COMP2111 Week 10
Term 1, 2019
Course review

Course goals

- Reinforce concepts from Discrete Mathematics
- Emphasise the connections between Discrete Mathematics and Computer Science
- Use mathematical concepts to reason about programs:
  - Acquire (and understand) languages to formally specify systems
  - Acquire (and understand) structures to formally model systems
  - Learn how to prove that a program satisfies its specification

Course goals

- Reinforce concepts from Discrete Mathematics
- Emphasise the connections between Discrete Mathematics and Computer Science
- Use mathematical concepts to reason about programs:
  - Acquire (and understand) languages to formally specify systems
  - Acquire (and understand) structures to formally model systems
  - Learn how to prove that a program satisfies its specification

Course goals

- Reinforce concepts from Discrete Mathematics
- Emphasise the connections between Discrete Mathematics and Computer Science
- Use mathematical concepts to reason about programs:
    - Acquire (and understand) languages to formally specify systems
    - Acquire (and understand) structures to formally model systems
    - Learn how to prove that a program satisfies its specification

Course goals

- Reinforce concepts from Discrete Mathematics
- Emphasise the connections between Discrete Mathematics and Computer Science
- Use mathematical concepts to reason about programs:
  - Acquire (and understand) languages to formally specify systems
  - Acquire (and understand) structures to formally model systems
  - Learn how to prove that a program satisfies its specification

Course goals

- Reinforce concepts from Discrete Mathematics
- Emphasise the connections between Discrete Mathematics and Computer Science
- Use mathematical concepts to reason about programs:
    - Acquire (and understand) languages to formally specify systems
    - Acquire (and understand) structures to formally model systems
    - Learn how to prove that a program satisfies its specification

Course goals

- Reinforce concepts from Discrete Mathematics
- Emphasise the connections between Discrete Mathematics and Computer Science
- Use mathematical concepts to reason about programs:
  - Acquire (and understand) languages to formally specify systems
  - Acquire (and understand) structures to formally model systems
  - Learn how to prove that a program satisfies its specification

Assessment details

- Assignment 1: 20%
- Assignment 2: 15%
- Assignment 3: 15%
- Final exam: 50%

NB

You **must** *achieve 40% on the final exam and 50% overall to pass.*

Final exam

Goal: Assess your understanding of the concepts in this course

Requires you to demonstrate:

- Understanding of the topics covered
- Ability to apply these concepts and explain how they work

Lectures, assignments and tutorials have built you up to this point.

Exam details

Wednesday, 15 May, 8:45AM
Randwick Racecourse Ballroom

- 5 short answer questions and 5 long answer questions
- Topics taken from all content up to (and including) Context-Free Grammars.
- Each short answer question is worth 4 marks
  Each long answer question is worth 20 marks
  Total exam marks = 120 (i.e. 1 mark/minute)
- Time allowed: 120 minutes + 10 minutes reading time
- One handwritten or typed A4-sized sheet (double-sided is ok) of your own notes
- Formula sheet with rules/laws included

Exam structure

Short answer questions:

- Short questions designed to check your understanding of definitions
- 2–3 sentence justifications if necessary
- Answer in exam booklet **not on exam paper**

Long answer questions:

- "Proof" questions designed to examine your understanding at a deeper level
- Answer in exam booklet: **start each question on a new page**
- Put the order questions were attempted on the front.

Exam structure

Short answer questions:

- Short questions designed to check your understanding of definitions
- 2–3 sentence justifications if necessary
- Answer in exam booklet **not on exam paper**

Long answer questions:

- “Proof” questions designed to examine your understanding at a deeper level
- Answer in exam booklet: **start each question on a new page**
- Put the order questions were attempted on the front.

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Fundamentals

- Sets
- Languages
- Relations and Functions

Need to know for this course:

- Formal language definitions
- Relation/function definitions
- Equivalence relations
- Partial orders

Relation/Function definitions

- Reflexive, anti-reflexive
- Symmetric, anti-symmetric
- Transitive
- Composition, converse, inverse
- Injective, surjective, bijective

Example (Properties)

Example

Common relations and their properties

	(R)	(AR)	(S)	(AS)	(T)
$=$	✓		✓	✓	✓
$\leq$	✓			✓	✓
$<$		✓		✓	✓
$\emptyset$		✓	✓	✓	✓
$\mathcal{U}$	✓		✓		✓
$\mid$	✓			✓	✓

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Set Theory and Boolean Algebras

- Sets
- Boolean Algebras

Need to know for this course:

- Proofs using the Laws of Set Operations
- Proofs using the Laws of Boolean Algebras
- Principle of duality

Definition: Boolean Algebra

A *Boolean algebra* is a structure $(T, \vee, \wedge, ', 0, 1)$ where

- $0, 1 \in T$
- $\vee : T \times T \to T$ (called **join**)
- $\wedge : T \times T \to T$ (called **meet**)
- $' : T \to T$ (called **complementation**)

and the following laws hold for all $x, y, z \in T$:

commutative:
- $x \vee y = y \vee x$
- $x \wedge y = y \wedge x$

associative:
- $(x \vee y) \vee z = x \vee (y \vee z)$
- $(x \wedge y) \wedge z = x \wedge (y \wedge z)$

distributive:
- $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
- $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

identity: $x \vee 0 = x, \quad x \wedge 1 = x$

complementation: $x \vee x' = 1, \quad x \wedge x' = 0$

Example (Proof using Laws of Set Operations)

Example (Idempotence of $\cup$)

$$
\begin{aligned}
A &= A \cup \emptyset && \text{(Identity)} \\
&= A \cup (A \cap A^c) && \text{(Complementation)} \\
&= (A \cup A) \cap (A \cup A^c) && \text{(Distributivity)} \\
&= (A \cup A) \cap \mathcal{U} && \text{(Complementation)} \\
&= (A \cup A) && \text{(Identity)}
\end{aligned}
$$

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Inductive definitions, datatypes, and proofs

- Recursion
- Recursive datatypes and functions
- Induction and structural induction

Need to know for this course:

- How to define structures/functions recursively
- How to prove properties of recursively defined structures

Inductive definitions

Inductively defined structure:

- Base case(s): "Minimal" structures
- Inductive case(s): How to build more complex structures from simple ones

Recursively defined functions:

- Base case(s): Terminating conditions
- Recursive case(s): Call functions with "smaller" inputs

Example (Inductive definitions)

Example (Inductively defined structures)

- Natural numbers:
    - Base case: 0
    - Inductive case: $n+1$ where n is a Natural number
- Σ^*:
    - Base case: λ
    - Inductive case: aw where $a \in \Sigma$ and $w \in \Sigma^*$
- Well-formed formulas
- $\mathcal{L}$ programs
- Regular expressions

Example (Inductive definitions)

Example (Recursively defined functions)

- $\mathsf{length} : \Sigma^* \to \Sigma^*$
  - Base case: $\mathsf{length}(\lambda) = 0$
  - Inductive case: $\mathsf{length}(aw) = 1 + \mathsf{length}(w)$
- $[\![\cdot]\!]_v : \textsc{WFFs} \to \mathbb{B}$
- $[\![\cdot]\!]^{\eta}_{\mathcal{M}} : \textsc{WFFs} \to \mathbb{B}$
- $[\![\cdot]\!] : \textsc{Programs} \to \mathsf{Pow}(\textsc{Env} \times \textsc{Env})$
- $L(\cdot) : \textsc{RegExp} \to \mathsf{Pow}(\Sigma^*)$

Structural Induction

Basic induction allows us to assert properties over **all natural numbers**. The induction scheme (layout) uses the recursive definition of $\mathbb{N}$.

The induction schemes can be applied not only to natural numbers (and integers) but to any partially ordered set in general – especially those defined recursively.

The basic approach is always the same — we need to verify that

- **[B]** the property holds for all minimal objects — objects that have no predecessors; they are usually very simple objects allowing immediate verification
- **[I]** for any given object, if the property in question holds for all its predecessors ('smaller' objects) then it holds for the object itself

Example (Structural Induction)

Example

Let $P(w)$ be the proposition that, for all $v \in \Sigma^*$:

$$\text{length}(wv) = \text{length}(w) + \text{length}(v).$$

We will show that $P(w)$ holds for all $w \in \Sigma^*$ by structural induction on w.

Base case ($w = \lambda$):

$$\begin{aligned} \text{length}(\lambda v) &= \text{length}(v) && \text{(concat.B)} \\ &= 0 + \text{length}(v) \\ &= \text{length}(w) + \text{length}(v) && \text{(length.B)} \end{aligned}$$

Example (Structural Induction)

Example

Let $P(w)$ be the proposition that, for all $v \in \Sigma^*$:

$$\text{length}(wv) = \text{length}(w) + \text{length}(v).$$

We will show that $P(w)$ holds for all $w \in \Sigma^*$ by structural induction on w.

Inductive case ($w = aw'$): Assume that $P(w')$ holds. That is, for all $v \in \Sigma^*$: $\text{length}(w'v) = \text{length}(w') + \text{length}(v)$. Then:

$$\begin{aligned}
\text{length}((aw')v) &= \text{length}(a(w'v)) && \text{(concat.I)} \\
&= 1 + \text{length}(w'v) && \text{(length.I)} \\
&= 1 + \text{length}(w') + \text{length}(v) && \text{(IH)} \\
&= \text{length}(aw') + \text{length}(v) && \text{(length.I)}
\end{aligned}$$

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Propositional Logic

- Well-formed formulas (SYNTAX)
- Truth assignments and valuations (SEMANTICS)
- Conjunctive/Disjunctive Normal Forms

Need to know for this course

- Difference between syntax and semantics
- CNF/DNF definitions and (any) technique for converting a formula into CNF/DNF

Syntax of Prop. Logic (Well-formed formulas)

Let $\textsc{Prop} = \{p, q, r, \ldots\}$ be a set of propositional letters. Consider the alphabet

$$\Sigma = \textsc{Prop} \cup \{\top, \bot, \neg, \wedge, \vee, \to, \leftrightarrow, (,)\}.$$

The **well-formed formulas** (wffs) over $\textsc{Prop}$ is the smallest set of words over Σ such that:

- $\top$, $\bot$ and all elements of $\textsc{Prop}$ are wffs
- If φ is a wff then $\neg\varphi$ is a wff
- If φ and ψ are wffs then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, and $(\varphi \leftrightarrow \psi)$ are wffs.

Semantics of Propositional Logic (Valuations)

A **truth assignment** (or **model**) is a function $v : \textsc{Prop} \to \mathbb{B}$

We can extend v to a function $[\![\cdot]\!]_v : \textsc{WFFs} \to \mathbb{B}$ recursively:

- $[\![\top]\!]_v = \texttt{true}$, $[\![\bot]\!]_v = \texttt{false}$
- $[\![p]\!]_v = v(p)$
- $[\![\neg\varphi]\!]_v = ![\![\varphi]\!]_v$
- $[\![(\varphi \wedge \psi)]\!]_v = [\![\varphi]\!]_v \ \&\& \ [\![\psi]\!]_v$
- $[\![(\varphi \vee \psi)]\!]_v = [\![\varphi]\!]_v \parallel [\![\psi]\!]_v$
- $[\![(\varphi \to \psi)]\!]_v = ![\![\varphi]\!]_v \parallel [\![\psi]\!]_v$
- $[\![(\varphi \leftrightarrow \psi)]\!]_v = (![\![\varphi]\!]_v \parallel [\![\psi]\!]_v) \ \&\& \ (![\![\psi]\!]_v \parallel [\![\varphi]\!]_v)$

Satisfiability, Entailment, Equivalence

- A valuation **satisfies** a theory T if $[\![\varphi]\!]_v = \texttt{true}$ for every $\varphi \in T$
- A theory/formula is **satisfiable** if there is some valuation that satisfies it
- A formula is a **tautology** if every valuation satisfies it
- Entailment: $T \models \varphi$ if for every valuation that satisfies T, we have $[\![\varphi]\!]_v = \texttt{true}$
- Logical equivalence: $\varphi \equiv \psi$ if $[\![\varphi]\!]_v = [\![\psi]\!]_v$ for all valuations

Example (Working with Propositional Logic)

Example

You are planning a party, but your friends are a bit touchy about who will be there.

1. If John comes, he will get very hostile if Sarah is there.
2. Sarah will only come if Kim will be there also.
3. Kim says she will not come unless John does.

Who can you invite without making someone unhappy?

Example (Working with Propositional Logic)

Example

Translation to logic: let J, S, K represent "John (Sarah, Kim) comes to the party". Then the constraints are:

1. $J \rightarrow \neg S$
2. $S \rightarrow K$
3. $K \rightarrow J$

Thus, for a successful party to be possible, we want the formula $\phi = (J \rightarrow \neg S) \land (S \rightarrow K) \land (K \rightarrow J)$ to be satisfiable.

Example (Working with Propositional Logic)

Example

Truth table: Each row corresponds to a valuation

J	K	S	$J \to \neg S$	$S \to K$	$K \to J$	ϕ
F	F	F				
F	F	T		F		F
F	T	F			F	F
F	T	T			F	F
T	F	F				
T	F	T	F	F		F
T	T	F				
T	T	T	F			F

Conclusion: a party satisfying the constraints can be held. Invite nobody, or invite John only, or invite Kim and John.

Conjunctive/Disjunctive Normal Forms

CNFs and DNFs are *syntactic* forms:

Literal: A propositional variable or the negation of a propositional variable

Clause: A CNF-clause is a disjunction ($\vee$) of literals. A DNF-clause is a conjunction ($\wedge$) of literals

CNF/DNF: A formula is in CNF (DNF) if it is a conjunction (disjunction) of CNF-clauses (DNF-clauses).

Theorem

Every propositional formula is logically equivalent to one in CNF and one in DNF.

Example (CNF/DNF)

Example

Consider $\varphi = (y \to x)$:

x	y	$y \to x$
F	F	T
F	T	F
T	F	T
T	T	T

Then φ is logically equivalent to:

- $(\neg x \land \neg y) \lor (x \land \neg y) \lor (x \land y)$ (DNF)
- $\neg y \lor (x \land y)$ (DNF)
- $\neg y \lor x$ (CNF and DNF)

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Predicate Logic

- Well-formed formulas (SYNTAX)
- Models and environments (SEMANTICS)

Need to know for this course

- Translate requirements into Propositonal and/or Predicate logic
- Syntax and Semantics definitions
- Satifiability, Validity, Logical equivalence

Syntax of Predicate (First-Order) Logic

Given a vocabulary (predicate symbols, function symbols, constant symbols):

- **Terms** defined recursively over a set of variables
- **Formulas** defined recursively:
  - Atomic formulas: built from predicates and equality, and terms
  - Other formulas: built recursively using Boolean connectives and quantifiers
- Parentheses usage relaxed to aid readability
- Free variables "captured" syntactically with $\varphi(x)$ notation

Semantics of Predicate (First-Order) Logic

Given a vocabulary (predicate symbols, function symbols, constant symbols):

- A **model** interprets all the symbols of the vocabulary over some domain
- $[\![\varphi]\!]_{\mathcal{M}}$ is a **relation** on $\mathrm{Dom}(\mathcal{M})$
- An *environment* assigns variables to elements of $\mathrm{Dom}(\mathcal{M})$
- $[\![\varphi]\!]^{\eta}_{\mathcal{M}}$ "looks-up" the tuple defined by η in the relation $[\![\varphi]\!]_{\mathcal{M}}$ and returns an element of $\mathbb{B}$ depending on its presence

Satisfiability and Validity

A formula, φ, is:

- **Satisfiable** if there is some model and environment (interpretation) such that $[\![\varphi]\!]^{\eta}_{\mathcal{M}} = \texttt{true}$ ($\mathcal{M}, \eta \models \varphi$)
- **True in a model** $\mathcal{M}$ if for all environments η, $\mathcal{M}, \eta \models \varphi$
- A **Logical validity** if it is true in all models
- A **Logical consequence** of a set of formulas T (written $T \models \varphi$) if $\mathcal{M}, \eta \models \varphi$ for all interpretations which satisfy every element of T.
- **Logically equivalent** to a formula ψ if $[\![\varphi]\!]^{\eta}_{\mathcal{M}} = [\![\psi]\!]^{\eta}_{\mathcal{M}}$ for all interpretations

Satisfiability and Validity

Theorem

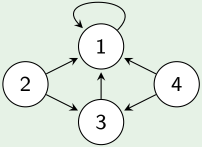
- $\emptyset \models \varphi$ *if, and only if,* φ *is a logical validity*
- $\varphi \models \psi$ *if, and only if,* $\varphi \to \psi$ *is a logical validity*
- $\varphi \equiv \psi$ *if, and only if,* $\varphi \leftrightarrow \psi$ *is a logical validity*

Example (Interpretations)

Example

$$\forall x \forall y((y = x + 1) \rightarrow (x \leq y))$$

- $\mathbb{N}$ with the standard definitions of $\leq$, $+$, and 1: true
- $\{0, 1, 2, 3, 4\}$ with the standard definition of $\leq$ and 1, and $m + n$ defined as $m + n \pmod 5$: false
- The directed graph $G = (V, E)$ shown below with $\leq = E$; and $v + w$ defined to be w: true

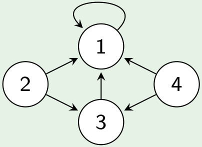


Example (Interpretations)

Example

$$\forall x \forall y ((y = x + 1) \rightarrow (x \leq y))$$

- $\mathbb{N}$ with the standard definitions of $\leq$, $+$, and 1: `true`
- $\{0, 1, 2, 3, 4\}$ with the standard definition of $\leq$ and 1, and $m + n$ defined as $m + n \pmod 5$: `false`
- The directed graph $G = (V, E)$ shown below with $\leq = E$; and $v + w$ defined to be w: `true`



Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Natural Deduction

- Formal proofs
- Natural Deduction for Propositional Logic
- Natural Deduction for Predicate Logic

Need to know for this course:

- One formal proof style
- How to present proofs in the style
- Rules of Natural Deduction for Propositional Logic
- Rules of Natural Deduction for Predicate Logic
- Relation between Proofs and fundamental logical concepts

Formal proof styles

Three main styles:

- Tabular
- Fitch-style
- Tree

Each logical step should indicate:

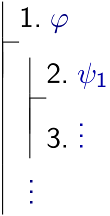
- A line number for later reference
- The (undischarged) assumptions required to make the derivation
- The result of the derivation
- The derivation rule used
- Which preiously computed results were required to for the rule

Proof styles: Table

Line	Premises	Formula	Rule	Ref
⋮	⋮	⋮	⋮	⋮

Advantages:	Easy to typeset; Compact representation
Disadvantages:	Structure can be difficult to follow

Proof styles: Fitch-style



1. φ

2. ψ_1

3. $\vdots$

$\vdots$

Advantages: Scope of assumptions is clear;
Style used in online checker

Disadvantages: Rule application often not obvious

Proof styles: Tree

$$\dfrac{A \qquad \dfrac{A \to B \qquad A}{B}\;(\to\text{-E})}{A \land B}\;(\land\text{-I})$$

Advantages: Proof structure is clear;
Construct directly from rules

Disadvantages: Often unwieldy presentations

Natural Deduction

$T \vdash \varphi$: **Prove** φ from T

15+7 Inference rules based on introducing/eliminating boolean operators:

$$\frac{A \qquad B}{A \wedge B}\,(\wedge\text{-I}) \qquad \frac{A \wedge B}{A}\,(\wedge\text{-E1}) \qquad \frac{A \wedge B}{B}\,(\wedge\text{-E2})$$

$$\frac{A}{A \vee B}\,(\vee\text{-I1}) \qquad \frac{B}{A \vee B}\,(\vee\text{-I2}) \qquad \frac{A \vee B \qquad \begin{array}{c}[A]\\ \vdots\\ C\end{array} \qquad \begin{array}{c}[B]\\ \vdots\\ C\end{array}}{C}\,(\vee\text{-E})$$

Natural Deduction

$T \vdash \varphi$: **Prove** φ from T

15+7 Inference rules based on introducing/eliminating boolean operators:

$$\frac{A \qquad B}{A \wedge B}\ (\wedge\text{-I}) \qquad \frac{A \wedge B}{A}\ (\wedge\text{-E1}) \qquad \frac{A \wedge B}{B}\ (\wedge\text{-E2})$$

$$\frac{A}{A \vee B}\ (\vee\text{-I1}) \qquad \frac{B}{A \vee B}\ (\vee\text{-I2}) \qquad \frac{A \vee B \qquad \begin{matrix}[A]\\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[B]\\ \vdots \\ C\end{matrix}}{C}\ (\vee\text{-E})$$

Natural Deduction

$$\dfrac{\begin{matrix}[A]\\ \vdots\\ B\end{matrix}}{A \to B}\ (\to\text{-I}) \qquad\qquad \dfrac{A \to B \qquad A}{B}\ (\to\text{-E})$$

$$\dfrac{\begin{matrix}[A]\\ \vdots\\ B\end{matrix} \qquad \begin{matrix}[B]\\ \vdots\\ A\end{matrix}}{A \leftrightarrow B}\ (\leftrightarrow\text{-I}) \qquad \dfrac{A \leftrightarrow B \qquad A}{B}\ (\leftrightarrow\text{-E1}) \qquad \dfrac{A \leftrightarrow B \qquad B}{A}\ (\leftrightarrow\text{-E1})$$

Natural Deduction

$$\begin{array}{c} [A] \\ \vdots \\ \bot \\ \hline \neg A \end{array} \;(\neg\text{-I}) \qquad\qquad \frac{A \qquad \neg A}{\bot}\;(\neg\text{-E})$$

$$\begin{array}{c} [\neg A] \\ \vdots \\ \bot \\ \hline A \end{array} \;(\text{IP}) \qquad\qquad \frac{\bot}{A}\;(\text{X})$$

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	3, 4
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	8, 10
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \lor (B \land C) \vdash (A \lor B) \land (A \lor C)$

Line	Premises	Formula	Rule	References
1		$A \lor (B \land C)$	Premise	
2		A	Premise	
3	2	$A \lor B$	$\lor$-I1	2
4	2	$A \lor C$	$\lor$-I1	2
5	2	$(A \lor B) \land (A \lor C)$	$\land$-I	3, 4
6		$(B \land C)$	Premise	
7	6	B	$\land$-E1	6
8	6	$A \lor B$	$\lor$-I2	7
9	6	C	$\land$-E2	6
10	6	$A \lor C$	$\lor$-I2	9
11	6	$(A \lor B) \land (A \lor C)$	$\land$-I	8, 10
12	1	$(A \lor B) \land (A \lor C)$	$\lor$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	3, 4
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	8, 10
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	3, 4
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	8, 10
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	3, 4
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	8, 10
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	3, 4
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	8, 10
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	3, 4
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	8, 10
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \lor (B \land C) \vdash (A \lor B) \land (A \lor C)$

Line	Premises	Formula	Rule	References
1		$A \lor (B \land C)$	Premise	
2		A	Premise	
3	2	$A \lor B$	$\lor$-I1	2
4	2	$A \lor C$	$\lor$-I1	2
5	2	$(A \lor B) \land (A \lor C)$	$\land$-I	3, 4
6		$(B \land C)$	Premise	
7	6	B	$\land$-E1	6
8	6	$A \lor B$	$\lor$-I2	7
9	6	C	$\land$-E2	6
10	6	$A \lor C$	$\lor$-I2	9
11	6	$(A \lor B) \land (A \lor C)$	$\land$-I	8, 10
12	1	$(A \lor B) \land (A \lor C)$	$\lor$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	$3, 4$
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	$8, 10$
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	$5, 11$

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	3, 4
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	8, 10
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	$3, 4$
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	$8, 10$
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	$5, 11$

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	3, 4
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	8, 10
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	5, 11

Example (Tabular proof)

Example

Prove: $A \vee (B \wedge C) \vdash (A \vee B) \wedge (A \vee C)$

Line	Premises	Formula	Rule	References
1		$A \vee (B \wedge C)$	Premise	
2		A	Premise	
3	2	$A \vee B$	$\vee$-I1	2
4	2	$A \vee C$	$\vee$-I1	2
5	2	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	$3, 4$
6		$(B \wedge C)$	Premise	
7	6	B	$\wedge$-E1	6
8	6	$A \vee B$	$\vee$-I2	7
9	6	C	$\wedge$-E2	6
10	6	$A \vee C$	$\vee$-I2	9
11	6	$(A \vee B) \wedge (A \vee C)$	$\wedge$-I	$8, 10$
12	1	$(A \vee B) \wedge (A \vee C)$	$\vee$-E	$5, 11$

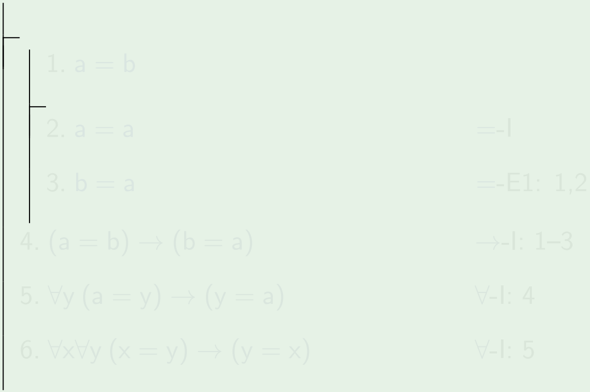
Natural Deduction (Predicate Logic only)

$\dfrac{}{a = a}$ (=-I)	$\dfrac{a = b \qquad A(a)}{A(b)}$ (=-E1)	$\dfrac{a = b \qquad A(b)}{A(a)}$ (=-E2)
$\dfrac{A(c) \qquad (1,2,3)}{\forall x A(x)}$ (∀-I)	$\dfrac{A(c) \qquad (2)}{\exists x A(x)}$ (∃-I)	$\dfrac{\forall x A(x)}{A(c)}$ (∀-E)
$\dfrac{\exists x A(x) \qquad \begin{matrix}[A(c)] \\ \vdots \\ B\end{matrix} \qquad (1,2,4)}{B}$ (∃-E)		(1): c is arbitrary (2): x is not free in $A(c)$ (3): c is not free in $A(x)$ (4): c is not free in B

Example (Fitch-style proof)

Example

Prove: $\vdash \forall x \forall y\ (x = y) \rightarrow (y = x)$



1. a = b	
2. a = a	=-I
3. b = a	=-E1: 1,2
4. (a = b) → (b = a)	→-I: 1–3
5. ∀y (a = y) → (y = a)	∀-I: 4
6. ∀x∀y (x = y) → (y = x)	∀-I: 5

Example (Fitch-style proof)

Example

Prove: $\vdash \forall x \forall y\,(x = y) \rightarrow (y = x)$

1. a = b	
2. a = a	=-I
3. b = a	=-E1: 1,2
4. (a = b) → (b = a)	→-I: 1–3
5. ∀y (a = y) → (y = a)	∀-I: 4
6. ∀x∀y (x = y) → (y = x)	∀-I: 5

Example (Fitch-style proof)

Example

Prove: $\vdash \forall x \forall y \, (x = y) \rightarrow (y = x)$

1. $\mathrm{a} = \mathrm{b}$

2. $\mathrm{a} = \mathrm{a}$ =-I

3. $\mathrm{b} = \mathrm{a}$ =-E1: 1,2

4. $(\mathrm{a} = \mathrm{b}) \rightarrow (\mathrm{b} = \mathrm{a})$ →-I: 1–3

5. $\forall \mathrm{y} \, (\mathrm{a} = \mathrm{y}) \rightarrow (\mathrm{y} = \mathrm{a})$ ∀-I: 4

6. $\forall \mathrm{x} \forall \mathrm{y} \, (\mathrm{x} = \mathrm{y}) \rightarrow (\mathrm{y} = \mathrm{x})$ ∀-I: 5

Example (Fitch-style proof)

Example

Prove: $\vdash \forall x \forall y\, (x = y) \rightarrow (y = x)$

1. a = b	
2. a = a	=-I
3. b = a	=-E1: 1,2
4. (a = b) → (b = a)	→-I: 1–3
5. ∀y (a = y) → (y = a)	∀-I: 4
6. ∀x∀y (x = y) → (y = x)	∀-I: 5

Example (Fitch-style proof)

Example

Prove: $\vdash \forall x \forall y\, (x = y) \to (y = x)$

1. $a = b$	
2. $a = a$	$=$-I
3. $b = a$	$=$-E1: 1,2
4. $(a = b) \to (b = a)$	$\to$-I: 1–3
5. $\forall y\, (a = y) \to (y = a)$	$\forall$-I: 4
6. $\forall x \forall y\, (x = y) \to (y = x)$	$\forall$-I: 5

Example (Fitch-style proof)

Example

Prove: $\vdash \forall x \forall y\, (x = y) \rightarrow (y = x)$

1. $a = b$

2. $a = a$ — $=$-I

3. $b = a$ — $=$-E1: 1,2

4. $(a = b) \rightarrow (b = a)$ — $\rightarrow$-I: 1–3

5. $\forall y\, (a = y) \rightarrow (y = a)$ — $\forall$-I: 4

6. $\forall x \forall y\, (x = y) \rightarrow (y = x)$ — $\forall$-I: 5

Example (Fitch-style proof)

Example

Prove: $\vdash \forall x \forall y\, (x = y) \rightarrow (y = x)$

1. $a = b$	
2. $a = a$	$=$-I
3. $b = a$	$=$-E1: 1,2
4. $(a = b) \rightarrow (b = a)$	$\rightarrow$-I: 1–3
5. $\forall y\, (a = y) \rightarrow (y = a)$	$\forall$-I: 4
6. $\forall x \forall y\, (x = y) \rightarrow (y = x)$	$\forall$-I: 5

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Hoare Logic

- Simple imperative language $\mathcal{L}$
- Hoare triple $\{\varphi\}\, P\, \{\psi\}$ (SYNTAX)
- Derivation rules (PROOFS)
- Semantics for Hoare logic (SEMANTICS)

Need to know for this course:

- Write programs in $\mathcal{L}$.
- Give proofs using the Hoare logic rules
- Definition of $[\![\cdot]\!]$

The language $\mathcal{L}$

The language $\mathcal{L}$ is a simple imperative programming language made up of four statements:

Assignment: $x := e$

where x is a variable and e is an arithmetic expression.

Sequencing: $P;Q$

Conditional: **if** b **then** P **else** Q **fi**

where b is a boolean expression.

While: **while** b **do** P **od**

Hoare triple (Syntax)

$$\{\varphi\}\, P\, \{\psi\}$$

Intuition:
If φ holds in a state of some computational model
then ψ holds in the state reached after a successful execution of P.

$$\vdash \{\varphi\}\, P\, \{\psi\}$$

$\{\varphi\}\, P\, \{\psi\}$ is **derivable** using the proof rules of Hoare Logic

$$\models \{\varphi\}\, P\, \{\psi\}$$

$\{\varphi\}\, P\, \{\psi\}$ is **valid** according to the semantic interpretation.

Hoare triple (Syntax)

$$\{\varphi\}\, P\, \{\psi\}$$

Intuition:
If φ holds in a state of some computational model
then ψ holds in the state reached after a successful execution of P.

$$\vdash \{\varphi\}\, P\, \{\psi\}$$

$\{\varphi\}\, P\, \{\psi\}$ is **derivable** using the proof rules of Hoare Logic

$$\models \{\varphi\}\, P\, \{\psi\}$$

$\{\varphi\}\, P\, \{\psi\}$ is **valid** according to the semantic interpretation.

Hoare triple (Syntax)

$$\{\varphi\}\, P\, \{\psi\}$$

Intuition:
If φ holds in a state of some computational model
then ψ holds in the state reached after a successful execution of P.

$$\vdash \{\varphi\}\, P\, \{\psi\}$$

$\{\varphi\}\, P\, \{\psi\}$ is **derivable** using the proof rules of Hoare Logic

$$\models \{\varphi\}\, P\, \{\psi\}$$

$\{\varphi\}\, P\, \{\psi\}$ is **valid** according to the semantic interpretation.

Hoare logic rules

$$\frac{}{\{\varphi[e/x]\}\ x := e\ \{\varphi\}}\quad \text{(ass)}$$

$$\frac{\{\varphi\}\ P\ \{\psi\} \qquad \{\psi\}\ Q\ \{\rho\}}{\{\varphi\}\ P; Q\ \{\rho\}}\quad \text{(seq)}$$

$$\frac{\{\varphi \wedge g\}\ P\ \{\psi\} \qquad \{\varphi \wedge \neg g\}\ Q\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ g\ \textbf{then}\ P\ \textbf{else}\ Q\ \textbf{fi}\ \{\psi\}}\quad \text{(if)}$$

Hoare logic rules

$$\frac{}{\{\varphi(e)\}\; x := e\; \{\varphi(x)\}} \quad \text{(ass)}$$

$$\frac{\{\varphi\}\; P\; \{\psi\} \qquad \{\psi\}\; Q\; \{\rho\}}{\{\varphi\}\; P; Q\; \{\rho\}} \quad \text{(seq)}$$

$$\frac{\{\varphi \wedge g\}\; P\; \{\psi\} \qquad \{\varphi \wedge \neg g\}\; Q\; \{\psi\}}{\{\varphi\}\; \mathbf{if}\; g\; \mathbf{then}\; P\; \mathbf{else}\; Q\; \mathbf{fi}\; \{\psi\}} \quad \text{(if)}$$

Hoare logic rules

$$\frac{\{\varphi \wedge g\}\ P\ \{\varphi\}}{\{\varphi\}\ \textbf{while}\ g\ \textbf{do}\ P\ \textbf{od}\ \{\varphi \wedge \neg g\}} \quad \text{(loop)}$$

$$\frac{\varphi' \to \varphi \qquad \{\varphi\}\ P\ \{\psi\} \qquad \psi \to \psi'}{\{\varphi'\}\ P\ \{\psi'\}} \quad \text{(cons)}$$

Example (Hoare Logic proof [annotated])

Example

```
                                          ${\rm T{\small RUE}}$
                                          $\{1 = 0!\}$
$f := 1;$                                 $\{f = 0!\}$
$k := 0;$                                 $\{f = k!\}$
while $\neg(k = n)$ do                    $\{(f = k!) \wedge \neg(k = n)\}$
                                          $\{f(k+1) = (k+1)!\}$
    $k := k + 1;$                         $\{fk = k!\}$
    $f := f * k$                          $\{f = k!\}$
od                                        $\{(f = k!) \wedge (k = n)\}$
                                          $\{f = n!\}$
```

Example (Hoare Logic proof [annotated])

Example

```
                                        $\{\textsc{True}\}$
                                        $\{1 = 0!\}$
$f := 1;$                               $\{f = 0!\}$
$k := 0;$                               $\{f = k!\}$
while $\neg(k = n)$ do                  $\{(f = k!) \wedge \neg(k = n)\}$
                                        $\{f(k + 1) = (k + 1)!\}$
    $k := k + 1;$                       $\{fk = k!\}$
    $f := f * k$                        $\{f = k!\}$
od                                      $\{(f = k!) \wedge (k = n)\}$
                                        $\{f = n!\}$
```

Example (Hoare Logic proof [annotated])

Example

```
                                         $\{\textsc{True}\}$
                                         $\{1 = 0!\}$
$f := 1;$                                $\{f = 0!\}$
$k := 0;$                                $\{f = k!\}$
while $\neg(k = n)$ do                   $\{(f = k!) \wedge \neg(k = n)\}$
                                         $\{f(k + 1) = (k + 1)!\}$
    $k := k + 1;$                        $\{fk = k!\}$
    $f := f * k$                         $\{f = k!\}$
od                                       $\{(f = k!) \wedge (k = n)\}$
                                         $\{f = n!\}$
```

Example (Hoare Logic proof [annotated])

Example

```
                                        $\{\textsc{True}\}$
                                        $\{1 = 0!\}$
$f := 1;$                               $\{f = 0!\}$
$k := 0;$                               $\{f = k!\}$
while $\neg(k = n)$ do                  $\{(f = k!) \wedge \neg(k = n)\}$
                                        $\{f(k+1) = (k+1)!\}$
    $k := k + 1;$                       $\{fk = k!\}$
    $f := f * k$                        $\{f = k!\}$
od                                      $\{(f = k!) \wedge (k = n)\}$
                                        $\{f = n!\}$
```

Example (Hoare Logic proof [annotated])

Example

```
                                   $\{\textsc{True}\}$
                                   $\{1 = 0!\}$
$f := 1;$                          $\{f = 0!\}$
$k := 0;$                          $\{f = k!\}$
while $\neg(k = n)$ do             $\{(f = k!) \wedge \neg(k = n)\}$
                                   $\{f(k+1) = (k+1)!\}$
    $k := k + 1;$                  $\{fk = k!\}$
    $f := f * k$                   $\{f = k!\}$
od                                 $\{(f = k!) \wedge (k = n)\}$
                                   $\{f = n!\}$
```

Example (Hoare Logic proof [annotated])

Example

$$
\begin{array}{ll}
 & \{\textsc{True}\} \\
 & \{1 = 0!\} \\
f := 1; & \{f = 0!\} \\
k := 0; & \{f = k!\} \\
\textbf{while } \neg(k = n) \textbf{ do} & \{(f = k!) \wedge \neg(k = n)\} \\
 & \{f(k+1) = (k+1)!\} \\
\quad k := k + 1; & \{fk = k!\} \\
\quad f := f * k & \{f = k!\} \\
\textbf{od} & \{(f = k!) \wedge (k = n)\} \\
 & \{f = n!\}
\end{array}
$$

Example (Hoare Logic proof [annotated])

Example

```
                                        $\{\textsc{True}\}$
                                        $\{1 = 0!\}$
$f := 1;$                               $\{f = 0!\}$
$k := 0;$                               $\{f = k!\}$
while $\neg(k = n)$ do                  $\{(f = k!) \wedge \neg(k = n)\}$
                                        $\{f(k+1) = (k+1)!\}$
    $k := k + 1;$                       $\{fk = k!\}$
    $f := f * k$                        $\{f = k!\}$
od                                      $\{(f = k!) \wedge (k = n)\}$
                                        $\{f = n!\}$
```

Hoare logic semantics

$\textsc{Env}$: set of environments (functions that map variables to numeric values)

$\langle\cdot\rangle : \textsc{Predicates} \to \mathsf{Pow}(\textsc{Env})$, given by:

$$\langle\varphi\rangle := \{\eta \in \textsc{Env} \ : \ [\![\varphi]\!]^{\eta} = \mathtt{true}\}.$$

$[\![\cdot]\!] : \textsc{Programs} \cup \textsc{Predicates} \to \mathsf{Pow}(\textsc{Env} \times \textsc{Env})$

Hoare logic semantics

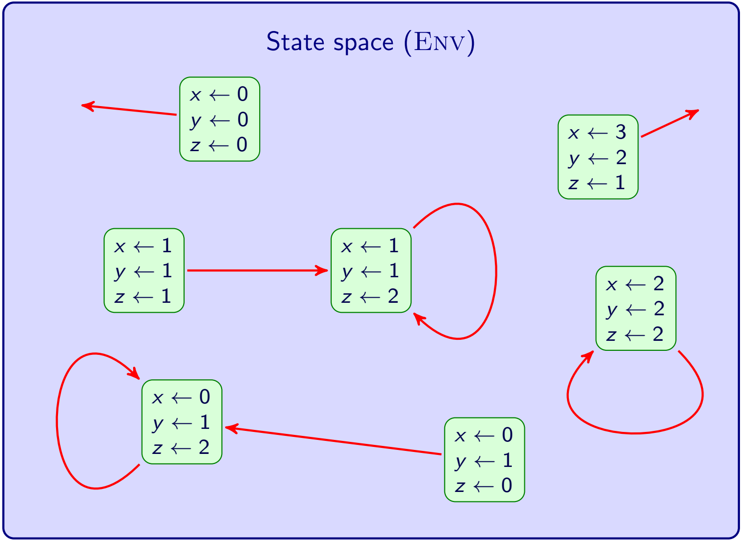
$[\![\cdot]\!] : \textsc{Programs} \cup \textsc{Predicates} \to \mathsf{Pow}(\textsc{Env} \times \textsc{Env})$

For predicates: $[\![\varphi]\!] = \{(\eta, \eta) \ : \ \eta \in \langle\varphi\rangle\}$

For programs: Inductively:

- $[\![x := e]\!] = \{(\eta, \eta') \ : \ \eta' = \eta[x \mapsto [\![e]\!]^{\eta}]\}$
- $[\![P; Q]\!] = [\![P]\!]; [\![Q]\!]$
- $[\![\textbf{if } b \textbf{ then } P \textbf{ else } Q \textbf{ fi}]\!] = [\![b; P]\!] \cup [\![\neg b; Q]\!]$
- $[\![\textbf{while } b \textbf{ do } P \textbf{ od}]\!] = [\![b; P]\!]^{*}; [\![\neg b]\!]$

Example ($[\![z := 2]\!]$)


State space (Env)
$x \leftarrow 0$ $y \leftarrow 0$ $z \leftarrow 0$
$x \leftarrow 3$ $y \leftarrow 2$ $z \leftarrow 1$
$x \leftarrow 1$ $y \leftarrow 1$ $z \leftarrow 1$
$x \leftarrow 1$ $y \leftarrow 1$ $z \leftarrow 2$
$x \leftarrow 2$ $y \leftarrow 2$ $z \leftarrow 2$
$x \leftarrow 0$ $y \leftarrow 1$ $z \leftarrow 2$
$x \leftarrow 0$ $y \leftarrow 1$ $z \leftarrow 0$

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

Transition systems

- Definitions:
    - States and Transitions
    - (Non-)determinism
    - Reachability
- The Invariant Principle
- Termination

Need to know for this course:

- Definitions
- Invariant principle
- Termination proofs

The Invariant Principle

A **preserved invariant** of a transition system is a unary predicate φ on states such that if $\varphi(s)$ holds and $s \to s'$ then $\varphi(s')$ holds.

Invariant principle

If a preserved invariant holds at a state s, then it holds for all states reachable from s.

Termination

A transition system $(S, \to)$ **terminates** from a state s if there is an N such that all runs from s have length at most N.

A **derived variable** is a function $f : S \to \mathbb{R}$.

A derived variable is **strictly decreasing** if $s \to s'$ implies $f(s) > f(s')$.

Theorem

If f is an $\mathbb{N}$-valued, strictly decreasing derived variable, then the length of any run from s is at most $f(s)$.

Example (Transition system)

Example

- States: $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$
- Transition:
  - $(x, y, r) \to (x^2, \frac{y}{2}, r)$ if y is even
  - $(x, y, r) \to (x^2, \frac{y-1}{2}, rx)$ if y is odd
- Preserved invariant: rx^y is a constant
- $\Rightarrow$ All states reachable from $(m, n, 1)$ will satisfy $rx^y = m^n$
- $\Rightarrow$ if $(x, 0, r)$ is reachable from $(m, n, 1)$ then $r = m^n$.

Automata and formal languages

- Deterministic Finite Automata (DFAs)
- Non-deterministic Finite Automata (NFAs)
- Regular expressions
- Myhill-Nerode theorem
- Context-free languages

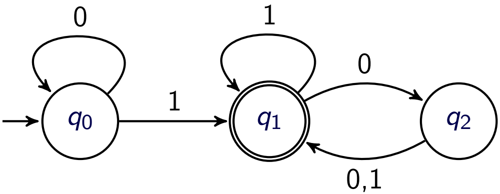
Need to know for this course:

- The language defined by DFAs, NFAs, Regular expressions, and context-free grammars
- Principal applications of the Myhill-Nerode theorem

Topic Summary

- Fundamentals
- Set Theory and Boolean Algebras
- Inductive definitions, datatypes, and proofs
- Propositional Logic
- Predicate Logic
- Natural Deduction
- Hoare Logic
- Transition systems
- Automata and formal languages

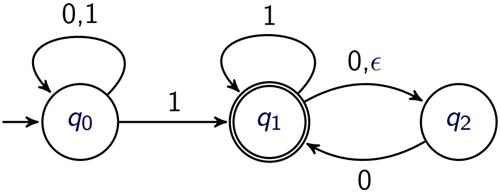
Deterministic Finite Automata



A **deterministic finite automaton (DFA)** is a tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is the input alphabet
- $\delta : Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of final/accepting states

Non-deterministic Finite Automata



A **non-deterministic finite automaton (NFA)** is a tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is the input alphabet
- $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ is the transition relation
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of final/accepting states

Language accepted by a DFA/NFA

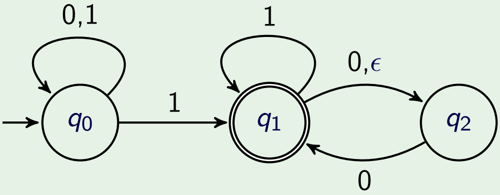
A sequence of input symbols defines a run in a DFA (or several runs in an NFA).

A run is accepting if it ends in a final state. A word is accepted if at least one run is accepting.

$L(M)$ is the set of all words accepted by M.

Example (Language of an NFA)

Example



Accepted words: 1, 01, 11, 10, ...

Regular expressions

Specify language by “matching”

Defined recursively:

- $\emptyset$ is a regular expression
- ϵ is a regular expression
- a is a regular expression for all $a \in \Sigma$
- If E_1, E_2 are regular expressions then so are:
  - $E_1 + E_2$
  - $E_1 E_2$
  - E_1^*

$L(E)$: set of words that match E

Example (Regular expression)

Example

The following words match $(000 + 10)^*01$:

- 01
- 101001
- 000101000001

Myhill-Nerode theorem

Algebraic characterization of regular languages

Syntactic (context) equivalence:

$$v \equiv_L w \quad \text{if, and only if,} \quad \forall z. wz \in L \leftrightarrow vz \in L.$$

Theorem (Myhill-Nerode theorem)

A language L is regular if, and only if, $\equiv_L$ has finitely many equivalence classes. Moreover the number of equivalence classes is equal to the minimum number of states of a DFA required to recognise L

Context free grammars

Generative means of specifying language.

Grammar consists of:

- Non-terminal symbols
- Terminal symbols
- Rules for rewriting non-terminal symbols into strings of non-terminal and terminal symbols
- A starting (non-terminal) symbol

Word w generated by a grammar if a series of rewrite rules, starting from the start symbol, will result in w.

Language of a grammar is the set of words generated by it.

Example (CFGs)

Example

Formal (recursive) definitions:

- Regular expressions
- Propositional formulas
- $\mathcal{L}$ programs (and other languages)