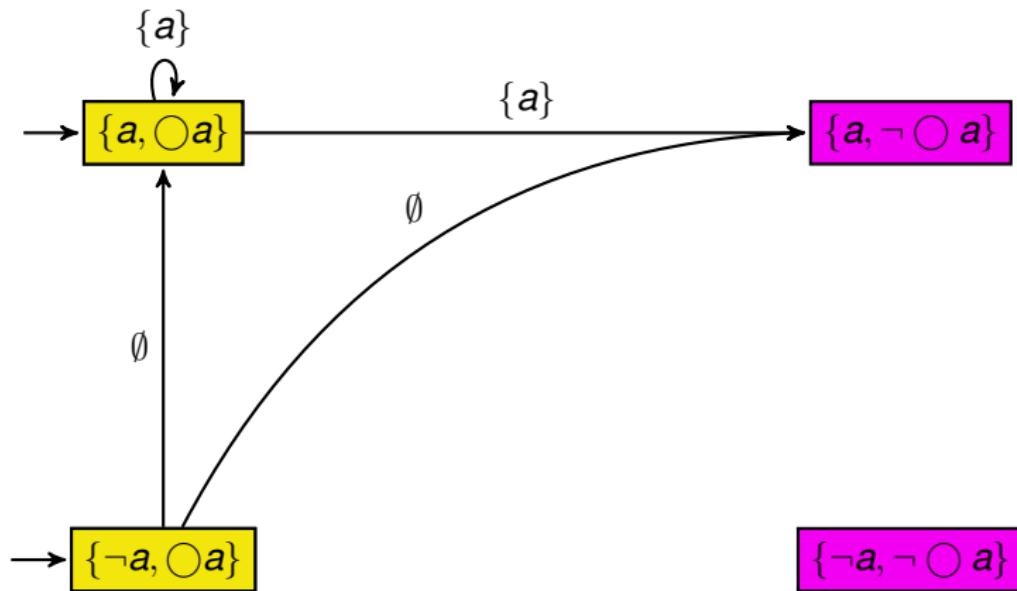
# LTL to GNBA

---



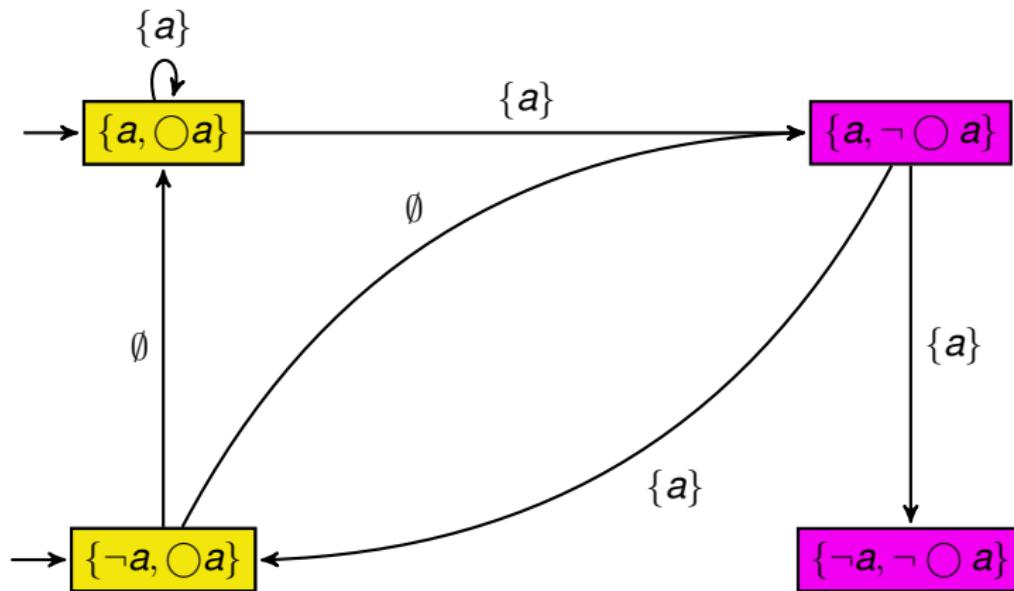
# LTL to GNBA

---

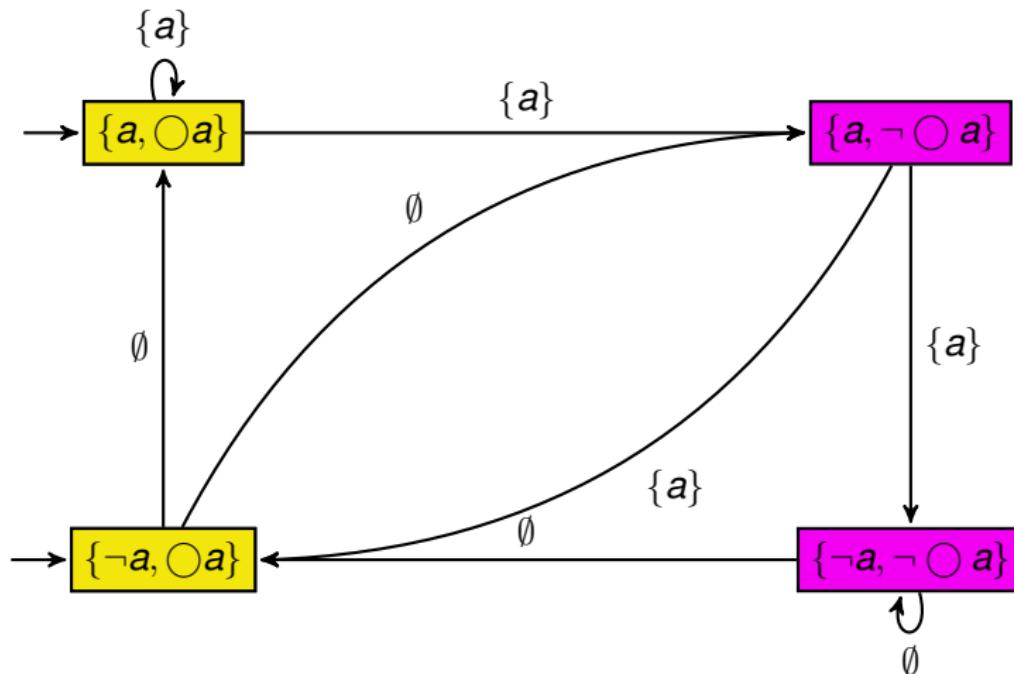


# LTL to GNBA

---



# LTL to GNBA



# LTL to GNBA

---

- ▶ Claim : Runs from a state labelled set  $B$  indeed satisfy  $B$
- ▶ No good states. All words having a run from a start state are accepted.
- ▶ Automaton for  $\neg \bigcirc a$  same, except for the start states.

# LTL to GNBA

---

- ▶ Let  $\varphi = a \mathbf{U} b$ .
- ▶ Subformulae of  $\varphi$  :  $\{a, b, a \mathbf{U} b\}$ . Let  $B = \{a, \neg a, b, \neg b, a \mathbf{U} b, \neg(a \mathbf{U} b)\}$ .
- ▶ Possibilities at each state
  - ▶  $\{a, \neg b, a \mathbf{U} b\}$
  - ▶  $\{\neg a, b, a \mathbf{U} b\}$
  - ▶  $\{a, b, a \mathbf{U} b\}$
  - ▶  $\{a, \neg b, \neg(a \mathbf{U} b)\}$
  - ▶  $\{\neg a, \neg b, \neg(a \mathbf{U} b)\}$
- ▶ Our initial state(s) must guarantee truth of  $a \mathbf{U} b$ . Thus, initial states:  $\{a, b, a \mathbf{U} b\}$  and  $\{\neg a, b, a \mathbf{U} b\}$  and  $\{a, \neg b, a \mathbf{U} b\}$ .

# LTL to GNBA

---

→  $\{a, b, a \mathbf{U} b\}$

$\{a, \neg b, \neg(a \mathbf{U} b)\}$

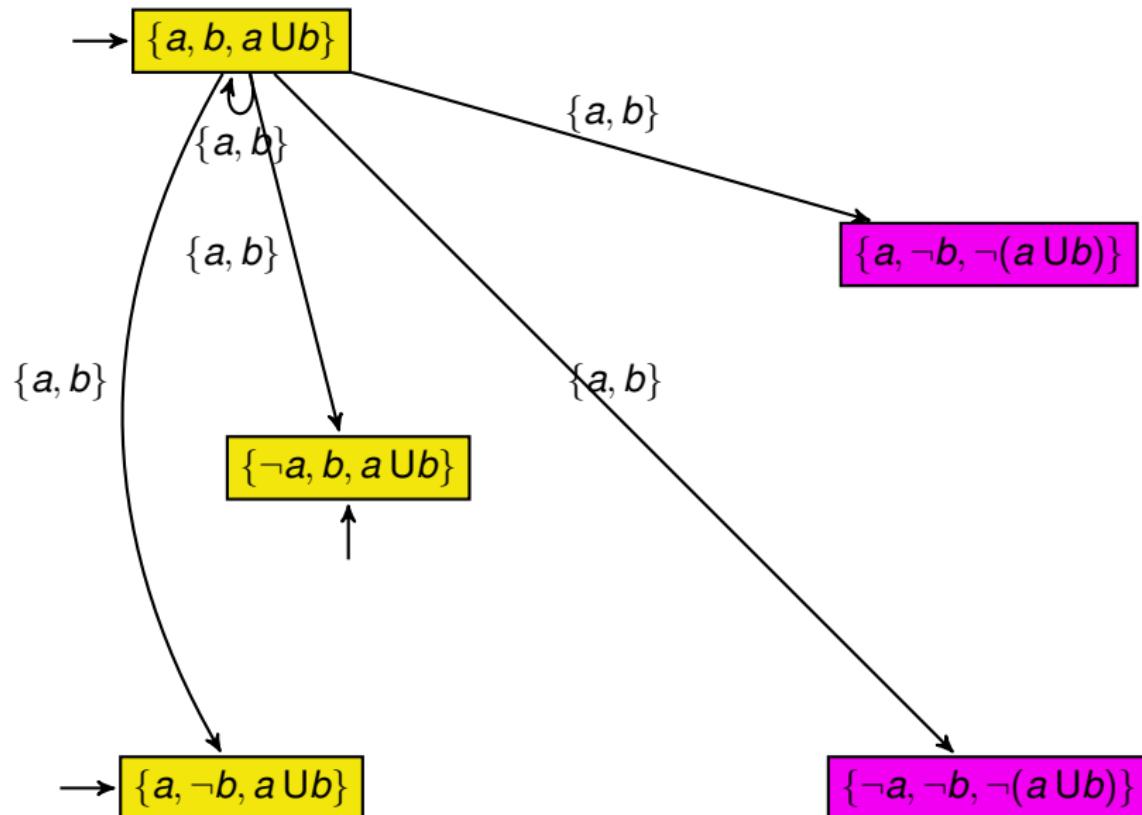
$\{\neg a, b, a \mathbf{U} b\}$



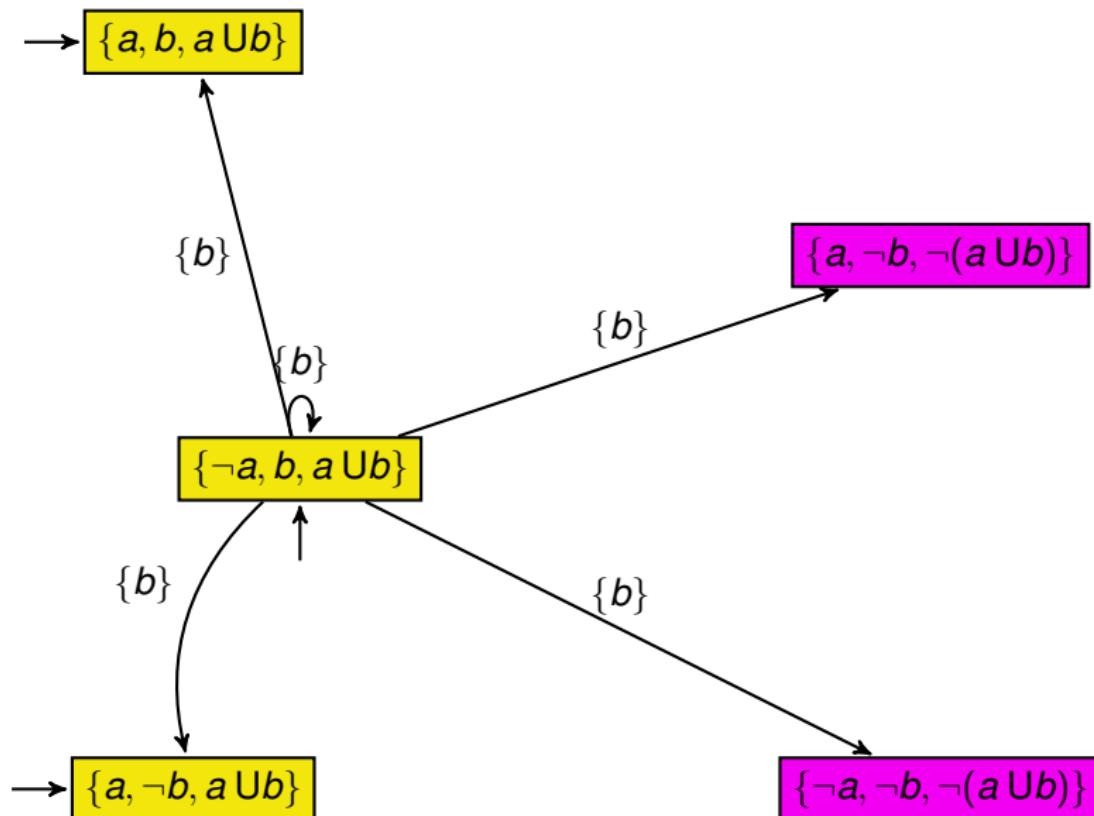
→  $\{a, \neg b, a \mathbf{U} b\}$

$\{\neg a, \neg b, \neg(a \mathbf{U} b)\}$

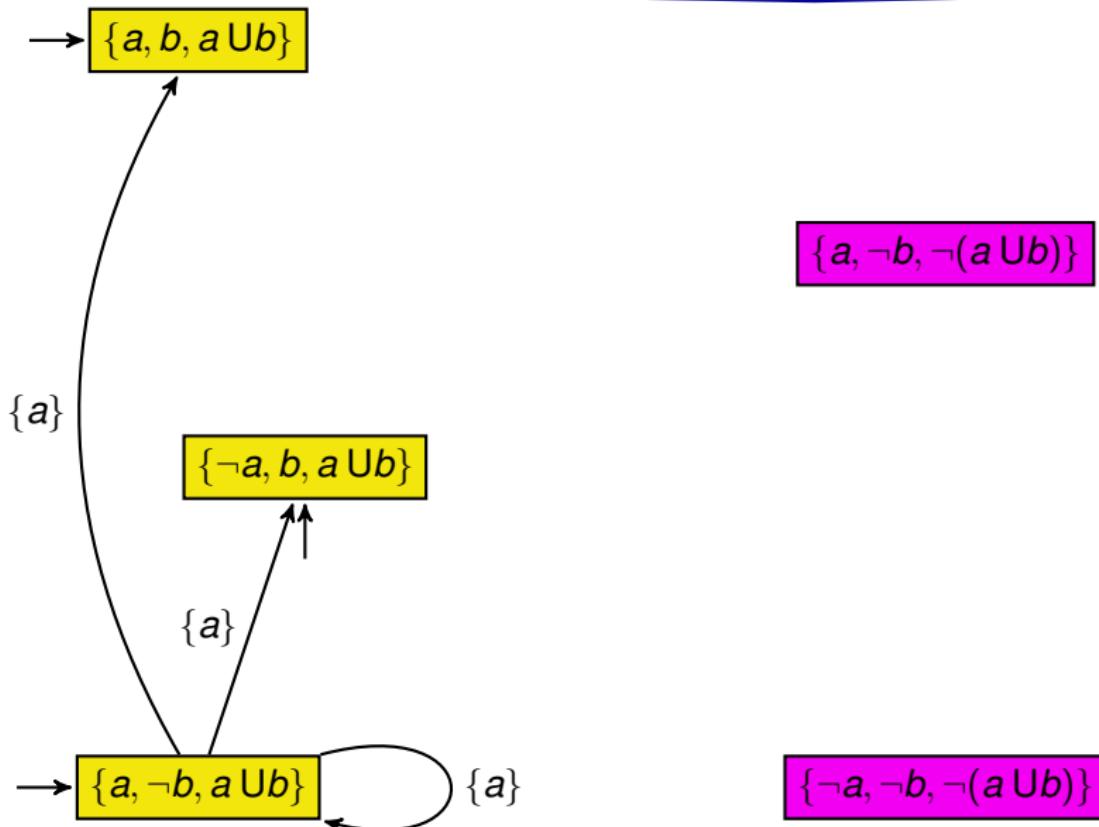
# LTL to GNBA



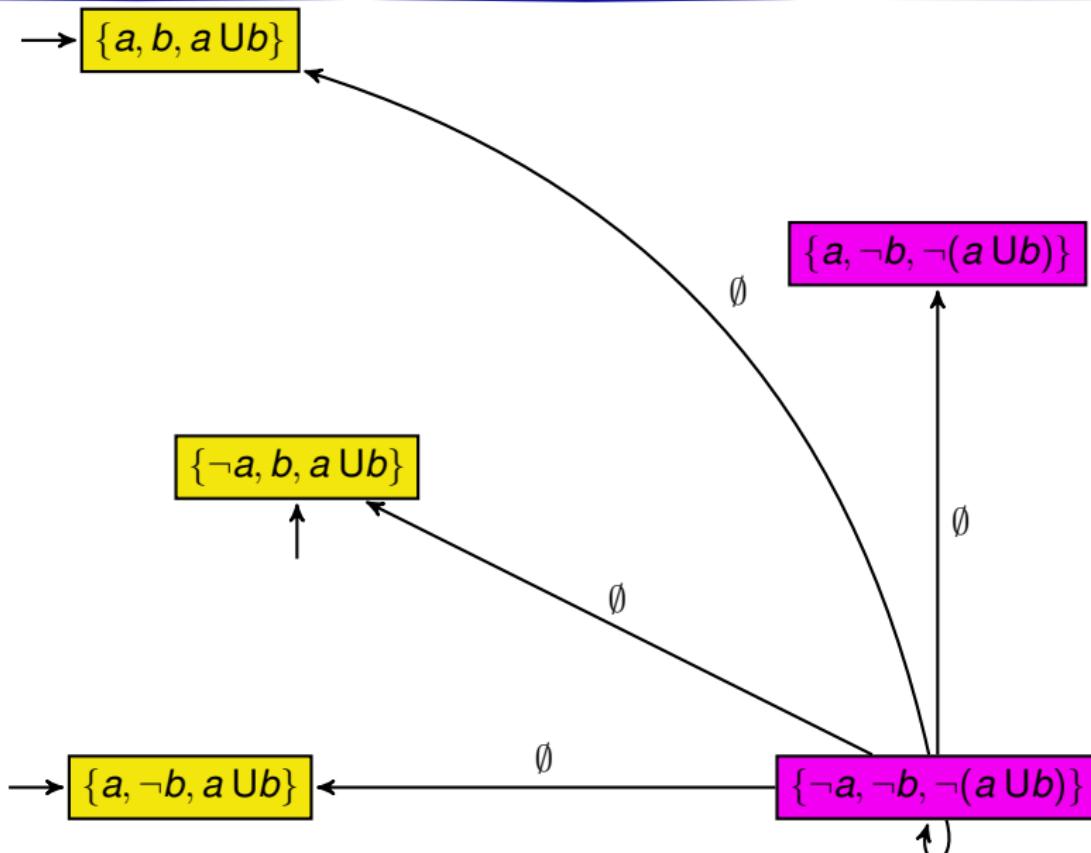
# LTL to GNBA



# LTL to GNBA



# LTL to GNBA



# LTL to GNBA

→  $\{a, b, a \mathbf{U} b\}$

→  $\{a, \neg b, a \mathbf{U} b\}$

$\{\neg a, b, a \mathbf{U} b\}$

$\{a\}$   
↓  
 $\{a, \neg b, \neg(a \mathbf{U} b)\}$

$\{\neg a, \neg b, \neg(a \mathbf{U} b)\}$

# LTL to GNBA : Accepting States

---

→  $\{a, b, a \mathbf{U} b\}$

$\{a, \neg b, \neg(a \mathbf{U} b)\}$

$\{\neg a, b, a \mathbf{U} b\}$



→  $\{a, \neg b, a \mathbf{U} b\}$

$\{\neg a, \neg b, \neg(a \mathbf{U} b)\}$

# LTL to GNBA

---

Construct GNBA for  $\neg(a \mathbf{U} b)$ .

# LTL to GNBA

---

- ▶ Let  $\varphi = a \mathsf{U} (\neg a \mathsf{U} c)$ . Let  $\psi = \neg a \mathsf{U} c$
- ▶ Subformulae of  $\varphi$  :  $\{a, \neg a, c, \psi, \varphi\}$ . Let  $B = \{a, \neg a, c, \neg c, \psi, \neg \psi, \varphi, \neg \varphi\}$ .
- ▶ Possibilities at each state
  - ▶  $\{a, c, \psi, \varphi\}$
  - ▶  $\{\neg a, c, \psi, \varphi\}$
  - ▶  $\{a, \neg c, \neg \psi, \varphi\}$
  - ▶  $\{a, \neg c, \neg \psi, \neg \varphi\}$
  - ▶  $\{\neg a, \neg c, \psi, \varphi\}$
  - ▶  $\{\neg a, \neg c, \neg \psi, \neg \varphi\}$

# LTL to GNBA

---

→  $\{a, c, \psi, \varphi\}$

$\{\neg a, \neg c, \psi, \varphi\}$  ←

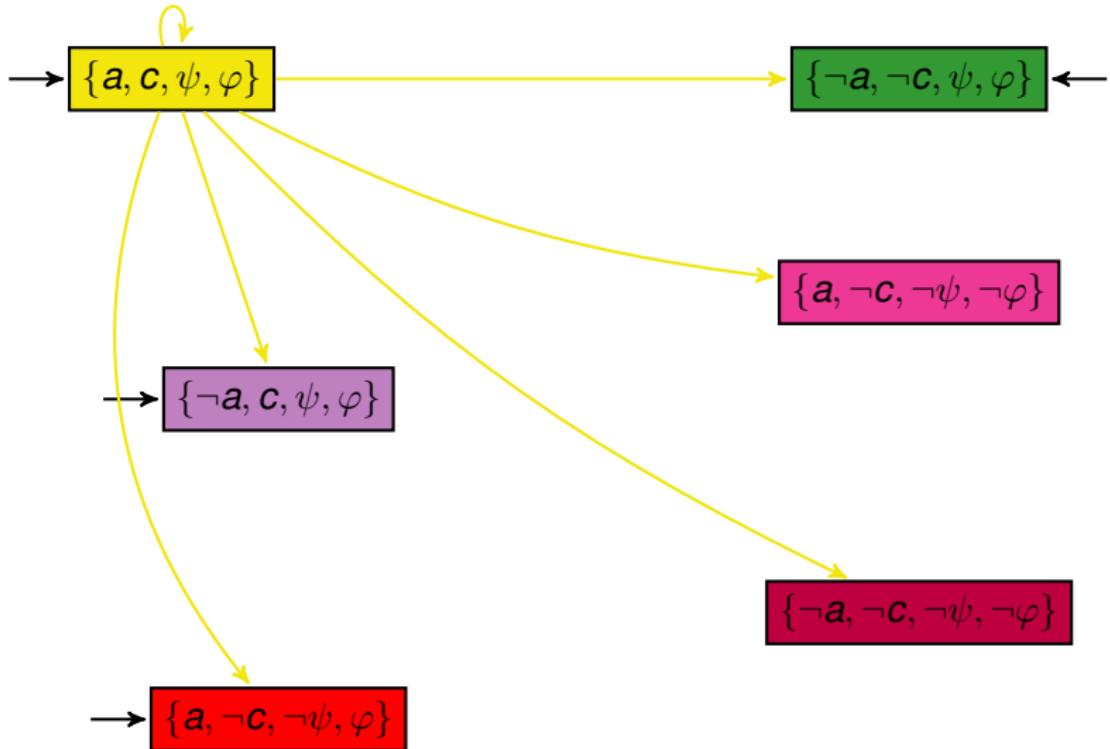
$\{a, \neg c, \neg \psi, \neg \varphi\}$

→  $\{\neg a, c, \psi, \varphi\}$

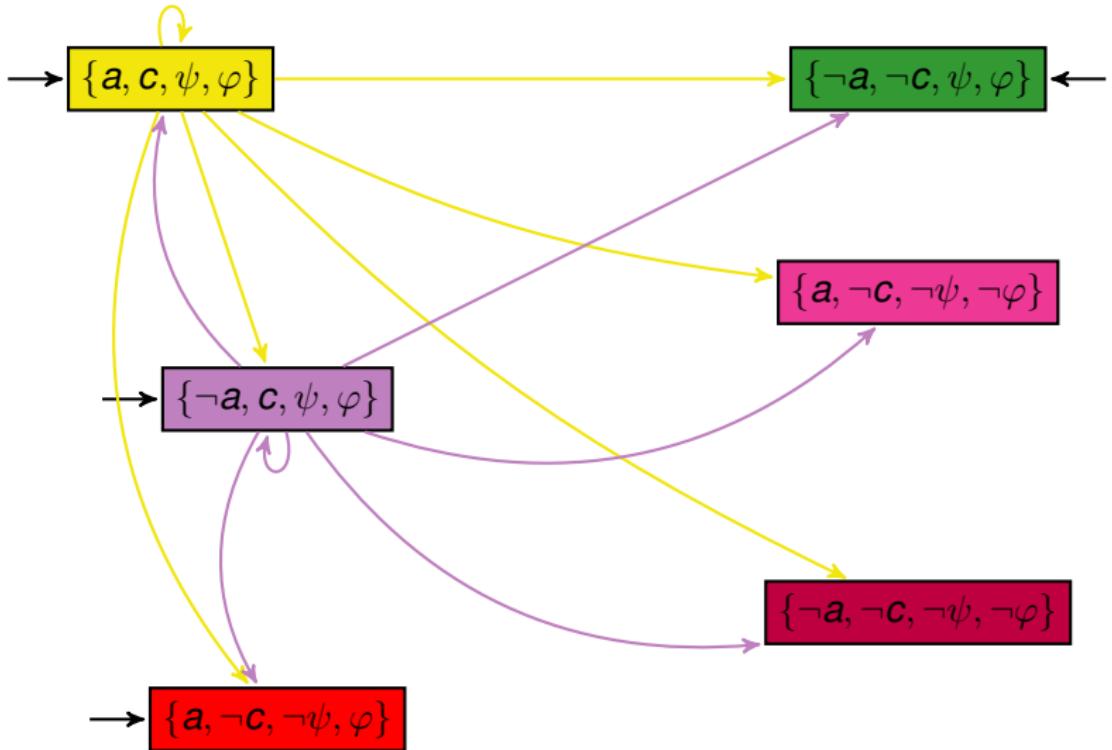
$\{\neg a, \neg c, \neg \psi, \neg \varphi\}$

→  $\{a, \neg c, \neg \psi, \varphi\}$

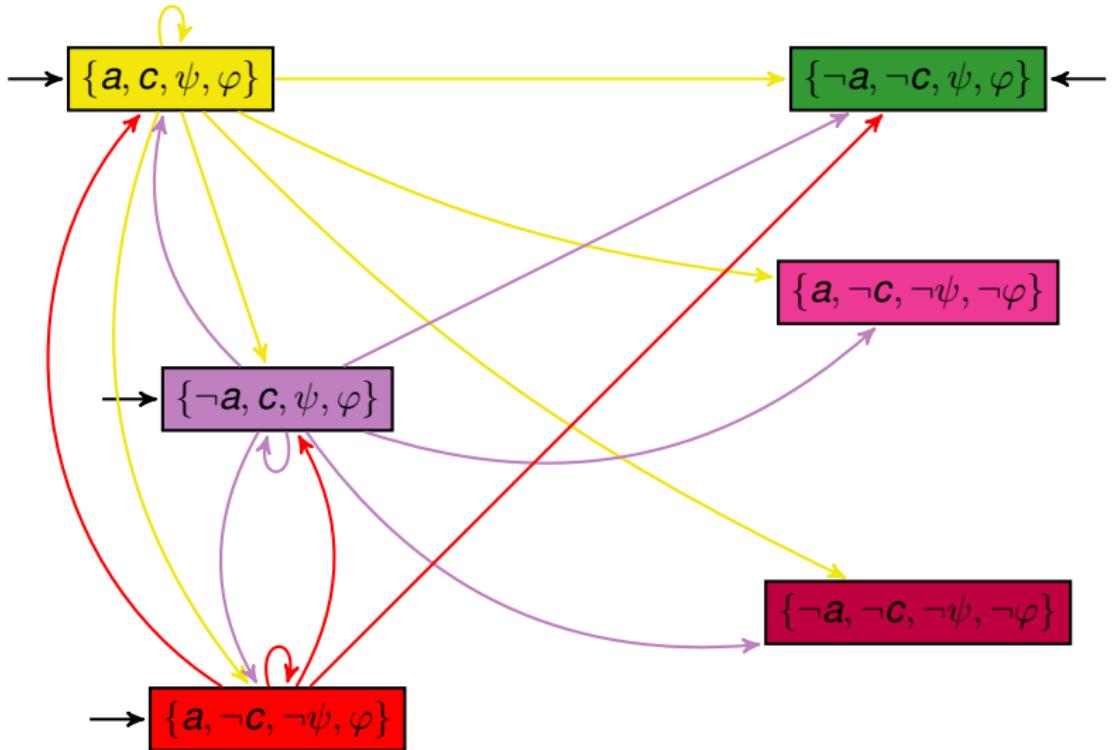
# LTL to GNBA



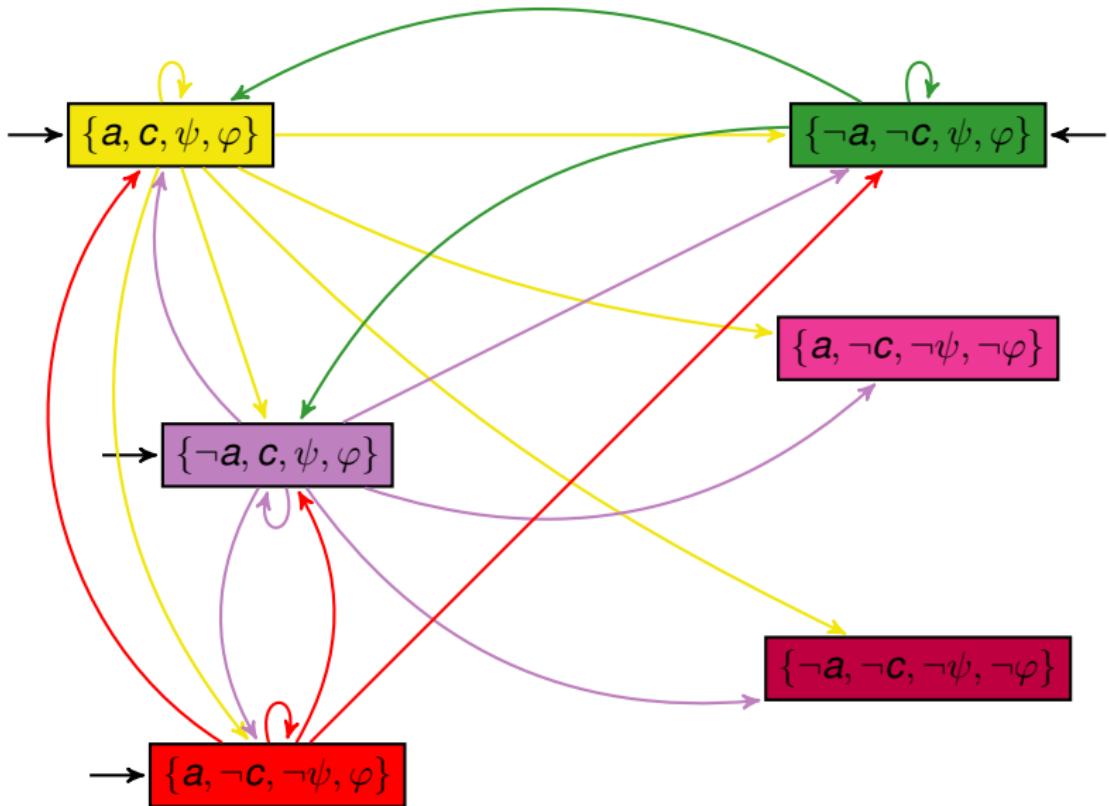
# LTL to GNBA



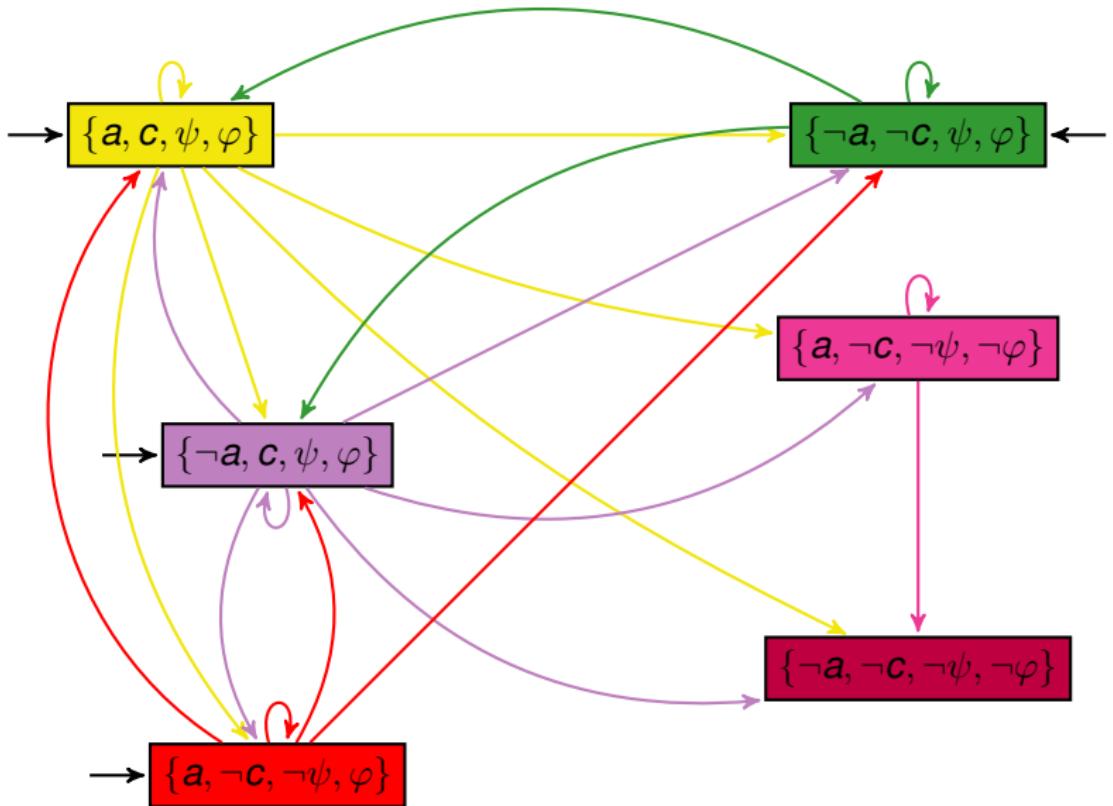
# LTL to GNBA



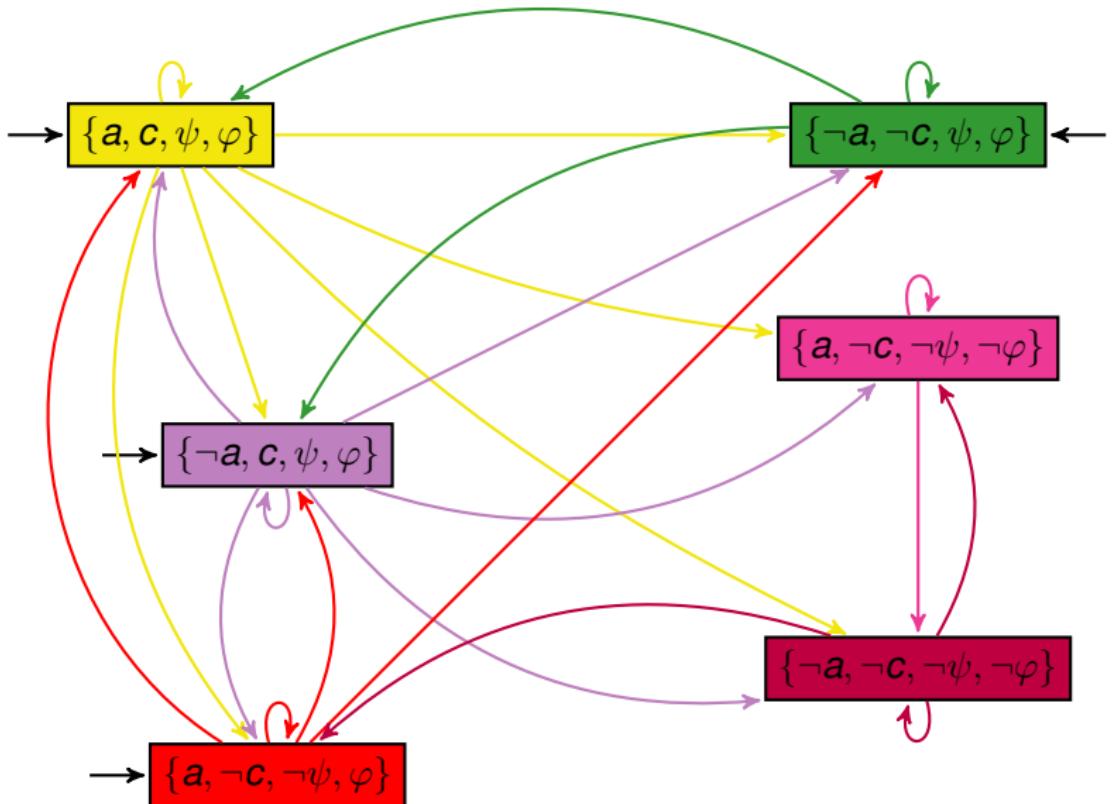
# LTL to GNBA



# LTL to GNBA



# LTL to GNBA



# GNBA Acceptance Condition

---

- ▶  $\psi = \neg a \mathbf{U} c$
- ▶  $\varphi = a \mathbf{U} \psi$
- ▶  $F_1 = \{B \mid \psi \in B \rightarrow c \in B\}$
- ▶  $F_2 = \{B \mid \varphi \in B \rightarrow \psi \in B\}$
- ▶  $\mathcal{F} = \{F_1, F_2\}$

# Final States

---

$$\rightarrow \{a, c, \psi, \varphi\} \in F_1, F_2$$

$$\{\neg a, \neg c, \psi, \varphi\} \in F_2 \leftarrow$$

$$\{a, \neg c, \neg \psi, \neg \varphi\} \in F_1, F_2$$

$$\rightarrow \{\neg a, c, \psi, \varphi\} \in F_1, F_2$$

$$\{\neg a, \neg c, \neg \psi, \neg \varphi\} \in F_1, F_2$$

$$\rightarrow \{a, \neg c, \neg \psi, \varphi\} \in F_1$$

# Putting Together

---

- ▶ Given  $\varphi$ , build  $CI(\varphi)$ , the set of all subformulae of  $\varphi$  and their negations

# Putting Together

---

- ▶ Given  $\varphi$ , build  $Cl(\varphi)$ , the set of all subformulae of  $\varphi$  and their negations
- ▶ Consider those  $B \subseteq Cl(\varphi)$  which are **consistent**
  - ▶  $\varphi_1 \wedge \varphi_2 \in B \leftrightarrow \varphi_1 \in B \text{ and } \varphi_2 \in B$

# Putting Together

---

- ▶ Given  $\varphi$ , build  $Cl(\varphi)$ , the set of all subformulae of  $\varphi$  and their negations
- ▶ Consider those  $B \subseteq Cl(\varphi)$  which are **consistent**
  - ▶  $\varphi_1 \wedge \varphi_2 \in B \leftrightarrow \varphi_1 \in B \text{ and } \varphi_2 \in B$
  - ▶  $\psi \in B \rightarrow \neg\psi \notin B \text{ and } \psi \notin B \rightarrow \neg\psi \in B$

# Putting Together

---

- ▶ Given  $\varphi$ , build  $Cl(\varphi)$ , the set of all subformulae of  $\varphi$  and their negations
- ▶ Consider those  $B \subseteq Cl(\varphi)$  which are **consistent**
  - ▶  $\varphi_1 \wedge \varphi_2 \in B \leftrightarrow \varphi_1 \in B \text{ and } \varphi_2 \in B$
  - ▶  $\psi \in B \rightarrow \neg\psi \notin B \text{ and } \psi \notin B \rightarrow \neg\psi \in B$
  - ▶ Whenever  $\psi_1 \cup \psi_2 \in Cl(\varphi)$ ,
    - ▶  $\psi_2 \in B \rightarrow \psi_1 \cup \psi_2 \in B$
    - ▶  $\psi_1 \cup \psi_2 \in B \text{ and } \psi_2 \notin B \rightarrow \psi_1 \in B$

# Putting Together

---

Given  $\varphi$  over  $AP$ , construct  $A_\varphi = (Q, 2^{AP}, \delta, Q_0, \mathcal{F})$ ,

- ▶  $Q = \{B \mid B \subseteq CI(\varphi) \text{ is consistent}\}$
- ▶  $Q_0 = \{B \mid \varphi \in B\}$
- ▶  $\delta : Q \times 2^{AP} \rightarrow 2^Q$  is such that

# Putting Together

---

Given  $\varphi$  over  $AP$ , construct  $A_\varphi = (Q, 2^{AP}, \delta, Q_0, \mathcal{F})$ ,

- ▶  $Q = \{B \mid B \subseteq Cl(\varphi) \text{ is consistent}\}$
- ▶  $Q_0 = \{B \mid \varphi \in B\}$
- ▶  $\delta : Q \times 2^{AP} \rightarrow 2^Q$  is such that
  - ▶ For  $C = B \cap AP$ ,  $\delta(B, C)$  is enabled and is defined as :
  - ▶ If  $\bigcirc\psi \in Cl(\varphi)$ ,  $\bigcirc\psi \in B$  iff  $\psi \in \delta(B, C)$
  - ▶ If  $\varphi_1 \cup \varphi_2 \in Cl(\varphi)$ ,  
 $\varphi_1 \cup \varphi_2 \in B$  iff  $(\varphi_2 \in B \vee (\varphi_1 \in B \wedge \varphi_1 \cup \varphi_2 \in \delta(B, C)))$

# Putting Together

---

Given  $\varphi$  over  $AP$ , construct  $A_\varphi = (Q, 2^{AP}, \delta, Q_0, \mathcal{F})$ ,

- ▶  $Q = \{B \mid B \subseteq Cl(\varphi) \text{ is consistent}\}$
- ▶  $Q_0 = \{B \mid \varphi \in B\}$
- ▶  $\delta : Q \times 2^{AP} \rightarrow 2^Q$  is such that
  - ▶ For  $C = B \cap AP$ ,  $\delta(B, C)$  is enabled and is defined as :
  - ▶ If  $\bigcirc\psi \in Cl(\varphi)$ ,  $\bigcirc\psi \in B$  iff  $\psi \in \delta(B, C)$
  - ▶ If  $\varphi_1 \cup \varphi_2 \in Cl(\varphi)$ ,  
 $\varphi_1 \cup \varphi_2 \in B$  iff  $(\varphi_2 \in B \vee (\varphi_1 \in B \wedge \varphi_1 \cup \varphi_2 \in \delta(B, C)))$
- ▶  $\mathcal{F} = \{F_{\varphi_1 \cup \varphi_2} \mid \varphi_1 \cup \varphi_2 \in Cl(\varphi)\}$ , with  
 $F_{\varphi_1 \cup \varphi_2} = \{B \in Q \mid \varphi_1 \cup \varphi_2 \in B \rightarrow \varphi_2 \in B\}$

# Putting Together

---

Given  $\varphi$  over  $AP$ , construct  $A_\varphi = (Q, 2^{AP}, \delta, Q_0, \mathcal{F})$ ,

- ▶  $Q = \{B \mid B \subseteq Cl(\varphi) \text{ is consistent}\}$
- ▶  $Q_0 = \{B \mid \varphi \in B\}$
- ▶  $\delta : Q \times 2^{AP} \rightarrow 2^Q$  is such that
  - ▶ For  $C = B \cap AP$ ,  $\delta(B, C)$  is enabled and is defined as :
  - ▶ If  $\bigcirc\psi \in Cl(\varphi)$ ,  $\bigcirc\psi \in B$  iff  $\psi \in \delta(B, C)$
  - ▶ If  $\varphi_1 \cup \varphi_2 \in Cl(\varphi)$ ,  
 $\varphi_1 \cup \varphi_2 \in B$  iff  $(\varphi_2 \in B \vee (\varphi_1 \in B \wedge \varphi_1 \cup \varphi_2 \in \delta(B, C)))$
- ▶  $\mathcal{F} = \{F_{\varphi_1 \cup \varphi_2} \mid \varphi_1 \cup \varphi_2 \in Cl(\varphi)\}$ , with  
 $F_{\varphi_1 \cup \varphi_2} = \{B \in Q \mid \varphi_1 \cup \varphi_2 \in B \rightarrow \varphi_2 \in B\}$
- ▶ Prove that  $L(\varphi) = L(A_\varphi)$

$$L(\varphi) \subseteq L(A_\varphi)$$

---

Let  $\sigma = A_0A_1A_2\cdots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2\cdots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1}\dots \models \psi\}$ .

Structural induction on  $\varphi$

- ▶  $\varphi = a$ . All starting states contain  $a$ , and can go to all successor states with all combinations of propositions.
- ▶ If  $a \in B_i$ , every run starting at  $B_i$  starts with  $a$ . Hence,  
 $A_iA_{i+1}\dots \models a$

$$L(\varphi) \subseteq L(A_\varphi)$$

---

Let  $\sigma = A_0A_1A_2 \dots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2 \dots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1} \dots \models \psi\}$ .

- ▶  $\varphi = \bigcirc a$ , then all initial states contain  $\bigcirc a$ , and all successor states contain  $a$  by construction.
- ▶ If  $\bigcirc a \in B_i$ , then by construction,  $B_{i+1} \in \delta(B_i, B_i \cap AP)$  iff  $a \in B_{i+1}$ , for every successor  $B_{i+1}$ .  
Then  $A_{i+1} \dots \models a$ , and hence  $A_iA_{i+1} \dots \models \bigcirc a$ .

$$L(\varphi) \subseteq L(A_\varphi)$$

---

Let  $\sigma = A_0A_1A_2 \dots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2 \dots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1} \dots \models \psi\}$ .

- ▶ If  $\varphi_1 \cup \varphi_2 \in B_i$ , then by construction, either  $\varphi_2 \in B_i$  or  $\varphi_1, \varphi_2 \in B_i$ . If  $\varphi_2 \in B_i$  then  $A_iA_{i+1} \dots \models \varphi_2$  by induction hypothesis, and hence,  $A_iA_{i+1} \dots \models \varphi_1 \cup \varphi_2$ .

$$L(\varphi) \subseteq L(A_\varphi)$$

---

Let  $\sigma = A_0A_1A_2 \dots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2 \dots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1} \dots \models \psi\}$ .

- ▶ If  $\varphi_1 \cup \varphi_2 \in B_i$ , then by construction, either  $\varphi_2 \in B_i$  or  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_i$ . If  $\varphi_2 \in B_i$  then  $A_iA_{i+1} \dots \models \varphi_2$  by induction hypothesis, and hence,  $A_iA_{i+1} \dots \models \varphi_1 \cup \varphi_2$ .
- ▶ If  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_i$ , then by construction,  $B_{i+1} \in \delta(B_i, B_i \cap AP)$  iff  $\varphi_2 \in B_{i+1}$  or  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_{i+1}$ . How long can we go like this?

$$L(\varphi) \subseteq L(A_\varphi)$$

Let  $\sigma = A_0A_1A_2\ldots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2\ldots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1}\ldots \models \psi\}$ .

- ▶ If  $\varphi_1 \cup \varphi_2 \in B_i$ , then by construction, either  $\varphi_2 \in B_i$  or  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_i$ . If  $\varphi_2 \in B_i$  then  $A_iA_{i+1}\ldots \models \varphi_2$  by induction hypothesis, and hence,  $A_iA_{i+1}\ldots \models \varphi_1 \cup \varphi_2$ .
- ▶ If  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_i$ , then by construction,  $B_{i+1} \in \delta(B_i, B_i \cap AP)$  iff  $\varphi_2 \in B_{i+1}$  or  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_{i+1}$ . How long can we go like this?
- ▶ If  $\varphi_2 \in B_k$  for  $k > i$ , and  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_{i+1}, \dots, B_{k-1}$  then  $A_iA_{i+1}\ldots \models \varphi_1 \cup \varphi_2$ .
- ▶ When is  $B_iB_{i+1}B_{i+2}\ldots$  an accepting run?
- ▶  $B_j \in F_{\varphi_1 \cup \varphi_2}$  for infinitely many  $j \geq i$ .

$$L(\varphi) \subseteq L(A_\varphi)$$

Let  $\sigma = A_0A_1A_2\ldots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2\ldots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1}\ldots \models \psi\}$ .

- ▶ If  $\varphi_1 \cup \varphi_2 \in B_i$ , then by construction, either  $\varphi_2 \in B_i$  or  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_i$ . If  $\varphi_2 \in B_i$  then  $A_iA_{i+1}\ldots \models \varphi_2$  by induction hypothesis, and hence,  $A_iA_{i+1}\ldots \models \varphi_1 \cup \varphi_2$ .
- ▶ If  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_i$ , then by construction,  $B_{i+1} \in \delta(B_i, B_i \cap AP)$  iff  $\varphi_2 \in B_{i+1}$  or  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_{i+1}$ . How long can we go like this?
- ▶ If  $\varphi_2 \in B_k$  for  $k > i$ , and  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_{i+1}, \dots, B_{k-1}$  then  $A_iA_{i+1}\ldots \models \varphi_1 \cup \varphi_2$ .
- ▶ When is  $B_iB_{i+1}B_{i+2}\ldots$  an accepting run?
- ▶  $B_j \in F_{\varphi_1 \cup \varphi_2}$  for infinitely many  $j \geq i$ .
- ▶  $\varphi_2 \notin B_j$  or  $\varphi_2, \varphi_1 \cup \varphi_2 \in B_j$  for infinitely many  $j \geq i$ .
- ▶ By construction, there is an accepting run where  $\varphi_2 \in B_k$  for some  $k \geq i$ . Hence,  $A_iA_{i+1}\ldots \models \varphi_1 \cup \varphi_2$ .

$$L(\varphi) \subseteq L(A_\varphi)$$

---

Let  $\sigma = A_0A_1A_2 \dots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2 \dots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1} \dots \models \psi\}$ .

- If  $\neg(\varphi_1 \cup \varphi_2) \in B_i$ , then either  $\neg\varphi_1, \neg\varphi_2 \in B_i$  or  $\varphi_1, \neg\varphi_2 \in B_i$ . If  $\neg\varphi_1, \neg\varphi_2 \in B_i$  then  $A_iA_{i+1} \dots \models \neg(\varphi_1 \cup \varphi_2)$ .

$$L(\varphi) \subseteq L(A_\varphi)$$

---

Let  $\sigma = A_0A_1A_2\cdots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2\ldots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1}\dots \models \psi\}$ .

- ▶ If  $\neg(\varphi_1 \cup \varphi_2) \in B_i$ , then either  $\neg\varphi_1, \neg\varphi_2 \in B_i$  or  $\varphi_1, \neg\varphi_2 \in B_i$ . If  $\neg\varphi_1, \neg\varphi_2 \in B_i$  then  $A_iA_{i+1}\dots \models \neg(\varphi_1 \cup \varphi_2)$ .
- ▶ If  $\varphi_1, \neg\varphi_2 \in B_i$ , then by construction,  $B_{i+1} \in \delta(B_i, B_i \cap AP)$  iff  $\varphi_1, \neg\varphi_2 \in B_{i+1}$  or  $\neg\varphi_1, \neg\varphi_2 \in B_{i+1}$ .

$$L(\varphi) \subseteq L(A_\varphi)$$

---

Let  $\sigma = A_0A_1A_2\cdots \in L(\varphi)$ . Show that there is an accepting run  $B_0A_0B_1A_1B_2A_2\ldots$  in  $A_\varphi$  for  $\sigma$ ,  $B_i$  are the states, such that  $B_i = \{\psi \mid A_iA_{i+1}\dots \models \psi\}$ .

- ▶ If  $\neg(\varphi_1 \cup \varphi_2) \in B_i$ , then either  $\neg\varphi_1, \neg\varphi_2 \in B_i$  or  $\varphi_1, \neg\varphi_2 \in B_i$ . If  $\neg\varphi_1, \neg\varphi_2 \in B_i$  then  $A_iA_{i+1}\dots \models \neg(\varphi_1 \cup \varphi_2)$ .
- ▶ If  $\varphi_1, \neg\varphi_2 \in B_i$ , then by construction,  $B_{i+1} \in \delta(B_i, B_i \cap AP)$  iff  $\varphi_1, \neg\varphi_2 \in B_{i+1}$  or  $\neg\varphi_1, \neg\varphi_2 \in B_{i+1}$ .
- ▶ Either case,  $A_iA_{i+1}\dots \models \neg(\varphi_1 \cup \varphi_2)$

$$L(A_\varphi) \subseteq L(\varphi)$$

---

For a sequence  $B_0 B_1 B_2 \dots$  of states satisfying

- ▶  $B_{i+1} \in \delta(B_i, A_i)$ ,
- ▶  $\forall F \in \mathcal{F}, B_j \in F$  for infinitely many  $j$ ,

we have  $\psi \in B_0 \leftrightarrow A_0 A_1 \dots \models \psi$

- ▶ Structural Induction on  $\psi$ . Interesting case :  $\psi = \varphi_1 \mathbf{U} \varphi_2$
- ▶ Assume  $A_0 A_1 \dots \models \varphi_1 \mathbf{U} \varphi_2$ . Then  $\exists j \geq 0, A_j A_{j+1} \dots \models \varphi_2$  and  $A_i A_{i+1} \dots \models \varphi_1, \varphi_1 \mathbf{U} \varphi_2$  for all  $i \leq j$ .
- ▶ By induction hypothesis (applied to  $\varphi_1, \varphi_2$ ), we obtain  $\varphi_2 \in B_j$  and  $\varphi_1 \in B_i$  for all  $i \leq j$
- ▶ By induction on  $j$ ,  $\varphi_1 \mathbf{U} \varphi_2 \in B_j, \dots, B_0$ .

# $L(A_\varphi) \subseteq L(\varphi)$

For a sequence  $B_0B_1B_2\dots$  of states satisfying

- (a)  $B_{i+1} \in \delta(B_i, A_i)$ ,
- (b)  $\forall F \in \mathcal{F}, B_j \in F$  for infinitely many  $j$ ,

we have  $\psi \in B_0 \leftrightarrow A_0A_1\dots \models \psi$

- ▶ Conversely, assume  $\varphi_1 \cup \varphi_2 \in B_0$ . Then  $\varphi_2 \in B_0$  or  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_0$ .
- ▶ If  $\varphi_2 \in B_0$ , by induction hypothesis,  $A_0A_1\dots \models \varphi_2$ , and hence  $A_0A_1\dots \models \varphi_1 \cup \varphi_2$
- ▶ If  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_0$ . Assume  $\varphi_2 \notin B_j$  for all  $j \geq 0$ . Then  $\varphi_1, \varphi_1 \cup \varphi_2 \in B_j$  for all  $j \geq 0$ .
- ▶ Since  $B_0B_1\dots$  satisfies (b),  $B_j \in F_{\varphi_1 \cup \varphi_2}$  for infinitely many  $j \geq 0$ , we obtain a contradiction.
- ▶ Thus,  $\exists$  a smallest  $k$  s.t.  $\varphi_2 \in B_k$ . Then by induction hypothesis,  $A_iA_{i+1}\dots \models \varphi_1$  and  $A_kA_{k+1}\models \varphi_2$  for all  $i < k$
- ▶ Hence,  $A_0A_1\dots \models \varphi_1 \cup \varphi_2$ .

# GNBA Size

---

- ▶ States of  $A_\varphi$  are subsets of  $CI(\varphi)$

# GNBA Size

---

- ▶ States of  $A_\varphi$  are subsets of  $CI(\varphi)$
- ▶ Maximum number of states  $\leq 2^{|\varphi|}$

# GNBA Size

---

- ▶ States of  $A_\varphi$  are subsets of  $CI(\varphi)$
- ▶ Maximum number of states  $\leq 2^{|\varphi|}$
- ▶ Number of sets in  $\mathcal{F} = |\varphi|$

# GNBA Size

---

- ▶ States of  $A_\varphi$  are subsets of  $CI(\varphi)$
- ▶ Maximum number of states  $\leq 2^{|\varphi|}$
- ▶ Number of sets in  $\mathcal{F} = |\varphi|$
- ▶ LTL  $\varphi \rightsquigarrow$  NBA  $A_\varphi$  : Number of states in  $A_\varphi \leq |\varphi|.2^{|\varphi|}$
- ▶ There is no LTL formula  $\varphi$  for the language

$$L = \{A_0 A_1 A_2 \cdots \mid a \in A_{2i}, i \geq 0\}$$

# Complexity of LTL Modelchecking

---

- ▶ Given  $\varphi$ ,  $A_{\neg\varphi}$  has  $\leq 2^{|\varphi|}$  states
  - ▶  $|\varphi|$ =size/length of  $\varphi$ , the number of operators in  $\varphi$
- ▶  $TS \otimes A_{\neg\varphi}$  has  $\leq |TS|.2^{|\varphi|}$  states
- ▶ Persistence checking : Checking  $\square\lozenge\eta$  on  $TS \otimes A_{\neg\varphi}$  takes time linear in  $\eta. |TS \otimes A_{\neg\varphi}|$

# AE Automata and the LTL connection

---

- ▶ For finite words

# Lecture 36.pdf

# CS 228 : Logic in Computer Science

Krishna. S

# Quantifier Rank

---

Let  $\varphi$  be a FO formula over words. Define the **quantifier rank** of  $\varphi$  denoted  $c(\varphi)$

- ▶ If  $\varphi$  is atomic ( $x = y, x < y, S(x, y), Q_a(x)$ ) then  $c(\varphi) = 0$
- ▶  $c(\neg\varphi) = c(\varphi)$

# Quantifier Rank

---

Let  $\varphi$  be a FO formula over words. Define the **quantifier rank** of  $\varphi$  denoted  $c(\varphi)$

- ▶ If  $\varphi$  is atomic ( $x = y, x < y, S(x, y), Q_a(x)$ ) then  $c(\varphi) = 0$
- ▶  $c(\neg\varphi) = c(\varphi)$
- ▶  $c(\varphi \wedge \psi) = \max(c(\varphi), c(\psi))$

# Quantifier Rank

---

Let  $\varphi$  be a FO formula over words. Define the **quantifier rank** of  $\varphi$  denoted  $c(\varphi)$

- ▶ If  $\varphi$  is atomic ( $x = y, x < y, S(x, y), Q_a(x)$ ) then  $c(\varphi) = 0$
- ▶  $c(\neg\varphi) = c(\varphi)$
- ▶  $c(\varphi \wedge \psi) = \max(c(\varphi), c(\psi))$
- ▶  $c(\exists x\varphi) = c(\varphi) + 1$

# Quantifier Rank

---

Let  $\varphi$  be a FO formula over words. Define the **quantifier rank** of  $\varphi$  denoted  $c(\varphi)$

- ▶ If  $\varphi$  is atomic ( $x = y, x < y, S(x, y), Q_a(x)$ ) then  $c(\varphi) = 0$
- ▶  $c(\neg\varphi) = c(\varphi)$
- ▶  $c(\varphi \wedge \psi) = \max(c(\varphi), c(\psi))$
- ▶  $c(\exists x\varphi) = c(\varphi) + 1$
- ▶ Quantifier free formulae written in DNF :  $C_1 \vee C_2 \vee \dots \vee C_n$

# Quantifier Rank

---

Let  $\varphi$  be a FO formula over words. Define the **quantifier rank** of  $\varphi$  denoted  $c(\varphi)$

- ▶ If  $\varphi$  is atomic ( $x = y, x < y, S(x, y), Q_a(x)$ ) then  $c(\varphi) = 0$
- ▶  $c(\neg\varphi) = c(\varphi)$
- ▶  $c(\varphi \wedge \psi) = \max(c(\varphi), c(\psi))$
- ▶  $c(\exists x\varphi) = c(\varphi) + 1$
- ▶ Quantifier free formulae written in DNF :  $C_1 \vee C_2 \vee \dots \vee C_n$
- ▶ Formulae of quantifier rank  $k + 1$  written as a disjunction of the conjunction of formulae, each formula of the form  $\exists x\varphi, \neg\exists x\varphi$  or  $\varphi$ , with  $c(\varphi) \leq k$ . Eliminate repeated disjuncts/conjunts.  
(Think Prenex normal form)

# Number of FO formulae of rank $\leq c$

---

Let  $\mathcal{V}$  be a finite set of first order variables, and let  $c \geq 0$ . There are finitely many FO formulae in normal form (therefore, upto equivalence) with rank  $\leq c$  over  $\mathcal{V}$ .

Proof ideas in the following slides.

# Number of FO formulae of rank $\leq c$

---

Let  $\mathcal{V}$  be a finite set of first order variables. Fix a finite signature  $\tau$ . Let there be  $m$  atomic formulae over  $\tau$  having variables from  $\mathcal{V}$ .

- ▶ If  $\mathcal{V}$  has 2 variables  $x, y$ , and  $\tau$  has  $Q_a, S, <$ .
- ▶ Atomic formulae :  $\{Q_a(x), Q_a(y), S(x, y), x < y\}$
- ▶ Let  
 $G = \{Q_a(x), \neg Q_a(x), Q_a(y), \neg Q_a(y), S(x, y), \neg S(x, y), x < y, \neg(x < y)\}$   
be the closure of all atomic formulae containing all formulae and their negations.
- ▶ Each subset of  $G$  is a possible conjunct  $C_i$ .

# Number of FO formulae of rank $\leq c$

---

Let  $\mathcal{V}$  be a finite set of first order variables. Fix a finite signature  $\tau$ . Let there be  $m$  atomic formulae over  $\tau$  having variables from  $\mathcal{V}$ .

- ▶ If  $\mathcal{V}$  has 2 variables  $x, y$ , and  $\tau$  has  $Q_a, S, <$ .
- ▶ Atomic formulae :  $\{Q_a(x), Q_a(y), S(x, y), x < y\}$
- ▶ Let  
 $G = \{Q_a(x), \neg Q_a(x), Q_a(y), \neg Q_a(y), S(x, y), \neg S(x, y), x < y, \neg(x < y)\}$   
be the closure of all atomic formulae containing all formulae and their negations.
- ▶ Each subset of  $G$  is a possible conjunct  $C_i$ .
- ▶ All possible disjuncts using each  $C_i$  : formulae in DNF of rank 0

# Number of FO formulae of rank $\leq c$

---

Let  $\mathcal{V}$  be a finite set of first order variables. Fix a finite signature  $\tau$ . Let there be  $m$  atomic formulae over  $\tau$  having variables from  $\mathcal{V}$ .

- ▶  $2m$  atomic/negated atomic formulae
- ▶ Number of conjunctions  $C_i$  possible  $\leq 2^{2^m}$
- ▶ Number of formulae in DNF  $\leq 2^{2^m}$  ( $c = 0$ )

# Rank 1

---

Let there be  $p$  formulae  $\varphi$  of rank 0.

- ▶  $2p$  formulae of the form  $\exists x\varphi, \neg\exists x\varphi$
- ▶  $2^{2p}$  conjunctions of rank 1
- ▶ Conjoining any one of the  $p$  formulae of rank 0 gives all conjuncts of rank 1 :  $p2^{2p}$  more
- ▶ Possible conjuncts of rank 1 is  $q = (p + 1)2^{2p}$
- ▶ Possible disjuncts of these :  $2^q$

# Some Notation

Given a word  $w = a_1 \dots a_n$ , and a finite set of variables  $\mathcal{V}$ , define a  $\mathcal{V}$ -enriched-word with respect to  $w$  as

- ▶  $(a_1, U_1)(a_2, U_2) \dots (a_n, U_n)$  where
  - ▶  $\bigcup_i U_i = \mathcal{V}$
  - ▶  $U_i \cap U_j = \emptyset$
- 
- ▶ A  $\mathcal{V}$ -enriched-word is over the alphabet  $\Sigma \times 2^{\mathcal{V}}$
  - ▶  $(a, \{x\})(b, \{y, z\})(c, \emptyset)(d, \{u, v\})$  is a  $\{x, y, z, u, v\}$ -enriched word with respect to the word  $abcd$ .
  - ▶ We will refer to  $\mathcal{V}$ -enriched-word structures as  $\mathcal{V}$ -structures from here on

# Notational Semantics

---

Given a  $\mathcal{V}$ -structure  $w = (a_1, S_1) \dots (a_n, S_n)$ ,

# Notational Semantics

---

Given a  $\mathcal{V}$ -structure  $w = (a_1, S_1) \dots (a_n, S_n)$ ,

- ▶  $w \models Q_a(x)$  iff there exists  $j$  such that  $a_j = a$  and  $x \in S_j$ 
  - ▶  $(a, \{y\})(b, \{u, v\})(a, \{x\}) \models Q_a(x)$

# Notational Semantics

---

Given a  $\mathcal{V}$ -structure  $w = (a_1, S_1) \dots (a_n, S_n)$ ,

- ▶  $w \models Q_a(x)$  iff there exists  $j$  such that  $a_j = a$  and  $x \in S_j$ 
  - ▶  $(a, \{y\})(b, \{u, v\})(a, \{x\}) \models Q_a(x)$
- ▶  $w \models (x = y)$  iff there exists  $j$  such that  $x, y \in S_j$ 
  - ▶  $(a, \{x\})(b, \{y, z\})(c, \emptyset) \not\models (x = y)$

# Notational Semantics

---

Given a  $\mathcal{V}$ -structure  $w = (a_1, S_1) \dots (a_n, S_n)$ ,

- ▶  $w \models Q_a(x)$  iff there exists  $j$  such that  $a_j = a$  and  $x \in S_j$ 
  - ▶  $(a, \{y\})(b, \{u, v\})(a, \{x\}) \models Q_a(x)$
- ▶  $w \models (x = y)$  iff there exists  $j$  such that  $x, y \in S_j$ 
  - ▶  $(a, \{x\})(b, \{y, z\})(c, \emptyset) \not\models (x = y)$
- ▶  $w \models x < y$  iff there exists  $i < j$  such that  $x \in S_i, y \in S_j$ 
  - ▶  $(a, \{x\})(b, \{y, z\})(c, \emptyset) \models x < y$

# Notational Semantics

---

Given a  $\mathcal{V}$ -structure  $w = (a_1, S_1) \dots (a_n, S_n)$ ,

- ▶  $w \models Q_a(x)$  iff there exists  $j$  such that  $a_j = a$  and  $x \in S_j$ 
  - ▶  $(a, \{y\})(b, \{u, v\})(a, \{x\}) \models Q_a(x)$
- ▶  $w \models (x = y)$  iff there exists  $j$  such that  $x, y \in S_j$ 
  - ▶  $(a, \{x\})(b, \{y, z\})(c, \emptyset) \not\models (x = y)$
- ▶  $w \models x < y$  iff there exists  $i < j$  such that  $x \in S_i, y \in S_j$ 
  - ▶  $(a, \{x\})(b, \{y, z\})(c, \emptyset) \models x < y$
- ▶  $w \models \exists x Q_a(x)$  iff there exists  $i$  such that  
 $(a_1, S_1) \dots (a_i, S_i \cup \{x\}) \dots (a_n, S_n) \models Q_a(x)$ 
  - ▶  $(b, \{y, z\})(a, \{u\})(c, \emptyset) \models \exists x Q_a(x)$  since  
 $(b, \{y, z\})(a, \{x, u\})(c, \emptyset) \models Q_a(x)$

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \neg \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \neg \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \not\models \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \neg \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \not\models \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
  - ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \neg \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \not\models \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
  - ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \{x\})(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \neg \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \not\models \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
  - ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \{x\})(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \emptyset)(b, \{x\}) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \neg \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \not\models \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
  - ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \{x\})(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \emptyset)(b, \{x\}) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$
- ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \models \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \neg \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \not\models \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
  - ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \{x\})(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \emptyset)(b, \{x\}) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$
- ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \models \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \{x\})(a, \emptyset)(b, \{y\}) \models (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$

# Notational Semantics

---

- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \forall x \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \models \neg \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
- ▶  $(a, \emptyset)(a, \emptyset)(b, \emptyset) \not\models \exists x \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  iff
  - ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \{x\})(b, \emptyset) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$  and
  - ▶  $(a, \emptyset)(a, \emptyset)(b, \{x\}) \not\models \neg [\exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])]$
- ▶  $(a, \{x\})(a, \emptyset)(b, \emptyset) \models \exists y (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  iff
- ▶  $(a, \{x\})(a, \emptyset)(b, \{y\}) \models (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$

Similarly,  $(a, \emptyset)(a, \{x\})(b, \{y\}) \models (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$  and  
 $(a, \emptyset)(a, \emptyset)(b, \{x, y\}) \models (Q_a(x) \rightarrow [(x < y) \wedge Q_b(y)])$

# Logical Equivalence

---

- ▶ Let  $w_1, w_2$  be two  $\mathcal{V}$ -structures and let  $r \geq 0$ .

# Logical Equivalence

---

- ▶ Let  $w_1, w_2$  be two  $\mathcal{V}$ -structures and let  $r \geq 0$ .
- ▶ Write  $w_1 \sim_r w_2$  iff  $w_1, w_2$  satisfy the same set of FO formulae of rank  $\leq r$ .

# Logical Equivalence

---

- ▶ Let  $w_1, w_2$  be two  $\mathcal{V}$ -structures and let  $r \geq 0$ .
- ▶ Write  $w_1 \sim_r w_2$  iff  $w_1, w_2$  satisfy the same set of FO formulae of rank  $\leq r$ .
- ▶  $(a, \emptyset)(b, \emptyset) \sim_0 (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶  $(a, \emptyset)(b, \emptyset) \sim_2 (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶  $(a, \emptyset)(b, \emptyset) \sim_1 (a, \emptyset)(b, \emptyset)(a, \emptyset)?$

# Logical Equivalence

---

- ▶ Let  $w_1, w_2$  be two  $\mathcal{V}$ -structures and let  $r \geq 0$ .
- ▶ Write  $w_1 \sim_r w_2$  iff  $w_1, w_2$  satisfy the same set of FO formulae of rank  $\leq r$ .
- ▶  $(a, \emptyset)(b, \emptyset) \sim_0 (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶  $(a, \emptyset)(b, \emptyset) \sim_2 (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶  $(a, \emptyset)(b, \emptyset) \sim_1 (a, \emptyset)(b, \emptyset)(a, \emptyset)?$
- ▶  $\sim_r$  is an equivalence relation

# Logical Equivalence

---

- ▶ Let  $w_1, w_2$  be two  $\mathcal{V}$ -structures and let  $r \geq 0$ .
- ▶ Write  $w_1 \sim_r w_2$  iff  $w_1, w_2$  satisfy the same set of FO formulae of rank  $\leq r$ .
- ▶  $(a, \emptyset)(b, \emptyset) \sim_0 (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶  $(a, \emptyset)(b, \emptyset) \sim_2 (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶  $(a, \emptyset)(b, \emptyset) \sim_1 (a, \emptyset)(b, \emptyset)(a, \emptyset)?$
- ▶  $\sim_r$  is an equivalence relation
- ▶ **Finitely** many equivalence classes : each class consists of words that behave the same way on formulae of rank  $\leq r$

## Non-Expressibility in FO : The Game Begins

# Come, Lets Play

---

- ▶ Given two  $\mathcal{V}$ -structures  $w_1, w_2$ , lets play a game on the pair of words  $w_1, w_2$

# Come, Lets Play

---

- ▶ Given two  $\mathcal{V}$ -structures  $w_1, w_2$ , lets play a game on the pair of words  $w_1, w_2$
- ▶ There are 2 players : **Spoiler** and **Duplicator**

# Come, Lets Play

---

- ▶ Given two  $\mathcal{V}$ -structures  $w_1, w_2$ , lets play a game on the pair of words  $w_1, w_2$
- ▶ There are 2 players : **Spoiler** and **Duplicator**
- ▶ Play for  $r$ -rounds,  $r \geq 0$

# Come, Lets Play

---

- ▶ Given two  $\mathcal{V}$ -structures  $w_1, w_2$ , lets play a game on the pair of words  $w_1, w_2$
- ▶ There are 2 players : **Spoiler** and **Duplicator**
- ▶ Play for  $r$ -rounds,  $r \geq 0$
- ▶ Spoiler wants to show that  $w_1, w_2$  are different ( $w_1 \not\sim_r w_2$ )

# Come, Lets Play

---

- ▶ Given two  $\mathcal{V}$ -structures  $w_1, w_2$ , lets play a game on the pair of words  $w_1, w_2$
- ▶ There are 2 players : **Spoiler** and **Duplicator**
- ▶ Play for  $r$ -rounds,  $r \geq 0$
- ▶ Spoiler wants to show that  $w_1, w_2$  are different ( $w_1 \not\sim_r w_2$ )
- ▶ Duplicator wants to show that they are same ( $w_1 \sim_r w_2$ )

# Come, Lets Play

---

- ▶ Given two  $\mathcal{V}$ -structures  $w_1, w_2$ , lets play a game on the pair of words  $w_1, w_2$
- ▶ There are 2 players : **Spoiler** and **Duplicator**
- ▶ Play for  $r$ -rounds,  $r \geq 0$
- ▶ Spoiler wants to show that  $w_1, w_2$  are different ( $w_1 \not\sim_r w_2$ )
- ▶ Duplicator wants to show that they are same ( $w_1 \sim_r w_2$ )
- ▶ Each player has  $r$  pebbles  $z_1, \dots, z_r$

# Moves of the Game

---

- ▶ At the start of each round, spoiler chooses a structure.

# Moves of the Game

---

- ▶ At the start of each round, spoiler chooses a structure.
- ▶ Duplicator gets the other structure

# Moves of the Game

---

- ▶ At the start of each round, spoiler chooses a structure.
- ▶ Duplicator gets the other structure
- ▶ Spoiler places his pebble say  $z_i$  on one of the positions of his chosen word

# Moves of the Game

---

- ▶ At the start of each round, spoiler chooses a structure.
- ▶ Duplicator gets the other structure
- ▶ Spoiler places his pebble say  $z_i$  on one of the positions of his chosen word
- ▶ Duplicator must keep the pebble  $z_i$  on one of the positions of her word

# Moves of the Game

---

- ▶ At the start of each round, spoiler chooses a structure.
- ▶ Duplicator gets the other structure
- ▶ Spoiler places his pebble say  $z_i$  on one of the positions of his chosen word
- ▶ Duplicator must keep the pebble  $z_i$  on one of the positions of her word
- ▶ A pebble once placed, cannot be removed

# Moves of the Game

---

- ▶ At the start of each round, spoiler chooses a structure.
- ▶ Duplicator gets the other structure
- ▶ Spoiler places his pebble say  $z_i$  on one of the positions of his chosen word
- ▶ Duplicator must keep the pebble  $z_i$  on one of the positions of her word
- ▶ A pebble once placed, cannot be removed
- ▶ The game ends after  $r$  rounds, when both players have used all their pebbles

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$
- ▶ Round 1:
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \emptyset)$

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$
- ▶ Round 1:
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \emptyset)$
  - ▶ Duplicator :  $(a, \{z_1\})(b, \emptyset)$

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$
- ▶ Round 1:
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \emptyset)$
  - ▶ Duplicator :  $(a, \{z_1\})(b, \emptyset)$
  - ▶ After round 1, we have two  $\{z_1\}$  structures ( $w'_1, w'_2$ )

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$
- ▶ Round 1:
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \emptyset)$
  - ▶ Duplicator :  $(a, \{z_1\})(b, \emptyset)$
  - ▶ After round 1, we have two  $\{z_1\}$  structures ( $w'_1, w'_2$ )
- ▶ Round 2:

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$
- ▶ Round 1:
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \emptyset)$
  - ▶ Duplicator :  $(a, \{z_1\})(b, \emptyset)$
  - ▶ After round 1, we have two  $\{z_1\}$  structures ( $w'_1, w'_2$ )
- ▶ Round 2:
  - ▶ Spoiler continues on the structure  $w'_2$

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$
- ▶ Round 1:
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \emptyset)$
  - ▶ Duplicator :  $(a, \{z_1\})(b, \emptyset)$
  - ▶ After round 1, we have two  $\{z_1\}$  structures ( $w'_1, w'_2$ )
- ▶ Round 2:
  - ▶ Spoiler continues on the structure  $w'_2$
  - ▶ Duplicator gets  $w'_1$  to play

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$
- ▶ Round 1:
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \emptyset)$
  - ▶ Duplicator :  $(a, \{z_1\})(b, \emptyset)$
  - ▶ After round 1, we have two  $\{z_1\}$  structures ( $w'_1, w'_2$ )
- ▶ Round 2:
  - ▶ Spoiler continues on the structure  $w'_2$
  - ▶ Duplicator gets  $w'_1$  to play
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\})$

# A Play

---

- ▶  $w_1 = (a, \emptyset)(b, \emptyset)$  and  $w_2 = (a, \emptyset)(b, \emptyset)(a, \emptyset)$
- ▶ 2 rounds, so 2 pebbles :  $z_1, z_2$
- ▶ Spoiler picks  $w_2$ , duplicator picks  $w_1$
- ▶ Round 1:
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \emptyset)$
  - ▶ Duplicator :  $(a, \{z_1\})(b, \emptyset)$
  - ▶ After round 1, we have two  $\{z_1\}$  structures ( $w'_1, w'_2$ )
- ▶ Round 2:
  - ▶ Spoiler continues on the structure  $w'_2$
  - ▶ Duplicator gets  $w'_1$  to play
  - ▶ Spoiler :  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\})$
  - ▶ Duplicator :  $(a, \{z_1, z_2\})(b, \emptyset)$  or  $(a, \{z_1\})(b, \{z_2\})$

# Winner

---

- ▶ Start with two  $\emptyset$  structures ( $w_1, w_2$ )

# Winner

---

- ▶ Start with two  $\emptyset$  structures  $(w_1, w_2)$
- ▶  $r$ -round game, pebble set  $\mathcal{V} = \{z_1, \dots, z_r\}$

# Winner

---

- ▶ Start with two  $\emptyset$  structures ( $w_1, w_2$ )
- ▶  $r$ -round game, pebble set  $\mathcal{V} = \{z_1, \dots, z_r\}$
- ▶ Each round changes the structures

# Winner

---

- ▶ Start with two  $\emptyset$  structures  $(w_1, w_2)$
- ▶  $r$ -round game, pebble set  $\mathcal{V} = \{z_1, \dots, z_r\}$
- ▶ Each round changes the structures
- ▶ At the end of  $r$ -rounds, we have two  $\mathcal{V}$ -structures  $(w'_1, w'_2)$

# Winner

---

- ▶ Start with two  $\emptyset$  structures  $(w_1, w_2)$
- ▶  $r$ -round game, pebble set  $\mathcal{V} = \{z_1, \dots, z_r\}$
- ▶ Each round changes the structures
- ▶ At the end of  $r$ -rounds, we have two  $\mathcal{V}$ -structures  $(w'_1, w'_2)$
- ▶ Duplicator wins iff for every atomic formula  $\alpha$ ,  
 $w'_1 \models \alpha$  iff  $w'_2 \models \alpha$

# Winner

---

- ▶ Start with two  $\emptyset$  structures  $(w_1, w_2)$
- ▶  $r$ -round game, pebble set  $\mathcal{V} = \{z_1, \dots, z_r\}$
- ▶ Each round changes the structures
- ▶ At the end of  $r$ -rounds, we have two  $\mathcal{V}$ -structures  $(w'_1, w'_2)$
- ▶ Duplicator wins iff for every atomic formula  $\alpha$ ,  
 $w'_1 \models \alpha$  iff  $w'_2 \models \alpha$
- ▶ That is,  $w'_1 \sim_0 w'_2$

# Winner

---

- ▶ Start with two  $\emptyset$  structures  $(w_1, w_2)$
- ▶  $r$ -round game, pebble set  $\mathcal{V} = \{z_1, \dots, z_r\}$
- ▶ Each round changes the structures
- ▶ At the end of  $r$ -rounds, we have two  $\mathcal{V}$ -structures  $(w'_1, w'_2)$
- ▶ Duplicator wins iff for every atomic formula  $\alpha$ ,  
 $w'_1 \models \alpha$  iff  $w'_2 \models \alpha$
- ▶ That is,  $w'_1 \sim_0 w'_2$
- ▶ Spoiler wins otherwise.

# Winner

---

Given two word structures  $(w_1, w_2)$ , duplicator wins on  $(w_1, w_2)$  if for every atomic formula  $\alpha$ ,  $w_1 \models \alpha$  iff  $w_2 \models \alpha$

# Play continues

---

- ▶ Who won in the earlier play?
- ▶ We had
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\})$  and  $(a, \{z_1, z_2\})(b, \emptyset)$
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\}) \models (z_1 < z_2)$
  - ▶  $(a, \{z_1, z_2\})(b, \emptyset) \not\models (z_1 < z_2)$  or

# Play continues

---

- ▶ Who won in the earlier play?
- ▶ We had
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\})$  and  $(a, \{z_1, z_2\})(b, \emptyset)$
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\}) \models (z_1 < z_2)$
  - ▶  $(a, \{z_1, z_2\})(b, \emptyset) \not\models (z_1 < z_2)$  or
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\})$  and  $(a, \{z_1\})(b, \{z_2\})$
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\}) \models Q_a(z_2)$
  - ▶  $(a, \{z_1\})(b, \{z_2\}) \not\models Q_a(z_2)$
- ▶ Spoiler wins in two rounds

# Play continues

---

- ▶ Who won in the earlier play?
- ▶ We had
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\})$  and  $(a, \{z_1, z_2\})(b, \emptyset)$
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\}) \models (z_1 < z_2)$
  - ▶  $(a, \{z_1, z_2\})(b, \emptyset) \not\models (z_1 < z_2)$  or
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\})$  and  $(a, \{z_1\})(b, \{z_2\})$
  - ▶  $(a, \{z_1\})(b, \emptyset)(a, \{z_2\}) \models Q_a(z_2)$
  - ▶  $(a, \{z_1\})(b, \{z_2\}) \not\models Q_a(z_2)$
- ▶ Spoiler wins in two rounds
- ▶ If the game was played only for one round, who will win?

# Unique Winner

Given structures  $w_1, w_2$ , and a number of rounds  $r$ , exactly one of the players win.

- ▶ Think of a tree of all possible sequences of plays.
- ▶ The root is  $(w_1, w_2)$ , and nodes are all possible pairs  $(w'_1, w'_2)$  of obtainable structures.
- ▶ Mark a leaf node as S or D depending on whether Spoiler won or Duplicator won.
- ▶ An interior node corresponding to a move of Duplicator is labeled D if there any children labeled D; likewise for Spoiler

# Logical Equivalence and Winning

---

Let  $w_1, w_2$  be  $\mathcal{V}$ -structures and let  $r \geq 0$ . Then  $w_1 \sim_r w_2$  iff Duplicator has a winning strategy in the  $r$ -round game on  $(w_1, w_2)$ .

# Logical Equivalence and Winning

---

Assume  $w_1 \sim_r w_2$ , and induct on  $r$

- ▶ Base :  $r = 0$  and  $w_1 \sim_0 w_2$ . Duplicator wins, since by assumption,  $w_1, w_2$  agree on all atomic formulae.

# Logical Equivalence and Winning

---

Assume  $w_1 \sim_r w_2$ , and induct on  $r$

- ▶ Base :  $r = 0$  and  $w_1 \sim_0 w_2$ . Duplicator wins, since by assumption,  $w_1, w_2$  agree on all atomic formulae.
- ▶ Assume for  $r - 1$  :  $w_1 \sim_{r-1} w_2 \Rightarrow$  Duplicator has a winning strategy in a  $r - 1$  round game

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$
  - ▶ The resultant structure is  $w'_1$

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$
  - ▶ The resultant structure is  $w'_1$
  - ▶ In response, duplicator places her pebble somewhere on  $w_2$

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$
  - ▶ The resultant structure is  $w'_1$
  - ▶ In response, duplicator places her pebble somewhere on  $w_2$
  - ▶ The resultant structure is  $w'_2$

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$
  - ▶ The resultant structure is  $w'_1$
  - ▶ In response, duplicator places her pebble somewhere on  $w_2$
  - ▶ The resultant structure is  $w'_2$
  - ▶ By assumption, spoiler wins the  $r - 1$  round game on  $(w'_1, w'_2)$

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$
  - ▶ The resultant structure is  $w'_1$
  - ▶ In response, duplicator places her pebble somewhere on  $w_2$
  - ▶ The resultant structure is  $w'_2$
  - ▶ By assumption, spoiler wins the  $r - 1$  round game on  $(w'_1, w'_2)$
  - ▶ By inductive hypothesis,  $w'_1 \sim_{r-1} w'_2$

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$
  - ▶ The resultant structure is  $w'_1$
  - ▶ In response, duplicator places her pebble somewhere on  $w_2$
  - ▶ The resultant structure is  $w'_2$
  - ▶ By assumption, spoiler wins the  $r - 1$  round game on  $(w'_1, w'_2)$
  - ▶ By inductive hypothesis,  $w'_1 \sim_{r-1} w'_2$
  - ▶ Let  $\psi$  be the conjunction of all formulae of rank  $r - 1$  in normal form that are satisfied by  $w'_1$

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$
  - ▶ The resultant structure is  $w'_1$
  - ▶ In response, duplicator places her pebble somewhere on  $w_2$
  - ▶ The resultant structure is  $w'_2$
  - ▶ By assumption, spoiler wins the  $r - 1$  round game on  $(w'_1, w'_2)$
  - ▶ By inductive hypothesis,  $w'_1 \sim_{r-1} w'_2$
  - ▶ Let  $\psi$  be the conjunction of all formulae of rank  $r - 1$  in normal form that are satisfied by  $w'_1$
  - ▶ Then  $w'_1 \models \psi, w'_2 \not\models \psi$

# Logical Equivalence and Winning

---

- ▶ Now, let  $w_1 \sim_r w_2$ , and assume spoiler wins the  $r$ -round game on  $(w_1, w_2)$ .
  - ▶ Assume spoiler starts on  $w_1$ , places a pebble  $z_1$  somewhere on  $w_1$
  - ▶ The resultant structure is  $w'_1$
  - ▶ In response, duplicator places her pebble somewhere on  $w_2$
  - ▶ The resultant structure is  $w'_2$
  - ▶ By assumption, spoiler wins the  $r - 1$  round game on  $(w'_1, w'_2)$
  - ▶ By inductive hypothesis,  $w'_1 \sim_{r-1} w'_2$
  - ▶ Let  $\psi$  be the conjunction of all formulae of rank  $r - 1$  in normal form that are satisfied by  $w'_1$
  - ▶ Then  $w'_1 \models \psi, w'_2 \not\models \psi$
  - ▶ We thus have

$$w_1 \models \exists z_1 \psi, w_2 \not\models \exists z_1 \psi$$

contradicting  $w_1 \sim_r w_2$

# Logical Equivalence and Winning : Converse

---

Assume Duplicator wins  $r$ -round games on  $(w_1, w_2)$  and induct on  $r$

- ▶ Base :  $r = 0$  and Duplicator wins. Then  $w_1, w_2$  agree on all atomic formulae, and hence  $w_1 \sim_0 w_2$

# Logical Equivalence and Winning : Converse

---

Assume Duplicator wins  $r$ -round games on  $(w_1, w_2)$  and induct on  $r$

- ▶ Base :  $r = 0$  and Duplicator wins. Then  $w_1, w_2$  agree on all atomic formulae, and hence  $w_1 \sim_0 w_2$
- ▶ Assume for  $r - 1$  : Duplicator has a winning strategy in a  $r - 1$  round game  $\Rightarrow w_1 \sim_{r-1} w_2$

# Logical Equivalence and Winning : Converse

---

- ▶ Now, let duplicator win in the  $r$  round game, but  $w_1 \not\sim_r w_2$ .
  - ▶  $w_1 \not\sim_r w_2 \Rightarrow$  there is some formula  $\psi$ ,  $c(\psi) = r$  such that  
 $w_1 \models \psi, w_2 \not\models \psi$

# Logical Equivalence and Winning : Converse

---

- ▶ Now, let duplicator win in the  $r$  round game, but  $w_1 \not\sim_r w_2$ .
  - ▶  $w_1 \not\sim_r w_2 \Rightarrow$  there is some formula  $\psi$ ,  $c(\psi) = r$  such that  
 $w_1 \models \psi, w_2 \not\models \psi$
  - ▶ Assume  $\psi = \exists z_1 \varphi$ . Then  $c(\varphi) = r - 1$

# Logical Equivalence and Winning : Converse

---

- ▶ Now, let duplicator win in the  $r$  round game, but  $w_1 \not\sim_r w_2$ .
  - ▶  $w_1 \not\sim_r w_2 \Rightarrow$  there is some formula  $\psi$ ,  $c(\psi) = r$  such that  
 $w_1 \models \psi$ ,  $w_2 \not\models \psi$
  - ▶ Assume  $\psi = \exists z_1 \varphi$ . Then  $c(\varphi) = r - 1$
  - ▶ Since  $w_1 \models \exists z_1 \varphi$ , spoiler can keep pebble  $z_1$  somewhere in  $w_1$  obtaining  $w'_1$  satisfying  $\varphi$

# Logical Equivalence and Winning : Converse

---

- ▶ Now, let duplicator win in the  $r$  round game, but  $w_1 \not\sim_r w_2$ .
  - ▶  $w_1 \not\sim_r w_2 \Rightarrow$  there is some formula  $\psi$ ,  $c(\psi) = r$  such that  $w_1 \models \psi, w_2 \not\models \psi$
  - ▶ Assume  $\psi = \exists z_1 \varphi$ . Then  $c(\varphi) = r - 1$
  - ▶ Since  $w_1 \models \exists z_1 \varphi$ , spoiler can keep pebble  $z_1$  somewhere in  $w_1$  obtaining  $w'_1$  satisfying  $\varphi$
  - ▶ In reply, duplicator keeps pebble  $z_1$  on  $w_2$  obtaining  $w'_2$

# Logical Equivalence and Winning : Converse

---

- ▶ Now, let duplicator win in the  $r$  round game, but  $w_1 \not\sim_r w_2$ .
  - ▶  $w_1 \not\sim_r w_2 \Rightarrow$  there is some formula  $\psi$ ,  $c(\psi) = r$  such that  
 $w_1 \models \psi, w_2 \not\models \psi$
  - ▶ Assume  $\psi = \exists z_1 \varphi$ . Then  $c(\varphi) = r - 1$
  - ▶ Since  $w_1 \models \exists z_1 \varphi$ , spoiler can keep pebble  $z_1$  somewhere in  $w_1$  obtaining  $w'_1$  satisfying  $\varphi$
  - ▶ In reply, duplicator keeps pebble  $z_1$  on  $w_2$  obtaining  $w'_2$
  - ▶ By assumption,  $w'_2 \not\models \varphi$

# Logical Equivalence and Winning : Converse

---

- ▶ Now, let duplicator win in the  $r$  round game, but  $w_1 \not\sim_r w_2$ .
  - ▶  $w_1 \not\sim_r w_2 \Rightarrow$  there is some formula  $\psi$ ,  $c(\psi) = r$  such that  $w_1 \models \psi, w_2 \not\models \psi$
  - ▶ Assume  $\psi = \exists z_1 \varphi$ . Then  $c(\varphi) = r - 1$
  - ▶ Since  $w_1 \models \exists z_1 \varphi$ , spoiler can keep pebble  $z_1$  somewhere in  $w_1$  obtaining  $w'_1$  satisfying  $\varphi$
  - ▶ In reply, duplicator keeps pebble  $z_1$  on  $w_2$  obtaining  $w'_2$
  - ▶ By assumption,  $w'_2 \not\models \varphi$
  - ▶ Also, by assumption, duplicator wins the  $r - 1$  round game on  $(w'_1, w'_2)$  : this by inductive hypothesis says that  $w'_1 \sim_{r-1} w'_2$

# Logical Equivalence and Winning : Converse

---

- ▶ Now, let duplicator win in the  $r$  round game, but  $w_1 \not\sim_r w_2$ .
  - ▶  $w_1 \not\sim_r w_2 \Rightarrow$  there is some formula  $\psi$ ,  $c(\psi) = r$  such that  $w_1 \models \psi, w_2 \not\models \psi$
  - ▶ Assume  $\psi = \exists z_1 \varphi$ . Then  $c(\varphi) = r - 1$
  - ▶ Since  $w_1 \models \exists z_1 \varphi$ , spoiler can keep pebble  $z_1$  somewhere in  $w_1$  obtaining  $w'_1$  satisfying  $\varphi$
  - ▶ In reply, duplicator keeps pebble  $z_1$  on  $w_2$  obtaining  $w'_2$
  - ▶ By assumption,  $w'_2 \not\models \varphi$
  - ▶ Also, by assumption, duplicator wins the  $r - 1$  round game on  $(w'_1, w'_2)$  : this by inductive hypothesis says that  $w'_1 \sim_{r-1} w'_2$
  - ▶ That is, either both  $w'_1, w'_2$  satisfy  $\varphi$ , or both dont, a contradiction.

# FO-definable languages

---

Assume  $L$  is FO-definable, and  $L = L(\varphi)$  with rank of  $\varphi$  being  $k$ .

- ▶ Let  $L = \{v_1, v_2, v_3, \dots\}$  and  $\bar{L} = \{w_1, w_2, w_3, \dots\}$

# FO-definable languages

---

Assume  $L$  is FO-definable, and  $L = L(\varphi)$  with rank of  $\varphi$  being  $k$ .

- ▶ Let  $L = \{v_1, v_2, v_3, \dots\}$  and  $\bar{L} = \{w_1, w_2, w_3, \dots\}$
- ▶ Play a  $k$  round game on  $v_i \in L$  and  $w_j \notin L$ . Let  $\psi_{v_i, w_j}$  be the formula of rank  $k$  that distinguishes the two words.

# FO-definable languages

---

Assume  $L$  is FO-definable, and  $L = L(\varphi)$  with rank of  $\varphi$  being  $k$ .

- ▶ Let  $L = \{v_1, v_2, v_3, \dots\}$  and  $\bar{L} = \{w_1, w_2, w_3, \dots\}$
- ▶ Play a  $k$  round game on  $v_i \in L$  and  $w_j \notin L$ . Let  $\psi_{v_i, w_j}$  be the formula of rank  $k$  that distinguishes the two words.
- ▶ Consider the formula

$$[\psi_{v_1, w_1} \wedge \psi_{v_1, w_2} \wedge \cdots \wedge \psi_{v_1, w_n} \wedge \dots]$$

$\vee$

$$[\psi_{v_2, w_1} \wedge \psi_{v_2, w_2} \wedge \cdots \wedge \psi_{v_2, w_n} \wedge \dots]$$

$\vee$

$\vdots$

# FO-definable languages

---

$$\psi_L = \bigvee_{v \in L} \bigwedge_{w \notin L} \psi_{vw}$$

# FO-definable languages

---

$$\psi_L = \bigvee_{v \in L} \bigwedge_{w \notin L} \psi_{vw}$$

- ▶ Each  $\psi_{vw}$  has rank atmost  $k$

# FO-definable languages

---

$$\psi_L = \bigvee_{v \in L} \bigwedge_{w \notin L} \psi_{vw}$$

- ▶ Each  $\psi_{vw}$  has rank atmost  $k$
- ▶ Upto equivalence, there are finitely many formulae of rank  $k$

# FO-definable languages

---

$$\psi_L = \bigvee_{v \in L} \bigwedge_{w \notin L} \psi_{vw}$$

- ▶ Each  $\psi_{vw}$  has rank atmost  $k$
- ▶ Upto equivalence, there are finitely many formulae of rank  $k$
- ▶ Hence the disjunction and conjunction are finite

# FO-definable languages

---

$$\psi_L = \bigvee_{v \in L} \bigwedge_{w \notin L} \psi_{vw}$$

- ▶ Each  $\psi_{vw}$  has rank atmost  $k$
- ▶ Upto equivalence, there are finitely many formulae of rank  $k$
- ▶ Hence the disjunction and conjunction are finite
- ▶  $\psi_L$  is a proper formula (of finite size)

# FO-definable languages

---

$$\psi_L = \bigvee_{v \in L} \bigwedge_{w \notin L} \psi_{vw}$$

- ▶ Each  $\psi_{vw}$  has rank atmost  $k$
- ▶ Upto equivalence, there are finitely many formulae of rank  $k$
- ▶ Hence the disjunction and conjunction are finite
- ▶  $\psi_L$  is a proper formula (of finite size)
- ▶  $\psi_L$  captures  $L$  since each  $v \in L$  satisfies  $\bigwedge_{w \notin L} \psi_{vw}$  while none of the  $w \notin L$  satisfy  $\bigwedge_{w \notin L} \psi_{vw}$

# FO-definable languages

Given a property  $\mathcal{K}$ , if for any pair  $v \in \mathcal{K}$  and  $w \notin \mathcal{K}$ , spoiler has a winning strategy in the  $k$ -round EF game on  $v$  and  $w$ , then there is a rank  $k$  FO formula  $\varphi_{\mathcal{K}}$  that defines the property  $\mathcal{K}$ .

$$\varphi_{\mathcal{K}} = \bigvee_{v \in \mathcal{K}} \bigwedge_{w \notin \mathcal{K}} \psi_{vw}$$

where  $\psi_{vw}$  is as explained in the previous slide.

- ▶ Note that  $k$  is fixed in the above, and is independent of the choices of the words.

# Implications of the Game on FO definability

---

## FO Definability

$L$  is FO definable  $\Rightarrow$  there exists an  $r$  such that for every  $(w_1, w_2)$  pair, such that  $w_1 \in L, w_2 \notin L$ , spoiler wins in  $r$  rounds

# Implications of the Game on FO definability

---

## FO Definability

$L$  is FO definable  $\Rightarrow$  there exists an  $r$  such that for every  $(w_1, w_2)$  pair, such that  $w_1 \in L, w_2 \notin L$ , spoiler wins in  $r$  rounds

## Non FO Definability

For all  $r \geq 0$ , there exists a  $(w_1, w_2)$  pair with  $w_1 \notin L, w_2 \in L$ , duplicator wins in  $r$  rounds  $\Rightarrow L$  is not FO definable

# $(aa)^*$ is not $FO[<]$ Definable

---

- ▶ Assume that there is a sentence  $\varphi$  that defines words of even length, with  $c(\varphi) = r$ .
- ▶ Then,  $a^i \models \varphi$  iff  $i$  is even
- ▶ Show that for all  $r > 0$ ,  $a^{2^r} \sim_r a^{2^r - 1}$

# $(aa)^*$ is not $FO[<]$ Definable

---

- ▶ Base case :  $(a, \emptyset)(a, \emptyset)$  and  $(a, \emptyset)$  for  $r = 1$
- ▶ In one round, duplicator wins on  $(a, \emptyset)(a, \emptyset)$  and  $(a, \emptyset)$

# $(aa)^*$ is not $FO[<]$ Definable

---

- ▶ Base case :  $(a, \emptyset)(a, \emptyset)$  and  $(a, \emptyset)$  for  $r = 1$
- ▶ In one round, duplicator wins on  $(a, \emptyset)(a, \emptyset)$  and  $(a, \emptyset)$
- ▶ Consider  $(aaaa, aaa)$  for  $r = 3$ . Who wins?
- ▶ Consider  $(aaaa, aaa)$  for  $r = 2$ . Who wins?

# $(aa)^*$ is not $FO[<]$ Definable

---

- ▶ Show that for all  $k \geq 2^r - 1$ , duplicator has a winning strategy for the  $r$ -round game in  $(a^k, a^{k+1})$ , for all  $r \geq 0$
- ▶ Induct on  $r$
- ▶ If  $r = 1$ , then on  $(a, aa)$  duplicator wins in one round
- ▶ Assume now that the claim is true for  $\leq r - 1$

# $(aa)^*$ is not $FO[<]$ Definable

---

- ▶ Let  $k \geq 2^r - 1$ , and consider the structures

$$(a^k, a^{k+1})$$

- ▶ Spoiler puts pebble  $z_1$  in one of the words obtaining

$$(a, \emptyset)^s (a, \{z_1\}) (a, \emptyset)^t$$

# $(aa)^*$ is not $FO[<]$ Definable

---

- ▶ Let  $k \geq 2^r - 1$ , and consider the structures

$$(a^k, a^{k+1})$$

- ▶ Spoiler puts pebble  $z_1$  in one of the words obtaining

$$(a, \emptyset)^s (a, \{z_1\}) (a, \emptyset)^t$$

- ▶  $s \leq \frac{k-1}{2}$  or  $t \leq \frac{k-1}{2}$

# $(aa)^*$ is not $FO[<]$ Definable

- ▶ Assume  $s \leq \frac{k-1}{2}$ . Duplicator puts her pebble  $z_1$  on the  $(s+1)$ th letter of the other word obtaining

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

where  $t' = t + 1$  or  $t' = t - 1$ .

# $(aa)^*$ is not $FO[<]$ Definable

- ▶ Assume  $s \leq \frac{k-1}{2}$ . Duplicator puts her pebble  $z_1$  on the  $(s+1)$ th letter of the other word obtaining

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

where  $t' = t + 1$  or  $t' = t - 1$ .

- ▶ The structures after round 1 are thus

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^t, (a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

# $(aa)^*$ is not $FO[<]$ Definable

- ▶ Assume  $s \leq \frac{k-1}{2}$ . Duplicator puts her pebble  $z_1$  on the  $(s+1)$ th letter of the other word obtaining

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

where  $t' = t + 1$  or  $t' = t - 1$ .

- ▶ The structures after round 1 are thus

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^t, (a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

- ▶ We have  $2^r - 1 \leq k = \min(t, t') + s + 1 \leq \min(t, t') + \frac{k-1}{2} + 1$

# $(aa)^*$ is not $FO[<]$ Definable

- ▶ Assume  $s \leq \frac{k-1}{2}$ . Duplicator puts her pebble  $z_1$  on the  $(s+1)$ th letter of the other word obtaining

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

where  $t' = t + 1$  or  $t' = t - 1$ .

- ▶ The structures after round 1 are thus

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^t, (a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

- ▶ We have  $2^r - 1 \leq k = \min(t, t') + s + 1 \leq \min(t, t') + \frac{k-1}{2} + 1$
- ▶ Hence  $\min(t, t') \geq \frac{k-1}{2} \geq 2^{r-1} - 1$

# $(aa)^*$ is not $FO[<]$ Definable

- ▶ Assume  $s \leq \frac{k-1}{2}$ . Duplicator puts her pebble  $z_1$  on the  $(s+1)$ th letter of the other word obtaining

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

where  $t' = t + 1$  or  $t' = t - 1$ .

- ▶ The structures after round 1 are thus

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^t, (a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

- ▶ We have  $2^r - 1 \leq k = \min(t, t') + s + 1 \leq \min(t, t') + \frac{k-1}{2} + 1$
- ▶ Hence  $\min(t, t') \geq \frac{k-1}{2} \geq 2^{r-1} - 1$
- ▶ By inductive hypothesis, duplicator has a winning strategy for the  $r-1$  round game on  $(a^t, a^{t'})$ .

# Duplicator's Win

---

- ▶ Use the duplicator's winning strategy for the  $r - 1$  round game on  $(a^t, a^{t'})$ , to obtain a winning strategy in  $r - 1$  rounds on

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^t, (a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

# Duplicator's Win

---

- ▶ Use the duplicator's winning strategy for the  $r - 1$  round game on  $(a^t, a^{t'})$ , to obtain a winning strategy in  $r - 1$  rounds on

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^t, (a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

- ▶ Whenever spoiler plays on a structure on letter  $i \leq s + 1$ , duplicator plays on the same position on the other structure

# Duplicator's Win

---

- ▶ Use the duplicator's winning strategy for the  $r - 1$  round game on  $(a^t, a^{t'})$ , to obtain a winning strategy in  $r - 1$  rounds on

$$(a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^t, (a, \emptyset)^s(a, \{z_1\})(a, \emptyset)^{t'}$$

- ▶ Whenever spoiler plays on a structure on letter  $i \leq s + 1$ , duplicator plays on the same position on the other structure
- ▶ When spoiler plays at a position  $i > s + 1$  in either word, duplicator plays in the part of the other word  $> s + 1$  using her winning strategy in  $(a^t, a^{t'})$

# Duplicator's Win

---

- ▶ At the end of  $r$  rounds, we have structures  $w'_1, w'_2$ .
- ▶ For  $i \leq s + 1$ , pebble  $z_j$  appears at position  $i$  of  $w'_1$  iff pebble  $z_j$  appears at position  $i$  of  $w'_2$
- ▶ Lets erase the first  $s + 1$  letters in  $w'_1, w'_2$ , obtaining  $v'_1, v'_2$
- ▶  $v'_1, v'_2$  are the words that result after  $r' \leq r - 1$  rounds of play on  $(a^t, a^{t'})$ . Recall that duplicator won this.
- ▶ Show that  $w'_1, w'_2$  satisfy the same atomic formulae

# Duplicator's Win

---

- ▶ Atomic Formulae :  $Q_a(z_j)$  : Both  $w'_1, w'_2$  satisfy this.
- ▶  $w'_1 \models z_i < z_j$ . If  $z_i, z_j$  are in the first  $s + 1$  letters, then  
 $w'_2 \models z_i < z_j$ .
- ▶ If  $z_i, z_j$  occur in the last  $|w'_1| - s - 1$  positions, then  $v'_1 \models z_i < z_j$ .  
By duplicator's win in  $(a^t, a^{t'})$ ,  $v'_2 \models z_i < z_j$
- ▶ If  $z_i$  appears among the first  $s + 1$  letters and  $z_j$  after the first  $s + 1$  letters of  $w'_1$ , same is true in  $w'_2$ .

# Historically Speaking

---

The games that we saw are due to Ehrenfeucht and Fraïssé

Reference: Finite Automata, Formal Logic and Circuit Complexity, by Howard Straubing.