Chapter 0

Course Introduction

Mathematical Modeling on January 12, 2017

Nguyen An Khuong,
Le Hong Trang,
Huynh Tuong Nguyen,
Tran Van Hoai
Faculty of Computer Science and Engineering
University of Technology - VNUHCM

Contents

Contents

Nguyen An Khuong,
Le Hong Trang,
Huynh Tuong Nguyen,
Tran Van Hoai

Aims

- The first part of this course introduces CSE students to the basic concepts of logic (e.g., theories, models, logical consequence, and proof).
- In the second part, students will be learned mathematical modeling through ILP, automata and formal language, Petri net, dynamical systems.
- This is the mathematical foundations for many CS areas, e.g., algorithm analysis & design, database, artificial intelligence, etc.
- Applications of logic in CSE will be highlighted.

Contents

5 chapters, 45 hours of class lectures, HW & exercices.

- Chapter 1. **Propositional Logic Review** and**(Advanced) Predicate Logic**: The need for a richer language; Predicate logic as a formal language; Proof theory of predicate logic; Semantics of predicate logic; Undecidability of predicate logic; Expressiveness of predicate logic.
- Chapter 2. **Mathematical Programming**: Constraints, objectives in ILP.
- Chapter 3. **Automata & Formal Language**: DFA, NFA, Expression, Context.
- Chapter 4. **Petri net**.
- Chapter 5. **Dynamical systems**.

Grading

- 01 Assignment (Project): 20%
- Midterm (MCQ and written; 60 minutes; tentatively after 2 first chapters): 30%
- Final (MCQs + Short Answer Questions, 120 minutes): 50% (**cover all 5 chapter!**)

HW and Attendance

- The course is very intensive and will move fast. It will be very easy to become confused and to fall behind. So **reading materials in advance** and **regular attendance** should be maintained.
- After each lecture, there will be homework problems based on the reading and lecture material. HW will typically be due **6 days** after instructor hand the set out.
- All homework in this class will be **written using the mathematical typesetting program LATEX**, submitted via SAKAI.
- Doing HW is **essential** in order to successfully complete the

Contents

Course outcomes

	Course learning outcomes
L.O.1	Understanding of predicate logic
	L.O.1.1 – Give an example of predicate logic L.O.1.2 – Explain predicate logic for some real problem
L.O.2	Understanding of deterministic modeling using some discrete structures
	L.O.2.1 – Explain a linear programming (mathematical statement) L.O.2.2 – State some well-known discrete structures L.O.2.3 – Give a counter-example for a given wrong automata L.O.2.4 – Construct an automata for a simple problem
L.O.3	Be able to compute solutions, parameters of models based on data
	L.O.3.1 – Compute/Determine optimal/feasible solutions of integer linear programming models, possibly utilizing adequate libraries L.O.3.2 – Compute/ optimize solution models based on automata, . . . , possibly utilizing adequate libraries

Nguyen An Khuong,
Le Hong Trang,
Huynh Tuong Nguyen,
Tran Van Hoai

Assignment Contents

Topics change every year. It may be

- construct correct mathematical reasoning
- design digital circuits
- verify the correctness of computer programs, software verification
- distinguish between valid and invalid mathematical statement
- artificial intelligence

Required Texts/Materials

Electronic copies of [2-6] are available on the WWW, or upon request to instructors.

1. Handouts (Obtained via SAKAI after classes.)
2. Michael R.A. Huth and Mark D. Ryan.*Logic in Computer Science* (2nd Ed.), Cambridge University Press, 2004. (**Chapters 1, 2**)
3. Michael R.A. Huth and Mark D. Ryan. *Logic in Computer Science: Solutions to designated exercises* (2nd Ed.), Cambridge University Press, 2004. (**Chapters 1, 2**)
4. F.R. Giordano, W.P. Fox & S.B. Horton, A First Course in Mathematical Modeling, 5th ed., Cengage, 2014.
5. K. M. Bliss K. R. Fowler B. J. Galluzzo, Math Modeling: getting started & getting solutions. Society for Industrial and Applied Mathematics (SIAM) Handbook, 2014.
6. Matousek et al. Understanding and using linear programming, Springer, 2007.
7. Peter Linz. *An Introduction to Formal Languages and Automata* (3rd Ed.) Jones and Bartlett, 2001. (**Chapters 1-6**)
8. Peter Linz. *An Introduction to Formal Languages and Automata: Instructors' Manual* (3rd Ed.) Jones and Bartlett, 2001. (**Chapters 1-6**)

Instructors

1. Nguyễn An Khương, `nakhuong@hcmut.edu.vn`
2. Lê Hồng Trang, `lhtrang@hcmut.edu.vn`
3. Huỳnh Tường Nguyên, `htnguyen@hcmut.edu.vn`
4. Trần Văn Hoài, `hoai@hcmut.edu.vn`
5. Nguyễn Văn Minh Mẫn, `mannguyen@hcmut.edu.vn`
6. Nguyễn Đức Dũng, `nddung@hcmut.edu.vn`

Tentative Schedule

Notice that the 1st week stars on Monday, January 9, 2017.

Lectures	Topic	Lecturer
01	Ch0. Intro + Demo	NAKhuong
02 - 03	Ch1. Logic	NAKhuong
04 - 06	Ch2. ILP	LHTrang/TVHoai
07	Assignment instruction	NAKhuong
08	Review and Midterm	NAKhuong
09 - 12	Ch3. Automata and Ch4. Petri Net	HTNguyen
13-14	Ch5. Dynamical systems	NAKhuong/NVMMan/NDDung
15	Assignement evaluation	NAKhuong

A Sudoku Grid and Variables

Sudoku

	1	2	3	4	5	6	7	8	9
1	5	3			7				
2	6			1	9	5			
3		9	8					6	
4	8				6				3
5	4			8		3			1
6	7				2				6
7		6					2	8	
8				4	1	9			5
9					8			7	9

Variables

$V = \{X_{ijk} \mid 1 \le i, j, k \le 9\}$

- X_{ijk} true iff cell at row i column j equals k.
- $|V| = 9^3 = 729$
- X_{726} is true
- X_{72k} is false for $k \neq 6$

Constraining exactly one variable to be true

Variables $= \{p, q, r, s\}$

- At least one is true:

$$\alpha = p \vee q \vee r \vee s$$

- No more than one is true:

$$\begin{aligned}\beta \quad = \quad & (\bar{p} \vee \bar{q}) \wedge (\bar{p} \vee \bar{r}) \wedge (\bar{p} \vee \bar{s}) \wedge \\ & (\bar{q} \vee \bar{r}) \wedge (\bar{q} \vee \bar{s}) \wedge (\bar{r} \vee \bar{s})\end{aligned}$$

- Exactly one is true

$$\psi = \alpha \wedge \beta$$

Sudoku row 2 contains exactly one 8

- Row 2 must contain at least one 8

$$\alpha_{2,8} = \bigvee_{1 \le j \le 9} X_{2j8}$$

- Row 2 has at most one 8

$$\beta_{2,8} = \bigwedge_{\substack{1 \le j,m \le 9 \\ j \ne m}} \left(\bar{X}_{2j8} \vee \bar{X}_{2m8} \right)$$

- Row 2 has exactly one 8

$$\psi_{2,8} = \alpha_{2,8} \wedge \beta_{2,8}$$

Sudoku row constraints

- Row 2 contains all 9 values exactly once

$$\gamma_2 = \bigwedge_{1 \le k \le 9} \psi_{2,k}$$

- All 9 rows contain all 9 values exactly once

$$\begin{aligned}
R &= \bigwedge_{1 \le i \le 9} \gamma_i = \bigwedge_{1 \le i \le 9} \bigwedge_{1 \le k \le 9} \psi_{i,k} \\
&= \bigwedge_{1 \le i \le 9} \bigwedge_{1 \le k \le 9} (\alpha_{i,k} \wedge \beta_{i,k}) \\
&= \bigwedge_{1 \le i \le 9} \bigwedge_{1 \le k \le 9} \left[\left(\bigvee_{1 \le j \le 9} X_{ijk} \right) \wedge \left(\bigwedge_{\substack{1 \le j,m \le 9 \\ j \ne m}} (\bar{X}_{ijk} \vee \bar{X}_{imk}) \right) \right]
\end{aligned}$$

Column constraints

- All 9 columns contain all 9 values exactly once

$$C = \bigwedge_{1\leq j\leq 9}\bigwedge_{1\leq k\leq 9}\left[\left(\bigvee_{1\leq i\leq 9} X_{ijk}\right)\wedge\left(\bigwedge_{\substack{1\leq i,m\leq 9\\ i\neq m}}(\bar{X}_{ijk}\vee\bar{X}_{mjk})\right)\right]$$

3×3 box constraints

- 3×3 box containing cell $(4, 7)$ has at least one 5

$$\xi_{475} = \bigvee_{\substack{i=4,5,6\\ j=7,8,9}} X_{ij5}$$

- 3×3 box containing cell $(4, 7)$ has at most one 5

$$\zeta_{475} = \bigwedge_{\substack{i,m=4,5,6\\ j,n=7,8,9\\ i\neq m \vee j\neq n}} (\bar{X}_{ij5} \vee \bar{X}_{mn5})$$

3×3 box constraints, cont...

- 3×3 box containing cell $(4, 7)$ has exactly one 5

$$\theta_{475} = \xi_{475} \wedge \zeta_{475}$$

- All 9 3×3 boxes contains exactly one 5

$$\mu_5 = \bigwedge_{\substack{i=1,4,7 \\ j=1,4,7}} \theta_{ij5}$$

3×3 box constraints, cont...

- All 9 3×3 boxes contain all 9 values

$$B = \bigwedge_{1\leq k\leq 9} \mu_k = \bigwedge_{1\leq k\leq 9} \bigwedge_{\substack{i=1,4,7 \\ j=1,4,7}} \theta_{ijk}$$

Initial predefined values

$$I = X_{115} \wedge X_{123} \wedge \cdots \wedge X_{999}$$

	1	2	3	4	5	6	7	8	9
1	5	3			7				
2	6			1	9	5			
3		9	8					6	
4	8				6				3
5	4			8		3			1
6	7				2				6
7		6					2	8	
8				4	1	9			5
9					8			7	9

Sudoku Boolean Formula

$$\phi = I \wedge R \wedge C \wedge B$$

- Note that ϕ is in CNF, where
  - Conjunctive Normal Form (CNF) if it is the AND of clauses, where a clause is the OR of literals.
  - A literal is a variable or its negation.
  - A clause is an expression formed from a finite collection of literals.
- ϕ can be altered so that it contains exactly 3 literals per clause (can be fed to 3-SAT solver): See Chapter 1b.

Bit strings expression

Consider the finite set of binary strings
$\{(000000), (100000), (110000), (111000), (111100), (111110), (111111), (011111), (001111), (000111), (000011), (000001)\}$
Represent such a set in a propositional formula and simplify that representation?

Solution:

- For each $0 \leq i \leq 5$, b_i is a proposition, which intuitively means that the i-th bit has value 1.
- Obviously, $\neg b_i$ means that the i-th bit does not have value 1, and thus it has value 0.
- A possible (compact) representation of the finite set of binary strings is given by the following formula:

$$\bigvee_{i=0}^{k}\left(\left(\bigwedge_{i=0}^{k}\neg b_i \wedge \bigwedge_{i=k+1}^{5} b_i\right) \bigvee \left(\bigwedge_{i=0}^{k} b_i \wedge \bigwedge_{i=k+1}^{k} \neg b_i\right)\right)$$

Chapter 1a
Propositional Logic Review I

Mathematical Modeling (CO2011)

(Materials drawn from **Chapter 1** in:
"Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*, 2nd Ed., Cambridge University Press, 2006.")

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê H`ng Trang
Faculty of Computer Science and Engineering
University of Technology, VNU-HCM

Contents

1 Propositional Calculus: Declarative Sentences

2 Propositional Calculus: Natural Deduction

3 Propositional Logic as a Formal Language

4 Semantics of Propositional Logic

5 Conjunctive Normal Form

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Propositional Calculus

Study of atomic propositions

Propositions are built from sentences whose internal structure is not of concern.

Building propositions

Boolean operators are used to construct propositions out of simpler propositions.

Example for Propositional Calculus

- **Atomic proposition:** One plus one equals two.
- **Atomic proposition:** The earth revolves around the sun.
- **Combined proposition:** One plus one equals two *and* the earth revolves around the sun.

Goals and Main Result of Propositional Calculus

Meaning of formula

Associate meaning to a set of formulas by assigning a value *true* or *false* to every formula in the set.

Proofs

Symbol sequence that formally establishes whether a formula is always true.

Soundness and completeness

The set of provable formulas is the same as the set of formulas which are always true.

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Uses of Propositional Calculus

Hardware design

The production of logic circuits uses propositional calculus at all phases; specification, design, testing.

Verification

Verification of hardware and software makes extensive use of propositional calculus.

Problem solving

Decision problems (scheduling, timetabling, etc) can be expressed as satisfiability problems in propositional calculus.

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Predicate Calculus: Central ideas

Richer language

Instead of dealing with atomic propositions, predicate calculus provides the formulation of statements involving sets, functions and relations on these sets.

Quantifiers

Predicate calculus provides statements that all or some elements of a set have specified properties.

Compositionality

Similar to propositional calculus, formulas can be built from composites using logical connectives.

The uses of Predicate Calculus

Progamming Language Semantics

The meaning of programs such as

$$\texttt{ifx} >= \texttt{0theny} := \texttt{sqrt}(\texttt{x})\texttt{elsey} := \texttt{abs}(\texttt{x})$$

can be captured with formulas of predicate calculus:

$$\forall x \forall y (x' = x \land (x \geq 0 \to y' = \sqrt{x}) \land (\neg(x \geq 0) \to y' = |x|))$$

Other Uses of Predicate Calculus

- **Specification:** Formally specify the purpose of a program in order to serve as input for software design,
- **Verification:** Prove the correctness of a program with respect to its specification.

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

An Example for Specification

Let P be a program of the form

```
while a <> b do
   if a > b then a := a - b else a:= b - a;
```

The specification of the program is given by the formula

$$\{a \geq 0 \wedge b \geq 0\}\ P\ \{a = \mathit{gcd}(a, b)\}$$

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Logic in Theorem Proving, Logic Programming, and Other Systems of Logic

Theorem proving

Formal logic has been used to design programs that can automatically prove mathematical theorems.

Logic programming

Research in theorem proving has led to an efficient way of proving formulas in predicate calculus, called *resolution*, which forms the basis for *logic programming*.

Some Other Systems of Logic

- **Three-valued logic:** A third truth value (denoting "don't know" or "undetermined") is often useful.
- **Intuitionistic logic:** A mathematical object is accepted only if a finite construction can be given for it.
- **Temporal logic:** Integrates time-dependent constructs such as ("always" and "eventually") explicitly into a logic framework; useful for reasoning about real-time systems.

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

1. **Propositional Calculus: Declarative Sentences**
2. Propositional Calculus: Natural Deduction
3. Propositional Logic as a Formal Language
4. Semantics of Propositional Logic
5. Conjunctive Normal Form

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Declarative Sentences

The language of propositional logic is based on *propositions* or *declarative sentences*.

Declarative Sentences

Sentences which one can—in principle—argue as being true or false.

Examples

1. The sum of the numbers 3 and 5 equals 8.
2. Jane reacted violently to Jack's accusations.
3. Every natural number > 2 is the sum of two prime numbers.
4. All Martians like pepperoni on their pizza.

Not Examples

- Could you please pass me the salt?
- Ready, steady, go!
- May fortune come your way.

Putting Propositions Together

Example 1.1

If *the train arrives late* and
there are no taxis at the station then
John is late for his meeting.

John is not late for his meeting.

The train did arrive late.

Therefore, *there were taxis at the station.*

Example 1.2

If *it is raining* and
Jane does not have her umbrella with her then
she will get wet.

Jane is not wet.

It is raining.

Therefore, *Jane has her umbrella with her.*

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Focus on Structure

We are primarily concerned about the structure of arguments in this class, not the validity of statements in a particular domain.

We therefore simply abbreviate sentences by letters such as p, q, r, p_1, p_2 etc.

From Concrete Propositions to Letters - Example 1.1

If *the train arrives late* and
there are no taxis at the station then
John is late for his meeting.

John is not late for his meeting.

The train did arrive late.

Therefore, *there were taxis at the station.*

becomes

Letter version

If p and not q, then r. Not r. p. Therefore, q.

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Focus on Structure

From Concrete Propositions to Letters - Example 1.2

If *it is raining* and
Jane does not have her umbrella with her then
she will get wet.

Jane is not wet.

It is raining.

Therefore, *Jane has her umbrella with her.*

has

the same letter version

If p and not q, then r. Not r. p. Therefore, q.

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Logical Connectives

Notations/Symbols

Sentences like "**If** p **and not** q, **then** r." occur frequently. Instead of English words such as "**if...then**", "**and**", "**not**", it is more convenient to use symbols such as $\rightarrow$, $\wedge$, $\neg$.

- $\neg$: negation of p is denoted by $\neg p$.
- $\vee$: disjunction of p and r is denoted by $p \vee r$, meaning at least one of the two statements is true.
- $\wedge$: conjunction of p and r is denoted by $p \wedge r$, meaning both are true.
- $\rightarrow$: implication between p and r is denoted by $p \rightarrow r$, meaning that r is a logical consequence of p. p is called the *antecedent*, and r the *consequent*.

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Example 1.1 Revisited

From Example 1.1

If *the train arrives late* and
there are no taxis at the station then
John is late for his meeting.

Symbolic Propositions

We replaced "*the train arrives late*" by p, etc.

The statement becomes: If p and not q, then r.

Symbolic Connectives

With symbolic connectives, the statement becomes:

$p \land \neg q \to r$

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

1 Propositional Calculus: Declarative Sentences

2 Propositional Calculus: Natural Deduction
- Sequents
- Rules for natural deduction
- Basic and Derived Rules
- Excursion: Intuitionistic Logic

3 Propositional Logic as a Formal Language

4 Semantics of Propositional Logic

5 Conjunctive Normal Form

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Introduction

Objective

We would like to develop a *calculus* for reasoning about propositions, so that we can establish the validity of statements such as Example 1.1.

Idea

We introduce *proof rules* that allow us to derive a formula ψ from a number of other formulas $\phi_1, \phi_2, \ldots \phi_n$.

Notation

We write a *sequent* $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$
to denote that we can derive ψ from $\phi_1, \phi_2, \ldots, \phi_n$.

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Example 1.1 Revisited

English

If *the train arrives late* and
there are no taxis at the station then
John is late for his meeting.

John is not late for his meeting.

The train did arrive late.

Therefore, *there were taxis at the station.*

Sequent

$$p \wedge \neg q \rightarrow r, \neg r, p \vdash q$$

Remaining task

Develop a set of proof rules that allows us to establish such sequents.

Rules for Conjunction

Introduction of Conjunction

$$\frac{\phi \qquad \psi}{\phi \wedge \psi}[\wedge i]$$

Elimination of Conjunction

$$\frac{\phi \wedge \psi}{\phi}[\wedge e_1] \qquad \frac{\phi \wedge \psi}{\psi}[\wedge e_2]$$

Example of Proof

To show

$$p \wedge q, r \vdash q \wedge r.$$

How to start?

$$p \wedge q \qquad\qquad r,$$

$$q \wedge r.$$

Proof Step-by-Step

1. $p \wedge q$ (premise)
2. r (premise)
3. q (by using Rule $\wedge e_2$ and Item 1)
4. $q \wedge r$ (by using Rule $\wedge i$ and Items 3 and 2)

Graphical Representation of Proof

$$\dfrac{\dfrac{p \wedge q}{q}\,[\wedge e_2] \qquad r}{q \wedge r}\,[\wedge i]$$

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Where are we heading with this?

- We would like to prove sequents of the form $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$
- We introduce rules that allow us to form "legal" proofs
- Then any proof of any formula ψ using the premises $\phi_1, \phi_2, \ldots, \phi_n$ is considered "correct".
- Can we say that sequents with a correct proof are somehow "valid", or "meaningful"?
- What does it mean to be meaningful?
- Can we say that any meaningful sequent has a valid proof?
- ...but first back to the proof rules...

Rules of Double Negation and Eliminating Implication

Double Negation

$$\frac{\neg\neg\phi}{\phi}[\neg\neg e] \qquad \frac{\phi}{\neg\neg\phi}[\neg\neg i]$$

Eliminating Implication

$$\frac{\phi \qquad \phi \to \psi}{\psi}[\to e]$$

Example

p:= "It rained," and $p \to q$:= "If it rained, then the street is wet." We can conclude from these two that the street is indeed wet.

Modus Ponens

The rule

$$\frac{\phi \qquad \phi \to \psi}{\psi} [\to e]$$

is often called "Modus Ponens" (or MP)

Origin of term

"Modus ponens" is an abbreviation of the Latin "modus ponendo ponens" which means in English "mode that affirms by affirming". More precisely, we could say "mode that affirms the antecedent of an implication".

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Modus Tollens

A similar rule of "Modus Ponens",

$$\frac{\phi \to \psi \qquad \neg\psi}{\neg\phi}[MT]$$

is called "Modus Tollens" (or MT).

Origin of term

"Modus tollens" is an abbreviation of the Latin "modus tollendo tollens" which means in English "mode that denies by denying". More precisely, we could say "mode that denies the consequent of an implication".

Example

$$p \to (q \to r), p, \neg r \vdash \neg q$$

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

How to introduce implication?

Compare the sequent (MT)

$$p \to q, \neg q \vdash \neg p$$

with the sequent

$$p \to q \vdash \neg q \to \neg p$$

The second sequent should be provable, but we don't have a rule to introduce implication yet!

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

A Proof We Would Like To Have

$$p \to q \vdash \neg q \to \neg p$$

1	$p \to q$	premise
2	$\neg q$	assumption
3	$\neg p$	MT 1,2
4	$\neg q \to \neg p$	$\to_i$ 2–3

We can start a box with an *assumption*, and use previously proven propositions (including premises) from the outside in the box. We cannot use assumptions from inside the box in rules outside the box.

Rule for Introduction of Implication

Introduction of Implication

$$\dfrac{\boxed{\begin{array}{c}\phi \\ \vdots \\ \psi\end{array}}}{\phi \to \psi}\,[\to i]$$

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Rule for Disjunction

Introduction of Disjunction

$$\frac{\phi}{\phi \vee \psi}\,[\vee i_1] \qquad \frac{\psi}{\phi \vee \psi}\,[\vee i_2]$$

Elimination of Disjunction

$$\frac{\phi \vee \psi \qquad \boxed{\begin{array}{c}\phi\\ \vdots\\ \chi\end{array}} \qquad \boxed{\begin{array}{c}\psi\\ \vdots\\ \chi\end{array}}}{\chi}\,[\vee e]$$

Example

1	$p \wedge (q \vee r)$	premise
2	p	$\wedge e_1$ 1
3	$q \vee r$	$\wedge e_2$ 1
4	q	assumption
5	$p \wedge q$	$\wedge i$ 2,4
6	$(p \wedge q) \vee (p \wedge r)$	$\vee i_1$ 5
7	r	assumption
8	$p \wedge r$	$\wedge i$ 2,7
9	$(p \wedge q) \vee (p \wedge r)$	$\vee i_2$ 8
10	$(p \wedge q) \vee (p \wedge r)$	$\vee e$ 3, 4–6, 7–9

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Special Propositions

- Recall: We are only interested in the truth value of propositions, not the subject matter that they refer to.
- Therefore, all propositions that we all agree must be true are the same!
- Example: $p \to p$, $p \lor \neg p$
- We denote the proposition that is always true (**tautology**) using the symbol $\top$.

Another Special Proposition

- Similarly, we denote the proposition that is always false (**contradiction**) using the symbol $\bot$.
- Example: $p \land \neg p$

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Rule for Negation

Elimination of Negation

$$\frac{\phi \qquad \neg\phi}{\bot}\,[\neg e]$$

Introduction of Negation

$$\frac{\boxed{\begin{array}{c}\phi\\ \vdots \\ \bot\end{array}}}{\neg\phi}\,[\neg i]$$

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Elimination of $\perp$

Elimination of $\perp$

$$\frac{\perp}{\phi}\,[\perp e]$$

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Basic Rules (conjunction and disjunction)

$$\frac{\phi \qquad \psi}{\phi \wedge \psi}\,[\wedge i] \qquad \frac{\phi \wedge \psi}{\phi}\,[\wedge e_1] \qquad \frac{\phi \wedge \psi}{\psi}\,[\wedge e_2]$$

$$\frac{\phi}{\phi \vee \psi}\,[\vee i_i] \qquad \frac{\psi}{\phi \vee \psi}\,[\vee i_2] \qquad \frac{\phi \vee \psi \qquad \boxed{\begin{matrix}\phi \\ \vdots \\ \chi\end{matrix}} \qquad \boxed{\begin{matrix}\psi \\ \vdots \\ \chi\end{matrix}}}{\chi}\,[\vee e]$$

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Basic Rules (implication)

$$\frac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \psi\end{array}}}{\phi \rightarrow \psi}[\rightarrow i] \qquad \frac{\phi \qquad \phi \rightarrow \psi}{\psi}[\rightarrow e]$$

Basic Rules (negation)

$$\dfrac{\begin{array}{c}\phi \\ \vdots \\ \bot\end{array}}{\neg\phi}\,[\neg i] \qquad \dfrac{\phi \qquad \neg\phi}{\bot}\,[\neg e]$$

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Basic Rules ($\perp$ and double negation)

$$\frac{\perp}{\phi}\,[\perp e]$$

$$\frac{\neg\neg\phi}{\phi}\,[\neg\neg e]$$

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Some Derived Rules: Introduction of Double Negation

$$\frac{\phi}{\neg\neg\phi}\,[\neg\neg i]$$

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Example: Deriving [¬¬ i] from [¬ i] and [¬ e]

1	ϕ	premise
2	$\neg\phi$	assumption
3	$\bot$	$\neg$e 1,2
4	$\neg\neg\phi$	$\neg$i 2–3

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Some Derived Rules: Modus Tollens

$$\frac{\phi \to \psi \qquad \neg\psi}{\neg\phi}[MT]$$

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Some Derived Rules: Proof By Contradiction

$$\dfrac{\boxed{\begin{array}{c}\neg\phi\\ \vdots\\ \bot\end{array}}}{\phi}\,[\mathrm{PBC}]$$

Some Derived Rules: Law of Excluded Middle

$$\frac{}{\phi \vee \neg\phi}\,[\mathrm{LEM}]$$

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Motivation

Consider the following theorem.

Theorem

There exist irrational numbers a and b such that a^b is rational.

Let us call this theorem χ. We give a Proof Outline for χ.
Let p be the following proposition.

Proposition p

$\sqrt{2}^{\sqrt{2}}$ is rational.

Then the proof of χ goes like this:

$$\dfrac{\dfrac{}{p \vee \neg p}[LEM] \qquad \boxed{\begin{array}{c} p \\ \vdots \\ \chi \end{array}} \qquad \boxed{\begin{array}{c} \neg p \\ \vdots \\ \chi \end{array}}}{\chi}[\vee e]$$

In detail (1)

$$p$$
$$\vdots$$
$$\chi$$

Assume $\sqrt{2}^{\sqrt{2}}$ is rational. Choose a and b to be $\sqrt{2}$, and we have found irrational a and b such that a^b is rational. Thus Theorem χ holds under the assumption p.

In detail (2)

$$\begin{array}{c} \neg p \\ \vdots \\ \chi \end{array}$$

Assume $\sqrt{2}^{\sqrt{2}}$ is irrational. Choose a to be $\sqrt{2}^{\sqrt{2}}$ and b to be $\sqrt{2}$. Then we have

$$a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{(\sqrt{2}\cdot\sqrt{2})} = (\sqrt{2})^2 = 2.$$

As 2 is rational, Theorem χ holds under the assumption $\neg p$.

Summary of Proof for χ

Proposition p

$\sqrt{2}^{\sqrt{2}}$ is rational.

$$\cfrac{\cfrac{}{p \vee \neg p}[LEM] \qquad \boxed{\begin{array}{c} p \\ \vdots \\ \chi \end{array}} \qquad \boxed{\begin{array}{c} \neg p \\ \vdots \\ \chi \end{array}}}{\chi}[\vee e]$$

There exist irrational numbers a and b such that a^b is rational...

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

The Magic of LEM

- There exist irrational numbers a and b such that a^b is rational.
- But: If they exist, do you have an example?
- Probably $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$..., but we haven't proven that $\sqrt{2}^{\sqrt{2}}$ is irrational!
- Note: $\sqrt{2}^{\sqrt{2}^{\sqrt{2}^{\cdot^{\cdot}}}} = 2$
- Using LEM, we can make use of the "probable irrationality" of $\sqrt{2}^{\sqrt{2}}$ without having to prove it!

Intuitionistic Logic

Intuitionistic logic does not accept the derived rule LEM.
The underlying argument for LEM is elimination of double negation.

$$\frac{\neg\neg\phi}{\phi}\,[\neg\neg e]$$

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Deriving LEM using Basic Rules

1	$\neg(\phi \vee \neg\phi)$	assumption
2	ϕ	assumption
3	$\phi \vee \neg\phi$	$\vee i_1$ 2
4	$\bot$	$\neg$ e 3,1
5	$\neg\phi$	$\neg$ i 2–4
6	$\phi \vee \neg\phi$	$\vee$ i_2 5
7	$\bot$	$\neg$ e 6,1
8	$\neg\neg(\phi \vee \neg\phi)$	$\neg$i 1–7
9	$\phi \vee \neg\phi$	$\neg\neg$e

Intuitionistic Logic

Intuitionistic logic is obtained from natural deduction by removing the rule $\neg\neg e$.

History of Intuitionistic Logic

- Late 19th century: Gottlob Frege proposes to reduce mathematics to set theory.
- Russell destroys this programme via paradox.
- In response, L.E.J. Brouwer proposes *intuitionistic* mathematics, with *intuitionistic logic* as its formal foundation.
- An alternative response is Hilbert's *formalistic* position.

Applications of Intuitionistic Logic

- Intuitionistic logic has a strong connection to *computability*
- For example, if we have an intuitionistic proof of

Theorem

There exist irrational numbers a and b such that a^b is rational.

then we would know irrational a and b such that a^b is rational.

1 Propositional Calculus: Declarative Sentences

2 Propositional Calculus: Natural Deduction

3 **Propositional Logic as a Formal Language**

4 Semantics of Propositional Logic

5 Conjunctive Normal Form

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Recap: Logical Connectives

- $\neg$: negation of p is denoted by $\neg p$.
- $\vee$: disjunction of p and r is denoted by $p \vee r$, meaning at least one of the two statements is true.
- $\wedge$: conjunction of p and r is denoted by $p \wedge r$, meaning both are true.
- $\rightarrow$: implication between p and r is denoted by $p \rightarrow r$, meaning that r is a logical consequence of p.

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Formal itemize Required

Use of Meta-Language

When we describe rules such as $\dfrac{}{\phi \vee \neg\phi}[LEM]$

we mean that letters such as ϕ can be replaced by *any* formula.

But what exactly is the set of formulas that can be used for ϕ?

Allowed

$(p \wedge (\neg q))$

Not allowed

$) \wedge p \quad q\neg($

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Definition of Well-formed Formulas

Definition

- Every propositional atom $p, q, r, \ldots$ and $p_1, p_2, p_3, \ldots$ is a well-formed formula.
- If ϕ is a well-formed formula, then so is $(\neg\phi)$.
- If ϕ and ψ are well-formed formulas, then so is $(\phi \land \psi)$.
- If ϕ and ψ are well-formed formulas, then so is $(\phi \lor \psi)$.
- If ϕ and ψ are well-formed formulas, then so is $(\phi \to \psi)$.

Definition very restrictive

How about this formula?

$$p \land \neg q \lor r$$

Usually, this is understood to mean

$$((p \land (\neg q)) \lor r)$$

...but for the formal treatment of this section and the first homework, we insist on the strict definition, and exclude such formulas.

Backus Naur Form: A more compact definition

Backus Naur Form for propositional formulas

$$\phi ::= p|(\neg\phi)|(\phi \wedge \phi)|(\phi \vee \phi)|(\phi \rightarrow \phi)$$

where p stands for any atomic proposition.

Inversion principle

How can we show that a a formula such as

$$(((\neg p) \wedge q) \to (p \wedge (q \vee (\neg r))))$$

is well-formed?

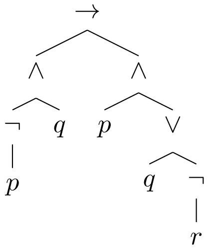
<u>**Answer:**</u> We look for the only applicable rule in the definition (the last rule in this case), and proceed on the parts.

Parse trees

A formula

$$(((\neg p) \wedge q) \to (p \wedge (q \vee (\neg r))))$$

...and its parse tree:



1 Propositional Calculus: Declarative Sentences

2 Propositional Calculus: Natural Deduction

3 Propositional Logic as a Formal Language

4 **Semantics of Propositional Logic**
Meaning of Logical Connectives
Preview: Soundness and Completeness

5 Conjunctive Normal Form

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Meaning of propositional formula

Meaning as mathematical object

We define the meaning of formulas as a function that maps formulas and valuations to truth values.

Approach

We define this mapping based on the structure of the formula, using the meaning of their logical connectives.

Truth Values

The set of truth values contains two elements $\mathtt{T}$ and $\mathtt{F}$, where $\mathtt{T}$ represents "**true**" and $\mathtt{F}$ represents "**false**".

Valuations

A *valuation* or *model* of a formula ϕ is an assignment of each propositional atom in ϕ to a truth value.

Meaning of logical connectives

The meaning of a connective is defined as a truth table that gives the truth value of a formula, whose root symbol is the connective, based on the truth values of its components.

ϕ	ψ	$\phi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

Truth tables of formulas

Truth tables use placeholders of formulas such as ϕ:

ϕ	ψ	$\phi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

Build the truth table for given formula:

p	q	r	$(p \wedge q)$	$((p \wedge q) \wedge r)$
T	T	T	T	T
T	T	F	T	F
$\vdots$				

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Truth tables of other connectives

ϕ	ψ	$\phi \vee \psi$
T	T	T
T	F	T
F	T	T
F	F	F

ϕ	ψ	$\phi \rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

ϕ	$\neg\phi$
T	F
F	T

$\top$
T

$\bot$
F

Constructing the truth table of a formula

p	q	$(\neg p)$	$\neg q$	$p \to \neg q$	$q \vee \neg p$	$(p \to \neg q) \to (q \vee \neg p)$
T	T	F	F	F	T	T
T	F	F	T	T	F	F
F	T	T	F	T	T	T
F	F	T	T	T	T	T

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Validity and Satisfiability

Validity

A formula is *valid* if it computes `T` for all its valuations.

Satisfiability

A formula is *satisfiable* if it computes `T` for at least one of its valuations.

Semantic Entailment, Soundness and Completeness of Propositional Logic

Semantic Entailment

If, for all valuations in which all $\phi_1, \phi_2, \ldots, \phi_n$ evaluate to $\mathtt{T}$, the formula ψ evaluates to $\mathtt{T}$ as well, we say that $\phi_1, \phi_2, \ldots, \phi_n$ **semantically entail** ψ, written:

$$\phi_1, \phi_2, \ldots, \phi_n \models \psi$$

Soundness

Let $\phi_1, \phi_2, \ldots, \phi_n$ and ψ be propositional formulas. If $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ valid (has a proof), then $\phi_1, \phi_2, \ldots, \phi_n \models \psi$.

Completeness

Let $\phi_1, \phi_2, \ldots, \phi_n$ and ψ be propositional formulas. If $\phi_1, \phi_2, \ldots, \phi_n \models \psi$, then $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ valid (has a proof).

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

1. Propositional Calculus: Declarative Sentences
2. Propositional Calculus: Natural Deduction
3. Propositional Logic as a Formal Language
4. Semantics of Propositional Logic
5. **Conjunctive Normal Form**

Conjunctive Normal Form

Definition

A literal L is either an atom p or the negation of an atom $\neg p$. A formula C is in *conjunctive normal form* (CNF) if it is a conjunction of clauses, where each clause is a disjunction of literals:

$$\begin{array}{rcl} L & ::= & p|\neg p, \\ D & ::= & L|L \vee D, \\ C & ::= & D|D \wedge C. \end{array}$$

Examples

- $(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$ is in CNF.
- $(\neg p \vee q \vee r) \wedge ((p \wedge \neg q) \vee r) \wedge (\neg r)$ is not in CNF.
- $(\neg p \vee q \vee r) \wedge \neg(\neg q \vee r) \wedge (\neg r)$ is not in CNF.

Usefulness of CNF

Lemma

A disjunction of literals $L_1 \vee L_2 \vee \cdots \vee L_m$ is valid iff there are $1 \leq i, j \leq m$ such that L_i is $\neg L_j$.

How to disprove

$$\models (\neg q \vee p \vee r) \wedge (\neg p \vee r) \wedge q?$$

Disprove any of:

$$\models (\neg q \vee p \vee r) \qquad \models (\neg p \vee r) \qquad \models q.$$

How to prove

$$\models (\neg q \vee p \vee q) \wedge (p \vee r \neg p) \wedge (r \vee \neg r)?$$

Prove all of:

$$\models (\neg q \vee p \vee q) \qquad \models (p \vee r \neg p) \qquad \models (r \vee \neg r).$$

Nguyen An Khuong,
Tran Van Hoai,
Huynh Tuong Nguyen,
Lê Hồng Trang

Usefulness of CNF (cont.) and Transformation to CNF

Proposition

Let ϕ be a formula of propositional logic. Then ϕ is satisfiable iff $\neg\phi$ is not valid.

Satisfiability test

We can test satisfiability of ϕ by transforming $\neg\phi$ into CNF, and show that some clause is not valid.

Theorem-Transformation to CNF

Every formula in the propositional calculus can be transformed into an equivalent formula in CNF.

Nguyen An Khuong, Tran Van Hoai, Huynh Tuong Nguyen, Lê Hồng Trang

Algorithm for CNF Transformation

1. Eliminate implication using:
   $A \to B \equiv \neg A \lor B$.
2. Push all negations inward using De Morgan's laws:

$$\neg(A \land B) \equiv (\neg A \lor \neg B),$$

$$\neg(A \lor B) \equiv (\neg A \land \neg B).$$

3. Eliminate double negations using the equivalence $\neg\neg A \equiv A$.
4. The formula now consists of disjunctions and conjunctions of literals. Use the distributive laws to eliminate conjunctions within disjunctions:

$$A \lor (B \land C) \equiv (A \lor B) \land (A \lor C),$$

$$(A \land B) \lor C \equiv (A \lor C) \land (B \lor C).$$

Example

$$
\begin{aligned}
(\neg p \rightarrow \neg q) \rightarrow (p \rightarrow q) &\equiv \neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q) \\
&\equiv (\neg\neg\neg p \wedge q) \vee (\neg p \vee q) \\
&\equiv (\neg p \wedge q) \vee (\neg p \vee q) \\
&\equiv (\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q) \\
&\equiv \top.
\end{aligned}
$$

Homeworks

I. Write down the explanations (in Vietnamese, or in English if possible) of the following terms, find examples for each term, what are the differences between them:

1) fallacy, contradiction, paradox, counterexample;
2) premise, assumption, axiom, hypothesis, conjecture;
3) tautology, valid, contradiction, satisfiable;
4) soundness, completeness;
5) sequent, consequence, implication, (semantic) entailment;
6) argument, variable, arity;

II. What are the differences between the following notations: '$\longrightarrow$', '$\Longrightarrow$', '$\vdash$', '$\models$'? And what are the differences between the following notations: '$\longleftrightarrow$', '$\Longleftrightarrow$', '$\dashv\vdash$', '$\equiv$', '$=$'? Find examples to illustrate these differences.

III. It is recommended that you should do as much as you can ALL marked exercises in [2] (notice that sample solutions for these exercises are available in [3]). For this lecture, the following are recommended exercises [2]:
1.1: 2d), 2g);
1.2: 1d), 1g), 1m), 1q), 1u), 1w), 3a), 3b), 3c), 3f), 3g), 3l), 3o);
1.4: 12d);
1.5: 3b), 3c), 7c).

Next Week?

- Exercises Session;
- [2, Section 1.6]: SAT Solvers;
- Application of SAT Solving.

Chapter 1b

Propositional Logic Review II

(SAT Solving and Application)

Mathematics Modeling

(Materials drawn from **Chapter 1** in:
"Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*, 2nd Ed., Cambridge University Press, 2006"
and some other sources)

Nguyen An Khuong,
Le Hong Trang,
Huynh Tuong Nguyen,
Tran Van Hoai
Faculty of Computer Science and Engineering
University of Technology, VNU-HCM

Contents

Contents

❶ Introduction

Quick review

Boolean Satisfiability (SAT)

Intermezzo: Classification of problems according to their difficulty

❷ 2-SAT is in P

❸ SAT Solvers

Motivated Example – A Logic Puzzle

- If the unicorn is mythical, then it is immortal;and
- If the unicorn is not mythical, then it is a mortal mammal;and
- If the unicorn is either immortal or a mammal, then it is horned;and
- The unicorn is magical if it is horned.

- **Q:** Is the unicorn mythical? Is it magical? Is it horned?

Nguyen An Khuong,
Le Hong Trang,
Huynh Tuong Nguyen,
Tran Van Hoai

CNF

- Boolean formula ϕ is defined over a set of propositional variables $p_1, ..., p_n$, using the standard propositional connectives $\neg, \wedge, \vee, \longrightarrow, \longleftrightarrow$, and parenthesis
  - The domain of propositional variables is $\{0, 1\}$.
  - Example: $\phi(p_1, p_2, p_3) = ((\neg p_1 \wedge p_2) \vee p_3) \wedge (\neg p_2 \vee p_3)$.
- A formula ϕ in conjunctive normal form (CNF) is a conjunction of disjunctions (clauses) of literals, where a literal is a variable or its complement.
  - Example: $\phi(p_1, p_2, p_3) = (\neg p_1 \vee p_2) \wedge (\neg p_2 \vee p_3)$.

Proposition (see [2, Subsection 1.5.1])

There is an algorithm to translate *any* Boolean formula into CNF.

Proposition 1.45, p. 57

ϕ-satisfiable iff $\neg\phi$-not tautology.

SAT

Problem

Find an assignment to the variables $p_1, ..., p_n$ such that $\phi(p_1, ..., p_n) = 1$, or prove that no such assignment exists.

Facts: SAT is an NP-complete decision problem [Cook'71]

- SAT was the first problem to be shown NP-complete.
- There are no known polynomial time algorithms for SAT.
- More-than-35-year old conjecture:
  "Any algorithm that solves SAT is exponential in the number of variables, in the worst-case."

Nguyen An Khuong, Le Hong Trang, Huynh Tuong Nguyen, Tran Van Hoai

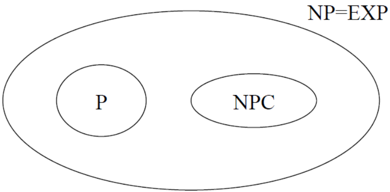
Polynomial time reductions and NP-Completeness

- Denote
  - EXP = {Decision problems solvable in exponential time}
  - P = {Decision problems solvable in polynomial time}
  - NP = {Decision problems where Yes solution can verified in polynomial time}
- A major open question in theoretical computer science is **if** $P = NP$ **or not**.
- Introduce the notion of **polynomial time reductions** $X \leq_P Y$:

  A problem X is polynomial time reducible to a problem Y ($X \leq_P Y$) if we can solve X in a polynomial number of calls to an algorithm for Y (and the instance of problem Y we solve can be computed in polynomial time from the instance of problem X).
- The class of NP-**complete** problems NPC: A problem Y is in NPC if
  a) $Y \in NP$, and
  b) $X \leq_P Y$ for all $X \in NP$.

P=NP question

- The problems in NPC are the hardest problems in NP and the key to resolving the $P = NP$ question.
- If one problem $Y \in NPC$ is in P then $P = NP$.
- If one problem $Y \in NP$ is not in P then $NPC \cap P = \emptyset$.
- By now a lot of problems have been proved NP-complete
- We think the world looks like this—but we really do not know:



- If someone found a polynomial time solution to a problem in NPC our world would "collapse" *and* a lot of smart people have tried really hard to solve NPC problems efficiently

$$\Downarrow$$

We regard $Y \in NPC$ a strong evidence for Y being hard!

NP-Complete Problems

- The following lemma helps us to prove a problem NP-complete using another NP-complete problem.

  Lemma: If $Y \in NP$ and $X \leq_P Y$ for some $X \in NPC$ then $Y \in NPC$

  Proof: To prove $Y \in NPC$ we just need to prove $Y \in NP$ (often easy) and reduce problem in NPC to Y (no lower bound proof needed!).
- Finding the first problem in NPC is somewhat difficult and require quite a lot of formalism
- It seems to be a easier problem 3Sat: Given a formula in 3-CNF, is it satisfiable?
  - A formula is in 3-CNF (conjunctive normal form) if it consists of an And of 'clauses' each of which is the Or of 3 'literals'
  - Example: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$
- We prove that 3SAT is in NPC, meaning that it is as hard as general SAT.
  - 3SAT $\in NP$
  - SAT $\leq_P$ 3SAT (we can show that transforming general formula into 3-CNF is in polynomial time.)

Example

- Consider the following 2-CNF formula consisting of the following clauses:

$$\bar{x}_1 \vee x_2, \qquad x_1 \vee x_2, \qquad \bar{x}_2 \vee x_3, \qquad x_3 \vee \bar{x}_4, \qquad x_1 \vee \bar{x}_2.$$

- Let's try to set $x_1 = 0$. Then the formula simplifies to:

$$T, \qquad x_2, \qquad \bar{x}_2 \vee x_3, \qquad x_3 \vee \bar{x}_4, \qquad \bar{x}_2.$$

  where T denotes the value "Truth".
- We are now *forced* to assign $x_2 = 1$ (as there is a unit-clause), and the formula simplifies to

$$T, \qquad T, \qquad x_3, \qquad x_3 \vee \bar{x}_4, \qquad \emptyset,$$

  where $\emptyset$ is the empty clause which denotes contradiction.
- So we have to backtrack to the last *free step*.
- Let's try $x_1 = 1$:

$$x_2, \qquad T, \qquad \bar{x}_2 \vee x_3, \qquad x_3 \vee \bar{x}_4, \qquad T.$$

- We are now forced to set $x_2 = 1$:

$$T, \qquad T, \qquad x_3, \qquad x_3 \vee \bar{x}_4, \qquad T.$$

- We are now forced to set $x_3 = 1$:

Algorithm(ϕ)

Abstracting the above example, we present an algorithm that attempts to satisfy a 2-CNF formula ϕ as follows.

Algorithm(ϕ)

(0) Initialize empty assignment $\sigma = *^n$.

(1) If all variables are assigned return σ.

(2) Choose an unassigned variable x_i.

  (a) (Try $x_i = 1$)
    - Set $\sigma_i = 1$, $\phi' \leftarrow$ Simplify(ϕ, x_i).
    - $\phi' \leftarrow$ Unit Clause Propagation(ϕ').
    - If ϕ' does not contain $\emptyset$ goto (1).

  (b) (Try $x_i = 0$)
    - Unassign variables from step (a).
    - Set $\sigma_i = 0$, $\phi' \leftarrow$ Simplify($\phi, \bar{x}_i$).
    - $\phi' \leftarrow$ Unit Clause Propagation(ϕ').
    - If ϕ' does not contain $\emptyset$ goto (1).

(3) Halt with "UNSAT".

Simplify(ϕ, ℓ_i)

Simplify(ϕ, ℓ_i)

- $\forall$ clause $C \in \phi$:
  - If $\ell_i \in C$, remove C.
  - If $\bar{\ell}_i \in C$, $C \leftarrow C \setminus \bar{\ell}_i$.
  - Otherwise, copy C as is.
- Output the modified formula.

Unit Clause Propagation(ϕ)

- While $\exists$ unit clause ℓ_i:
  - Update σ: if $\ell_i = x_i$ set $\sigma_i = 1$, else ($\ell_i = \bar{x}_i$) set $\sigma_i = 0$.
  - $\phi \leftarrow$Simplify(ϕ, ℓ_i).

Complexity: Let n denote the number of variables and let m denote the number of clauses. It is not hard to verify that there are at most n outer iterations and that each call to UCP takes at most $O(m)$ time, therefore the running time of Algorithm is $O(m \cdot n)$. (HW: Find an implementation in $O(n + m)$ complexity.)

Correctness of the Algorithm

Lemma

If the algorithm outputs an assignment σ, then σ satisfies ϕ.

We will need the following definition: A partial assignment $\sigma \in \{0, 1, *\}^n$ *violate* a clause $C = \ell_i \vee \ell_j$ if: σ_i and σ_j are assigned (i.e.,$\sigma_i, \sigma_j \neq *$) and σ_i doesn't satisfy ℓ_i and σ_j doesn't satisfy ℓ_j. The lemma follows from the following invariance.

Lemma

At the beginning of each iteration, the current partial assignment $\sigma^{(i)}$ does not violate any of the clauses of C.

Chứng minh.

Invariance 2 By induction on i. The basis is trivial as in the first iteration $\sigma = *^n$ and so none of the clauses are violated. Step: we'll prove that none of the clauses C are violated by $\sigma^{(i+1)}$. If both variables of C were assigned before the last iteration, then, by the induction hypothesis, $\sigma^{(i)}$ doesn't violate C, and therefore, so is $\sigma^{(i+1)}$. If both variables of C were assigned in the last iteration, then C must be satisfied by $\sigma^{(i+1)}$, otherwise, the algorithm finds a contradiction. $\square$

Correctness of the Algorithm (cont.)

Lemma

If the algorithm outputs UNSAT, then ϕ is unsatisfiable.

Chứng minh.

- Let ϕ' be the formula at the beginning of the iteration in which A halts, and let x_i be the variable chosen at step (2) of this last iteration.
- Note that ϕ' is a 2-CNF formula and $\phi' \subseteq \phi$ (i.e., all the clauses of ϕ' appear as clauses in ϕ).
- Hence, it suffices to show that ϕ' is unsatisfiable.
- Let ϕ_0=Simplify($\phi', x_i = 0$) and ϕ_1=Simplify($\phi', x_i = 1$). It suffices to show that both ϕ_0 and ϕ_1 are unsatisfiable.
- Recall that the formula UCP(ϕ_0) and the formula UCP(ϕ_1) contain a contradiction.
- The proof now follows by noting that if UCP(ψ) contains a contradiction, then ψ is UNSAT.

□

Therefore, we have an efficient algorithm for SAT of 2-CNF.

Graphical View of 2-SAT

- For a 2-CNF formula ϕ, define the implication graph $G = G_\phi$ as follows:
  - nodes $x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n$
  - for a clause $\ell_i \vee \ell_j$ define the edges:

    $\bar{\ell}_i \to \ell_j$
    $\bar{\ell}_j \to \ell_i$

  <u>Main property</u>: Let σ be a satisfying assignment.

  If σ satisfies a node v, then σ satisfies all nodes u achievable from v.

  The property can be proven by induction on the length of the path.

Theorem

ϕ is satisfiable iff the graph G does not contain a "contradiction path" of the form:

$$\ell_i \to \cdots \to \bar{\ell}_i \to \cdots \to \ell_i.$$

Proof for the previous theorem

1. ($\exists$ contradiction path $\Rightarrow \phi$ is UNSAT):
   - Take a potential assignment σ.
   - If σ satisfies ℓ_i, then by Property it must satisfy $\bar{\ell}_i$. Contradiction.
   - If σ satisfies $\bar{\ell}_i$, then by Property it must satisfy ℓ_i. Contradiction.
2. (ϕ is UNSAT $\Rightarrow \exists$ contradiction path):
   If ϕ is UNSAT $\Rightarrow$ Algorithm Halts.
   $\Rightarrow$ for some x_i we have:
   (a) $\ell_j \leftarrow \cdots \leftarrow x_i \rightarrow \cdots \rightarrow \bar{\ell}_j$
   (b) $\ell_k \leftarrow \cdots \leftarrow \bar{x}_i \rightarrow \cdots \rightarrow \bar{\ell}_k$

   In our graph, if $\ell_i \rightarrow \ell_j$ is an edge, then $\bar{\ell}_j \rightarrow \bar{\ell}_i$ is also an edge.

   By reversing edges and negating:
   (a) $\Rightarrow x_i \rightarrow \cdots \rightarrow \bar{\ell}_j \rightarrow \cdots \rightarrow \bar{x}_i$
   (b) $\Rightarrow \bar{x}_i \rightarrow \cdots \rightarrow \bar{\ell}_k \rightarrow \cdots \rightarrow x_i$

   Therefore, there exists a contradiction path.

1 Introduction

2 2-SAT is in P

3 **SAT Solvers**

WalkSAT: Idea

DPLL: Idea

A Linear Solver

A Cubic Solver

Nguyen An Khuong,
Le Hong Trang,
Huynh Tuong Nguyen,
Tran Van Hoai

WalkSAT: An Incomplete Solver

- **Idea:** Start with a random truth assignment, and then iteratively improve the assignment until model is found.
- **Details:** In each step, choose an unsatisfied clause (clause selection), and "flip" one of its variables (variable selection).

WalkSAT: Details

- **Termination criterion:** No unsatisfied clauses are left.
- **Clause selection:** Choose a random unsatisfied clause.
- **Variable selection:**
  - If there are variables that when flipped make no currently satisfied clause unsatisfied, flip one which makes the most unsatisfied clauses satisfied.
  - Otherwise, make a choice with a certain probability between:
    - picking a random variable, and
    - picking a variable that when flipped minimizes the number of unsatisfied clauses.

DPLL: Idea

- Simplify formula based on pure literal elimination and unit propagation
- If not done, pick an atom p and split: $\phi \wedge p$ or $\phi \wedge \neg p$

Nguyen An Khuong,
Le Hong Trang,
Huynh Tuong Nguyen,
Tran Van Hoai

A Linear Solver: Idea

- Transform formula to tree of conjunctions and negations.
- Transform tree into graph.
- Mark the top of the tree as `T`.
- Propagate constraints using obvious rules.
- If all leaves are marked, check that corresponding assignment makes the formula true.

Transformation

$$
\begin{array}{rcl}
T(p) & = & p \\
T(\phi_1 \wedge \phi_2) & = & T(\phi_1) \wedge T(\phi_2) \\
T(\neg\phi) & = & \neg\phi(T) \\
T(\phi_1 \rightarrow \phi_2) & = & \neg(T(\phi_1) \wedge \neg T(\phi_2)) \\
T(\phi_1 \vee \phi_2) & = & \neg(\neg T(\phi_1) \wedge \neg T(\phi_2))
\end{array}
$$

Example

$$\phi = p \wedge \neg(q \vee \neg p)$$

$$T(\phi) = p \wedge \neg\neg(\neg q \wedge \neg\neg p)$$

Nguyen An Khuong, Le Hong Trang, Huynh Tuong Nguyen, Tran Van Hoai

Binary Decision Tree: Example

See Example 1.48 and Figure 1.12 on page 70.

Nguyen An Khuong, Le Hong Trang, Huynh Tuong Nguyen, Tran Van Hoai

Problem

What happens to formulas of the kind $\neg(\phi_1 \land \phi_2)$?

Nguyen An Khuong, Le Hong Trang, Huynh Tuong Nguyen, Tran Van Hoai

A Cubic Solver: Idea

Improve the linear solver as follows:

- Run linear solver
- For every node n that is still unmarked:
  - Mark n with `T` and run linear solver, possibly resulting in temporary marks.
  - Mark n with `F` and run linear solver, possibly resulting in temporary marks.
  - Combine temporary marks, resulting in possibly new permanent marks

Nguyen An Khuong,
Le Hong Trang,
Huynh Tuong Nguyen,
Tran Van Hoai

An application of SAT solving: Solve Sudoku Boolean Formula

At the end of Chapter 0, we saw that

$$\phi = I \wedge R \wedge C \wedge B$$

- Note that ϕ is in CNF.
- ϕ can be altered so that it contains exactly 3 literals per clause (can be fed to 3-SAT solver).
- Problem: Solve this 3-SAT problem with a suitable solver?

Homeworks and Next Week Plan?

Homeworks

- Read carefully all proofs in this note.
- Try to solve the Sudoku in the Introduction note
- Show that $kSAT \in NPC$ for all $k \geq 3$.
- Do ALL marked questions of Exercises 1.6 in [2].
- Read carefully Subsections 1.6.1 and 1.6.2 in [2].

Next Week?

Predicate Logic

Nguyen An Khuong, Le Hong Trang, Huynh Tuong Nguyen, Tran Van Hoai

Chapter 1c
Advanced Predicate Logic

Discrete Mathematics II

(Materials drawn from **Chapter 2** in:
"Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*, 2nd Ed., Cambridge University Press, 2006.")

Nguyen An Khuong, Huynh Tuong Nguyen
Faculty of Computer Science and Engineering
University of Technology, VNU-HCM

Contents

More Declarative Sentences

- Propositional logic can easily handle simple declarative statements such as:

Example

Student Hung enrolled in DMII.

- Propositional logic can also handle combinations of such statements such as:

Example

Student Hung enrolled in Tutorial 1, *and* student Cuong is enrolled in Tutorial 2.

- *But:* How about statements with *"there exists..."* or *"every..."* or *"among..."*?

What is needed?

Nguyen An Khuong,
Huynh Tuong Nguyen

Example

Every student is younger than *some* instructor.

What is this statement about?

- Being a student
- Being an instructor
- Being younger than somebody else

These are *properties* of elements of a *set* of objects.

We express them in predicate logic using *predicates*.

Predicates

Nguyen An Khuong, Huynh Tuong Nguyen

Example

Every student is younger than *some* instructor.

- $S(\mathit{An})$ could denote that An is a student.
- $I(\mathit{Binh})$ could denote that Binh is an instructor.
- $Y(\mathit{An}, \mathit{Binh})$ could denote that An is younger than Binh.

The Need for Variables

Nguyen An Khuong,
Huynh Tuong Nguyen

Example

Every student is younger than *some* instructor.

We use the predicate S to denote student-hood.
How do we express *"every student"*?

We need *variables* that can stand for constant values, and a *quantifier* symbol that denotes *"every"*.

The Need for Variables

Nguyen An Khuong, Huynh Tuong Nguyen

Example

Every student is younger than *some* instructor.

Using variables and quantifiers, we can write:

$$\forall x(S(x) \to (\exists y(I(y) \land Y(x, y)))).$$

Literally: For every x, if x is a student, then there is some y such that y is an instructor and x is younger than y.

Another Example

Nguyen An Khuong, Huynh Tuong Nguyen

English

Not all birds can fly.

Predicates

$B(x)$: x is a bird
$F(x)$: x can fly

The sentence in predicate logic

$$\neg(\forall x(B(x) \to F(x)))$$

A Third Example

Nguyen An Khuong,
Huynh Tuong Nguyen

English

Every girl is younger than her mother.

Predicates

$G(x)$: x is a girl

$M(x, y)$: y is x's mother

$Y(x, y)$: x is younger than y

The sentence in predicate logic

$$\forall x \forall y (G(x) \land M(x, y) \to Y(x, y))$$

A "Mother" Function

The sentence in predicate logic

$$\forall x \forall y (G(x) \land M(x, y) \to Y(x, y))$$

Note that y is only introduced to denote the mother of x.

If everyone has exactly one mother, the predicate $M(x, y)$ is a function, when read from right to left.

We introduce a function symbol m that can be applied to variables and constants as in

$$\forall x (G(x) \to Y(x, m(x)))$$

Nguyen An Khuong, Huynh Tuong Nguyen

A Drastic Example

English

An and Binh have the same maternal grandmother.

The sentence in predicate logic without functions

$$\forall x \forall y \forall u \forall v (M(y, x) \land M(\textsf{\textit{An}}, y) \land \\ M(v, u) \land M(\textsf{\textit{Binh}}, v) \to x = u)$$

The same sentence in predicate logic with functions

$$m(m(\textsf{\textit{An}})) = m(m(\textsf{\textit{Binh}}))$$

Outlook

Nguyen An Khuong, Huynh Tuong Nguyen

Syntax: We formalize the language of predicate logic, including scoping and substitution.

Proof theory: We extend natural deduction from propositional to predicate logic

Semantics: We describe models in which predicates, functions, and formulas have meaning.

Further topics: Soundness/completeness (beyond scope of module), undecidability, incompleteness results, compactness results, extensions

Predicate Vocabulary

Nguyen An Khuong, Huynh Tuong Nguyen

At any point in time, we want to describe the features of a particular "world", using predicates, functions, and constants. Thus, we introduce for this world:

- a set of predicate symbols $\mathcal{P}$
- a set of function symbols $\mathcal{F}$
- a set of constant symbols $\mathcal{C}$

Arity of Functions and Predicates

Nguyen An Khuong, Huynh Tuong Nguyen

Every function symbol in $\mathcal{F}$ and predicate symbol in $\mathcal{P}$ comes with a fixed arity, denoting the number of arguments the symbol can take.

Special case

Function symbols with arity 0 are called *constants*.

Terms

$$t ::= \; x \mid c \mid f(t, \dots, t)$$

where

- x ranges over a given set of variables **var**,
- c ranges over nullary function symbols in $\mathcal{F}$, and
- f ranges over function symbols in $\mathcal{F}$ with arity $n > 0$.

Examples of Terms

Nguyen An Khuong, Huynh Tuong Nguyen

If n is nullary, f is unary, and g is binary, then examples of terms are:

- $g(f(n), n)$
- $f(g(n, f(n)))$

More Examples of Terms

Nguyen An Khuong,
Huynh Tuong Nguyen

If $0, 1, \ldots$ are nullary, s is unary, and $+, -$ and $*$ are binary, then

$$*(-(2, +(s(x), y)), x)$$

is a term.

Occasionally, we allow ourselves to use infix notation for function symbols as in

$$(2 - (s(x) + y)) * x$$

Formulas

$$\begin{aligned} \phi \quad ::= \quad & P(t_1, t_2, \dots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid \\ & (\phi \rightarrow \phi) \mid (\forall x \phi) \mid (\exists x \phi) \end{aligned}$$

where

- $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$,
- t_i are terms over $\mathcal{F}$ and
- x is a variable.

Conventions

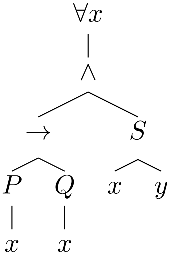
Just like for propositional logic, we introduce convenient conventions to reduce the number of parentheses:

- $\neg, \forall x$ and $\exists x$ bind most tightly;
- then $\wedge$ and $\vee$;
- then $\rightarrow$, which is right-associative.

Parse Trees

$$\forall x((P(x) \to Q(x)) \land S(x, y))$$

has parse tree



Another Example

Every son of my father is my brother.

Predicates

$S(x, y)$: x is a son of y

$B(x, y)$: x is a brother of y

Functions

m: constant for "me"

$f(x)$: father of x

The sentence in predicate logic

$$\forall x(S(x, f(m)) \to B(x, m))$$

Does this formula hold?

Equality as Predicate

Equality is a common predicate, usually used in infix notation.

$$= \in \mathcal{P}$$

Example

Instead of the formula

$$= (f(x), g(x))$$

we usually write the formula

$$f(x) = g(x)$$

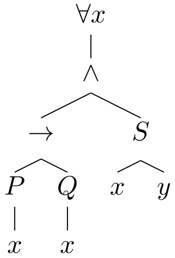
Nguyen An Khuong, Huynh Tuong Nguyen

Free and Bound Variables

Consider the formula

$$\forall x((P(x) \to Q(x)) \wedge S(x, y))$$

What is the relationship between variable "binder" x and occurrences of x?

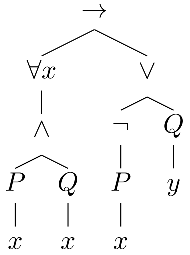


Free and Bound Variables

Consider the formula

$$(\forall x(P(x) \land Q(x))) \to (\neg P(x) \lor Q(y))$$

Which variable *occurrences* are free; which are bound?



Nguyen An Khuong,
Huynh Tuong Nguyen

Substitution

Variables are *place*holders. Re*plac*ing them by terms is called *substitution*.

Definition

Given a variable x, a term t and a formula ϕ, we define $[x \Rightarrow t]\phi$ to be the formula obtained by replacing each free occurrence of variable x in ϕ with t.

Example

$$[x \Rightarrow f(x,y)](\forall x(P(x) \wedge Q(x))) \rightarrow (\neg P(x) \vee Q(y)))$$
$$= \forall x(P(x) \wedge Q(x)) \rightarrow (\neg P(f(x,y)) \vee Q(y))$$

A Note on Notation

Instead of

$$[x \Rightarrow t]\phi$$

the textbook uses the notation

$$\phi[t/x]$$

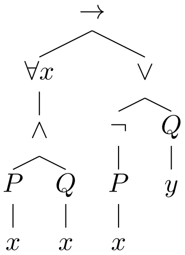
(we find the order of arguments in the latter notation hard to remember)

Nguyen An Khuong,
Huynh Tuong Nguyen

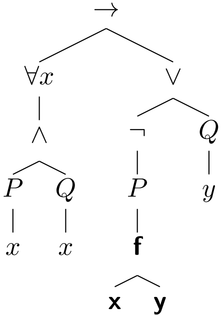
Example as Parse Tree

$$[x \Rightarrow f(x,y)]((\forall x(P(x) \land Q(x))) \rightarrow (\neg P(x) \lor Q(y)))$$

$$= (\forall x(P(x) \land Q(x))) \rightarrow (\neg P(f(x,y)) \lor Q(y))$$



Example as Parse Tree



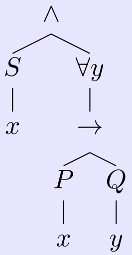
Capturing in $[x \Rightarrow t]\phi$

Problem

t contains variable y and x occurs under the scope of $\forall y$ in ϕ

Example

$$[x \Rightarrow f(y, y)](S(x) \wedge \forall y(P(x) \rightarrow Q(y)))$$



Avoiding Capturing

Nguyen An Khuong,
Huynh Tuong Nguyen

Definition

Given a term t, a variable x and a formula ϕ, we say that t is free for x in ϕ if no free x leaf in ϕ occurs in the scope of $\forall y$ or $\exists y$ for any variable y occurring in t.

Free-ness as precondition

In order to compute $[x \Rightarrow t]\phi$, we demand that t is free for x in ϕ.

What if not?

Rename the bound variable!

Example of Renaming

Nguyen An Khuong, Huynh Tuong Nguyen

$$[x \Rightarrow f(y,y)](S(x) \wedge \forall y(P(x) \rightarrow Q(y)))$$

$$\Downarrow$$

$$[x \Rightarrow f(y,y)](S(x) \wedge \forall z(P(x) \rightarrow Q(z)))$$

$$\Downarrow$$

$$S(f(y,y)) \wedge \forall z(P(f(y,y)) \rightarrow Q(z))$$

Natural Deduction for Predicate Logic

Nguyen An Khuong,
Huynh Tuong Nguyen

Relationship between propositional and predicate logic

If we consider propositions as nullary predicates, propositional logic is a sub-language of predicate logic.

Inheriting natural deduction

We can translate the rules for natural deduction in propositional logic directly to predicate logic.

Example

$$\frac{\phi \qquad \psi}{\phi \wedge \psi}[\wedge i]$$

Built-in Rules for Equality

Nguyen An Khuong, Huynh Tuong Nguyen

$$\frac{}{t = t}\,[=i] \qquad \frac{t_1 = t_2 \qquad [x \Rightarrow t_1]\phi}{[x \Rightarrow t_2]\phi}\,[=e]$$

Nguyen An Khuong,
Huynh Tuong Nguyen

Properties of Equality

We show:

$$f(x) = g(x) \vdash h(g(x)) = h(f(x))$$

using

$$\frac{}{t = t}[= i] \qquad \frac{t_1 = t_2 \qquad [x \Rightarrow t_1]\phi}{[x \Rightarrow t_2]\phi}[= e]$$

1	$f(x) = g(x)$	premise
2	$h(f(x)) = h(f(x))$	$= i$
3	$h(g(x)) = h(f(x))$	$= e$ 1,2

Rules for Universal Quantification

Nguyen An Khuong, Huynh Tuong Nguyen

$$\frac{\forall x \phi}{[x \Rightarrow t]\phi} [\forall x\ e]$$

Example

$$\frac{\forall x \phi}{[x \Rightarrow t]\phi}\,[\forall x \; e]$$

We prove: $F(g(\textit{Duong})), \forall x(F(x) \rightarrow \neg M(x)) \vdash \neg M(g(\textit{Duong}))$

1	$F(g(\textit{Duong}))$	premise
2	$\forall x(F(x) \rightarrow \neg M(x))$	premise
3	$F(g(\textit{Duong})) \rightarrow \neg M(g(\textit{Duong}))$	$\forall x \; e$ 2
4	$\neg M(g(\textit{Duong}))$	$\rightarrow e$ 3,1

Rules for Universal Quantification

Nguyen An Khuong, Huynh Tuong Nguyen

If we manage to establish a formula ϕ about a fresh variable x_0, we can assume $\forall x \phi$.

$$\cfrac{\boxed{\begin{array}{l} x_0 \\ \quad\quad \vdots \\ \quad [x \Rightarrow x_0]\phi \end{array}}}{\forall x \phi} \; [\forall x \; i]$$

Example

Nguyen An Khuong, Huynh Tuong Nguyen

$$\forall x(P(x) \to Q(x)), \forall x P(x) \vdash \forall x Q(x) \text{ via } \frac{\boxed{\begin{array}{l} x_0 \\ \quad \vdots \\ \quad [x \Rightarrow x_0]\phi \end{array}}}{\forall x \phi}$$

1		$\forall x(P(x) \to Q(x))$	premise
2		$\forall x P(x)$	premise
3	x_0	$P(x_0) \to Q(x_0)$	$\forall x\ e$ 1
4		$P(x_0)$	$\forall x\ e$ 2
5		$Q(x_0)$	$\to e$ 3,4
6		$\forall x Q(x)$	$\forall x\ i$ 3–5

Rules for Existential Quantification

Nguyen An Khuong,
Huynh Tuong Nguyen

$$\frac{[x \Rightarrow t]\phi}{\exists x \phi}\ [\exists x\ i]$$

$$\frac{\exists x \phi \qquad \boxed{\begin{array}{ll} x_0 & [x \Rightarrow x_0]\phi \\ & \quad\vdots \\ & \quad\chi \end{array}}}{\chi}\ [\exists e]$$

Example

$$\forall x(P(x) \to Q(x)), \exists x P(x) \vdash \exists x Q(x)$$

1		$\forall x(P(x) \to Q(x))$	premise
2		$\exists x P(x)$	premise
3	x_0	$P(x_0)$	assumption
4		$P(x_0) \to Q(x_0)$	$\forall x\ e$ 1
5		$Q(x_0)$	$\to e$ 4,3
6		$\exists x Q(x)$	$\exists x\ i$ 5
7		$\exists x Q(x)$	$\exists x\ e$ 2,3–6

Examples of Quantifier Equivalences

Nguyen An Khuong, Huynh Tuong Nguyen

$$
\begin{aligned}
\neg\forall x\phi &\dashv\vdash \exists x\neg\phi \\
\neg\exists x\phi &\dashv\vdash \forall x\neg\phi \\
\exists x\exists y\phi &\dashv\vdash \exists y\exists x\phi
\end{aligned}
$$

Assume x is not free in ψ:

$$
\begin{aligned}
\forall x\phi \land \psi &\dashv\vdash \forall x(\phi \land \psi) \\
\exists x(\psi \to \phi) &\dashv\vdash \psi \to \exists x\phi
\end{aligned}
$$

Models

Definition

Let $\mathcal{F}$ contain function symbols and $\mathcal{P}$ contain predicate symbols. A model $\mathcal{M}$ for $(\mathcal{F}, \mathcal{P})$ consists of:

1. A non-empty set A, the *universe*;
2. for each nullary function symbol $f \in \mathcal{F}$ a concrete element $f^{\mathcal{M}} \in A$;
3. for each $f \in F$ with arity $n > 0$, a concrete function $f^{\mathcal{M}} : A^n \to A$;
4. for each $P \in \mathcal{P}$ with arity $n > 0$, a set $P^{\mathcal{M}} \subseteq A^n$.

Example

Let $\mathcal{F} = \{e, \cdot\}$ and $\mathcal{P} = \{\leq\}$.
Let model $\mathcal{M}$ for $(\mathcal{F}, \mathcal{P})$ be defined as follows:

1. Let A be the set of binary strings over the alphabet $\{0, 1\}$;
2. let $e^{\mathcal{M}} = \epsilon$, the empty string;
3. let $\cdot^{\mathcal{M}}$ be defined such that $s_1 \cdot^{\mathcal{M}} s_2$ is the concatenation of the strings s_1 and s_2; and
4. let $\leq^{\mathcal{M}}$ be defined such that $s_1 \leq^{\mathcal{M}} s_2$ iff s_1 is a prefix of s_2.

Nguyen An Khuong, Huynh Tuong Nguyen

Example (continued)

1. Let A be the set of binary strings over the alphabet $\{0, 1\}$;
2. let $e^{\mathcal{M}} = \epsilon$, the empty string;
3. let $\cdot^{\mathcal{M}}$ be defined such that $s_1 \cdot^{\mathcal{M}} s_2$ is the concatenation of the strings s_1 and s_2; and
4. let $\leq^{\mathcal{M}}$ be defined such that $s_1 \leq^{\mathcal{M}} s_2$ iff s_1 is a prefix of s_2.

Some Elements of A

- 10001
- ϵ
- $1010 \cdot^{\mathcal{M}} 1100 = 10101100$
- ϵ
- $000 \cdot^{\mathcal{M}} \epsilon = 000$

Nguyen An Khuong,
Huynh Tuong Nguyen

Equality Revisited

Interpretation of equality

Usually, we require that the equality predicate $=$ is interpreted as same-ness.

Extensionality restriction

This means that allowable models are restricted to those in which $a =^{\mathcal{M}} b$ holds if and only if a and b are the same elements of the model's universe.

Example (continued)

Nguyen An Khuong, Huynh Tuong Nguyen

1. Let A be the set of binary strings over the alphabet $\{0,1\}$;
2. let $e^{\mathcal{M}} = \epsilon$, the empty string;
3. let $\cdot^{\mathcal{M}}$ be defined such that $s_1 \cdot^{\mathcal{M}} s_2$ is the concatenation of the strings s_1 and s_2; and
4. let $\leq^{\mathcal{M}}$ be defined such that $s_1 \leq^{\mathcal{M}} s_2$ iff s_1 is a prefix of s_2.

Equality in $\mathcal{M}$

- $000 =^{\mathcal{M}} 000$
- $001 \neq^{\mathcal{M}} 100$

Another Example

Nguyen An Khuong,
Huynh Tuong Nguyen

Let $\mathcal{F} = \{z, s\}$ and $\mathcal{P} = \{\leq\}$.
Let model $\mathcal{M}$ for $(\mathcal{F}, \mathcal{P})$ be defined as follows:

1. Let A be the set of natural numbers;
2. let $z^{\mathcal{M}} = 0$;
3. let $s^{\mathcal{M}}$ be defined such that $s(n) = n + 1$; and
4. let $\leq^{\mathcal{M}}$ be defined such that $n_1 \leq^{\mathcal{M}} n_2$ iff the natural number n_1 is less than or equal to n_2.

How To Handle Free Variables?

Nguyen An Khuong,
Huynh Tuong Nguyen

Idea

We can give meaning to formulas with free variables by providing an environment (lookup table) that assigns variables to elements of our universe:

$$l : \textbf{var} \to A.$$

Environment extension

We define environment extension such that $l[x \mapsto a]$ is the environment that maps x to a and any other variable y to $l(y)$.

Satisfaction Relation

Nguyen An Khuong,
Huynh Tuong Nguyen

The model $\mathcal{M}$ satisfies ϕ with respect to environment l, written $\mathcal{M} \models_l \phi$:

- in case ϕ is of the form $P(t_1, t_2, \ldots, t_n)$, if the result $(a_1, a_2, \ldots, a_n)$ of evaluating $t_1, t_2, \ldots, t_n$ with respect to l is in $P^{\mathcal{M}}$;
- in case ϕ has the form $\forall x \psi$, if the $\mathcal{M} \models_{l[x \mapsto a]} \psi$ holds for all $a \in A$;
- in case ϕ has the form $\exists x \psi$, if the $\mathcal{M} \models_{l[x \mapsto a]} \psi$ holds for some $a \in A$;

Satisfaction Relation (continued)

- in case ϕ has the form $\neg\psi$, if $\mathcal{M} \models_l \psi$ does not hold;
- in case ϕ has the form $\psi_1 \vee \psi_2$, if $\mathcal{M} \models_l \psi_1$ holds or $\mathcal{M} \models_l \psi_2$ holds;
- in case ϕ has the form $\psi_1 \wedge \psi_2$, if $\mathcal{M} \models_l \psi_1$ holds and $\mathcal{M} \models_l \psi_2$ holds; and
- in case ϕ has the form $\psi_1 \rightarrow \psi_2$, if $\mathcal{M} \models_l \psi_1$ holds whenever $\mathcal{M} \models_l \psi_2$ holds.

Satisfaction of Closed Formulas

Nguyen An Khuong, Huynh Tuong Nguyen

If a formula ϕ has no free variables, we call ϕ a *sentence*.
$\mathcal{M} \models_l \phi$ holds or does not hold regardless of the choice of l. Thus we write $\mathcal{M} \models \phi$ or $\mathcal{M} \not\models \phi$.

Nguyen An Khuong, Huynh Tuong Nguyen

Semantic Entailment and Satisfiability

Let Γ be a possibly infinite set of formulas in predicate logic and ψ a formula.

Entailment

$\Gamma \models \psi$ iff for all models $\mathcal{M}$ and environments l, whenever $\mathcal{M} \models_l \phi$ holds for all $\phi \in \Gamma$, then $\mathcal{M} \models_l \psi$.

Satisfiability of Formulas

ψ is satisfiable iff there is some model $\mathcal{M}$ and some environment l such that $\mathcal{M} \models_l \psi$ holds.

Satisfiability of Formula Sets

Γ is satisfiable iff there is some model $\mathcal{M}$ and some environment l such that $\mathcal{M} \models_l \phi$, for all $\phi \in \Gamma$.

Nguyen An Khuong, Huynh Tuong Nguyen

Semantic Entailment and Satisfiability

Let Γ be a possibly infinite set of formulas in predicate logic and ψ a formula.

Validity

ψ is valid iff for all models $\mathcal{M}$ and environments l, we have $\mathcal{M} \models_l \psi$.

The Problem with Predicate Logic

Nguyen An Khuong, Huynh Tuong Nguyen

Entailment ranges over models

Semantic entailment between sentences: $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ requires that in *all* models that satisfy $\phi_1, \phi_2, \ldots, \phi_n$, the sentence ψ is satisfied.

How to effectively argue about all possible models?

Usually the number of models is infinite; it is very hard to argue on the semantic level in predicate logic.

Idea from propositional logic

Can we use natural deduction for showing entailment?

Central Result of Natural Deduction

$$\phi_1, \ldots, \phi_n \models \psi$$

iff

$$\phi_1, \ldots, \phi_n \vdash \psi$$

proven by Kurt Gödel, in 1929 in his doctoral dissertation

Recall: Decidability

Decision problems

A *decision problem* is a question in some formal system with a yes-or-no answer.

Decidability

Decision problems for which there is an algorithm that returns "yes" whenever the answer to the problem is "yes", and that returns "no" whenever the answer to the problem is "no", are called *decidable*.

Decidability of satisfiability

The question, whether a given propositional formula is satisifiable, is decidable.

Undecidability of Predicate Logic

Nguyen An Khuong,
Huynh Tuong Nguyen

Theorem

The decision problem of validity in predicate logic is undecidable: no program exists which, given any language in predicate logic and any formula ϕ in that language, decides whether $\models \phi$.

Proof

- Establish that the Post Correspondence Problem (PCP) is undecidable (here only as sketch).
- Translate an arbitrary PCP, say C, to a formula ϕ.
- Establish that $\models \phi$ holds if and only if C has a solution.
- Conclude that validity of pred. logic formulas is undecidable.

Post Correspondence Problem

Nguyen An Khuong,
Huynh Tuong Nguyen

Informally

Can we line up copies of the cards such that the top row spells out the same sequence as the bottom row?

Formally

Given a finite sequence of pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$ such that all s_i and t_i are binary strings of positive length, is there a sequence of indices $i_1, i_2, \ldots, i_n$ with $n \geq 1$ such that the concatenations $s_{i_1} s_{i_2} \ldots s_{i_n}$ and $t_{i_1} t_{i_2} \ldots t_{i_n}$ are equal?

Undecidability of Post Correspondence Problem

Nguyen An Khuong,
Huynh Tuong Nguyen

Turing machines

Basic abstract symbol-manipulating devices that can simulate in prinicple any computer algorithm. The input is a string of symbols on a *tape*, and the machine "accepts" the input string, if it reaches one of a number of *accepting states*.

Termination of Programs is Undecidable

It is undecidable, whether program with input terminates.

Proof idea

For a Turing machine with a given input, construct a PCP such that a solution of the PCP exists if and only if the Turing machine accepts the solution.

Translate Post Correspondence Problem to Formula

Nguyen An Khuong, Huynh Tuong Nguyen

Bits as Functions

Represent bits 0 and 1 by functions f_0 and f_1.

Strings as Terms

Represent the empty string by a constant e.
The string $b_1 b_2 \ldots b_l$ corresponds to the term

$$f_{b_l}(f_{b_{l-1}} \ldots (f_{b_2}(f_{b_1}(e))) \ldots)$$

Nguyen An Khuong, Huynh Tuong Nguyen

Towards a Formula for a PCP

Let C be the problem $\begin{matrix} s_1 & s_2 & \dots & s_k \\ t_1 & t_2 & \dots & t_k \end{matrix}$

Idea

$P(s,t)$ holds iff there is a sequence of indices $(i_1, i_2, \dots, i_m)$ such that s is $s_{i_1} s_{i_2} \dots s_{i_m}$ and t is $t_{i_1} t_{i_2} \dots t_{i_m}$.

The Formula ϕ

Nguyen An Khuong, Huynh Tuong Nguyen

$\phi = \phi_1 \wedge \phi_2 \to \phi_3$, where

$$\begin{aligned}
\phi_1 &= \bigwedge_{i=1}^{k} P(f_{s_i}(e), f_{t_i}(e)) \\
\phi_2 &= \forall v \forall w (P(v, w) \to \bigwedge_{i=1}^{k} P(f_{s_i}(v), f_{t_i}(w))) \\
\phi_3 &= \exists z P(z, z)
\end{aligned}$$

Undecidability of Predicate Logic

Nguyen An Khuong, Huynh Tuong Nguyen

So Far

Post correspondence problem is undecidable.
Constructed ϕ_C for Post correspondence problem C.

To Show

$\models \phi_C$ holds if and only if C has a solution.

Proof

Proof via construction of ϕ_C. Formally construct an interpretation of strings and show that whenever there is a solution, the formula ϕ_C holds and vice versa.

Summary of Undecidability Proof

Nguyen An Khuong, Huynh Tuong Nguyen

Theorem

The decision problem of validity in predicate logic is undecidable: no program exists which, given any language in predicate logic and any formula ϕ in that language, decides whether $\models \phi$.

Proof

- Establish that the Post Correspondence Problem (PCP) is undecidable
- Translate an arbitrary PCP, say C, to a formula ϕ.
- Establish that $\models \phi$ holds if and only if C has a solution.
- Conclude that validity of pred. logic formulas is undecidable.

Compactness Theorem

Let Γ be a set of sentences of predicate logic. If all finite subsets of Γ are satisfiable, then Γ is satisfiable.

Proof of Compactness Theorem

Assume Γ is not satisfiable.
We thus have $\Gamma \models \bot$.
Via completeness, we have $\Gamma \vdash \bot$.
The proof is finite, thus only uses a finite subset $\Delta \subset \Gamma$ of premises.
Thus, $\Delta \vdash \bot$, and $\Delta \models \bot$ via soundness.

Reachability not Expressible in Predicate Logic

Nguyen An Khuong,
Huynh Tuong Nguyen

There is no predicate logic formula $\phi_{G,u,v}$ with u and v as its only free variables and R as its only predicate symbol, such that $\phi_{G,u,v}$ holds iff there is a path from u to v in G.

Löwenheim-Skolem Theorem

Nguyen An Khuong, Huynh Tuong Nguyen

Let ψ be a sentence of predicate logic such that for any natural number $n \geq 1$ there is a model of ψ with at least n elements. Then ψ has a model with infinitely many elements.

Homeworks and Next Week Plan?

Homeworks

It is recommended that you should do as much as you can ALL marked exercises in [2, Sect. 2.8] (notice that sample solutions for these exercises are available in [3]). For this lecture, the following are recommended exercises [2]:

- 2.1: 1a); 2a)
- 2.2: 6
- 2.3: 1a); 1b); 6a); 6b); 6c); 7b); 9b); 9c); 13d)
- 2.4: 2); 3); 11a); 11c); 12e); 12f); 12h); 12k)
- 2.5: 1c); 1e).

Next Weeks?

- Exercises Session;
- Applications of FoL.

Chapter 1d
Examples on Using Proposition and Predicate Logic

Discrete Mathematics II

(Materials drawn from **Chapter 2** in:
"Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*, 2nd Ed., Cambridge University Press, 2006.")

Nguyen An Khuong, Huynh Tuong Nguyen
Faculty of Computer Science and Engineering
University of Technology, VNU-HCM

Contents

Electing Puzzle

- Four men and four women are nominated for two positions.
- Exactly one man and one woman are elected.
- The men are A, B, C, D and the women are E, F, G, H. We know:
  - if neither A nor E won, then G won
  - if neither A nor F won, then B won
  - if neither B nor G won, then C won
  - if neither C nor F won, then E won.
- Who were the two people elected?

Huth and Ryan [2], Exercises 2.1.5: Protocol Requirements

- The following sentences are taken from the **RFC3157 Internet Task-force Document 'Securely Available Credentials – Requirements.'**
- Specify it in predicate logic, defining predicate symbols as appropriate:
    a. An attacker can persuade a server that a successful login has occurred, even if it hasn't.
    b. An attacker can overwrite someone else's credentials on the server.
    c. All users enter passwords instead of names.
    d. Credential transfer both to and from a device MUST be supported.
    e. Credentials MUST NOT be forced by the protocol to be present in cleartext at any device other than the end user's.
    f. The protocol MUST support a range of cryptographic algorithms, including symmetric and asymmetric algorithms, hash algorithms, and MAC algorithms.
    g. Credentials MUST only be downloadable following user authentication or else only downloadable in a format that requires completion of user authentication for deciphering.
    h. Different end user devices MAY be used to download, upload, or manage the same set of credentials.

Huth and Ryan [2], Exercises 2.1.5: Solutions

a. An attacker can persuade a server that a successful login has occurred, even if it hasn't:
$\phi := \exists a \exists s((\neg loggedIn(a, s)) \longrightarrow (canPersuade(a, s)))$.

b. An attacker can overwrite someone else's credentials on the server: $\phi := \exists u \exists c \exists s \exists d((\neg ownsCredentials(u, c)) \longrightarrow canWrite(u, c, s, d))$.

Chapter 1e
Predicate Logic and Program Verification

Discrete Mathematics II

(Materials drawn from **Chapter 2** in:
"Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*, 2nd Ed., Cambridge University Press, 2006.")

Nguyen An Khuong, Huynh Tuong Nguyen
Faculty of Computer Science and Engineering
University of Technology, VNU-HCM

Contents

Warm-up questions

a) Are there expressions in Predicate Logic that do not evaluate to TRUE or FALSE? If so, give an example.

Ans.: Terms, unlike predicates and formulas, do not evaluate to the distinguished symbols true or false. Examples of terms include: a, a constant (or 0-ary function); x, a variable; $f(t)$, a unary function f applied to a term t.

b) Is $p(a) \longrightarrow \exists x.p(x)$ a valid formula?

Ans.: Yes

c) How do you represent a propositional variable (as used in Propositional Logic) in a Predicate Logic formula?

Ans.: As a 0-ary predicate.

d) Fermat's Last Theorem is the name of the statement in number theory that: *It is impossible to separate any power higher than the second into two like powers.*

Or, more precisely:

If an integer n is greater than 2, then the equation $x^n + y^n = z^n$ has no solutions in positive integers x, y, and z.

Formulate the above statement in Predicate Logic with Equality?

Warm-up questions (cont'd): An answer to Fermat's Last Theorem Formulation

$$\forall n.integer(n) \wedge n > 2 \longrightarrow \forall x, y, z.integer(x) \wedge integer(y) \wedge integer(z) \wedge x > 0 \wedge y > 0 \wedge z > 0 \longrightarrow x^n + y^n \neq z^n.$$

Program Verification

- Below is a function written in an imperative programming language to perform *binary search*, by returning TRUE iff the array a contains the value e in the range $[l, u]$ and FALSE otherwise, under the assumption that the input range is sorted.

```
bool binarySearch ( int [] a, int l, int u, int e) {
if (l > u) return false ;
else {
int m = (l + u) div 2;
if (a[m] == e) return true ;
else if (a[m] < e) return binarySearch (a, m + 1, u, e);
else return binarySearch (a, l, m - 1, e);
}
}
```

- As a first step towards determining whether an implementation (such as that in the function above) fulfills its specification, the specification has to be formalized. We do so in terms of *preconditions* and *postconditions*.

Program Verification (cont'd)

- A *precondition* specifies what should be true upon entering the function (i.e., under what inputs the function is expected to work).
- The *postcondition* is a formula G whose free variables include only the formal parameters and the special variable rv representing the return value of the function.
- The postcondition relates the function's output (the return value rv) to its input (the parameters).

Prob: Formulate in Predicate Logic the precondition and the postcondition for `binarySearch`.

Program Verification (cont'd): Answer

Nguyen An Khuong, Huynh Tuong Nguyen

- First precondition: $0 \leq l \wedge u < |a|$
- Second precondition:
  $\forall i, j. integer(i) \wedge integer(j) \wedge 0 \leq i \leq j < |a| \longrightarrow a[i] \leq a[j]$
- Postcondition: $rv \longleftrightarrow \exists i. l \leq i \leq u \wedge a[i] = e$

HW

1. Do all HWs which have not been done in previous lectures.
2. Try to understand deeply the following notations/terms

   arity, expression, term, formula, atomic formula, sentence, clause, Backus Naur form (BNF), parse tree, precondition, postcondition, binding priorities, provability, witness, scope, bound, verification, model checking, Hoare triple, and their other related notation/terms.
3. Do exercise 1.5.14 on page 89 in [2].
4. Consider the following program

   ```
   temp := x
   x := y
   y := temp
   ```

   What does this tinny program do? Find preconditions, postconditions and verify its correctness?