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

Nguyen An Khuong,
Huynh Tuong Nguyen

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..."*?

Nguyen An Khuong, Huynh Tuong Nguyen

What is needed?

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*.

Nguyen An Khuong, Huynh Tuong Nguyen

Predicates

Example

Every student is younger than *some* instructor.

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

The Need for Variables

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

Example

Every student is younger than *some* instructor.

Using variables and quantifiers, we can write:

$$\forall x(S(x) \rightarrow (\exists y(I(y) \wedge Y(x, y)))).$$

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

English

Not all birds can fly.

Predicates

$B(x)$: x is a bird

$F(x)$: x can fly

The sentence in predicate logic

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

Nguyen An Khuong, Huynh Tuong Nguyen

A Third Example

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

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

A "Mother" Function

The sentence in predicate logic

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

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

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

A Drastic Example

English

An and Binh have the same maternal grandmother.

The sentence in predicate logic without functions

$$\forall x \forall y \forall u \forall v (M(y, x) \land M(\mathit{An}, y) \land M(v, u) \land M(\mathit{Binh}, v) \rightarrow x = u)$$

The same sentence in predicate logic with functions

$$m(m(\mathit{An})) = m(m(\mathit{Binh}))$$

Nguyen An Khuong, Huynh Tuong Nguyen

Outlook

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

Nguyen An Khuong, Huynh Tuong Nguyen

Nguyen An Khuong,
Huynh Tuong Nguyen

Predicate Vocabulary

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 $\mathcal{C}$

Nguyen An Khuong, Huynh Tuong Nguyen

Arity of Functions and Predicates

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*.

Nguyen An Khuong, Huynh Tuong Nguyen

Terms

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

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 $n > 0$.

Nguyen An Khuong,
Huynh Tuong Nguyen

Examples of Terms

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

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

Nguyen An Khuong, Huynh Tuong Nguyen

More Examples of Terms

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

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

Nguyen An Khuong, Huynh Tuong Nguyen

Formulas

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

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.

Nguyen An Khuong, Huynh Tuong Nguyen

Conventions

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

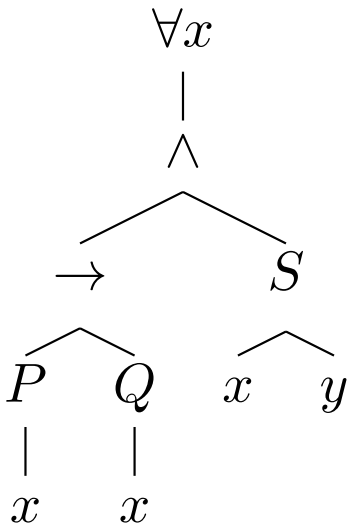
- $\neg, \forall x$ and $\exists x$ bind most tightly;
- then $\wedge$ and $\vee$;
- then $\rightarrow$, which is right-associative.

Nguyen An Khuong, Huynh Tuong Nguyen

Parse Trees

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

has parse tree



Nguyen An Khuong, Huynh Tuong Nguyen

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

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

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

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

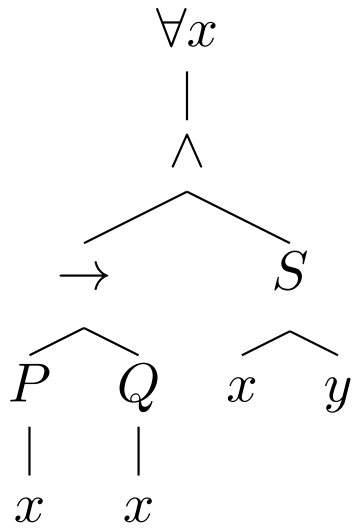
Nguyen An Khuong, Huynh Tuong Nguyen

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?

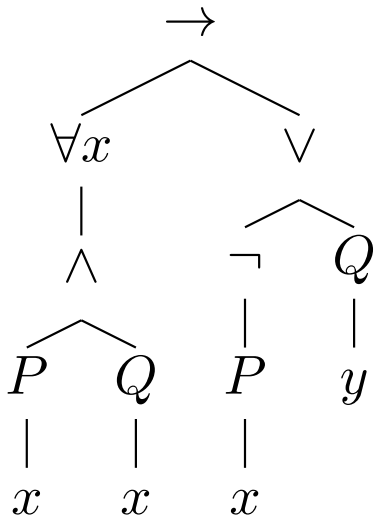


Free and Bound Variables

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

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

Nguyen An Khuong, Huynh Tuong Nguyen

A Note on Notation

Instead of

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

the textbook uses the notation

$$\phi[t/x]$$

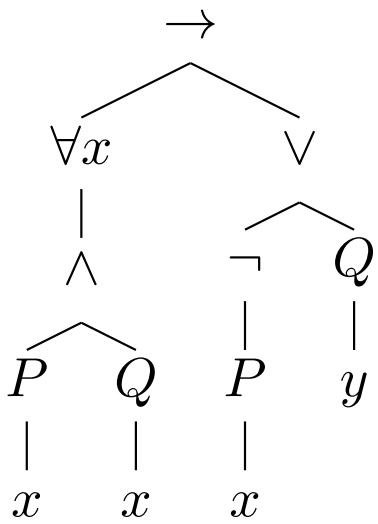
(we find the order of arguments in the latter notation hard to remember)

Nguyen An Khuong, Huynh Tuong Nguyen

Example as Parse Tree

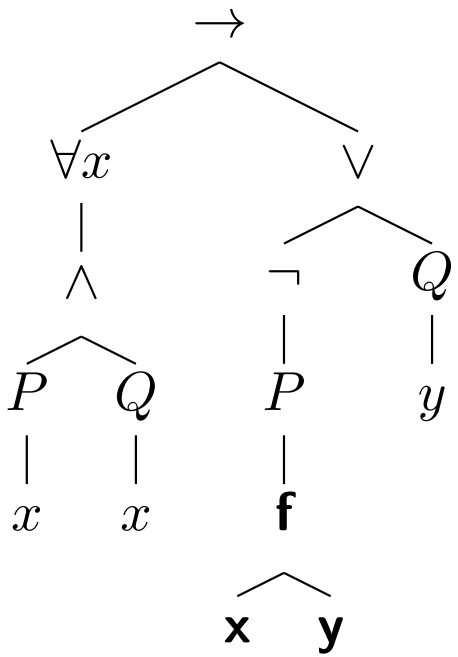
$$[x \Rightarrow f(x,y)]((\forall x(P(x) \wedge Q(x))) \rightarrow (\neg P(x) \vee Q(y)))$$

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



Nguyen An Khuong, Huynh Tuong Nguyen

Example as Parse Tree



Nguyen An Khuong, Huynh Tuong Nguyen

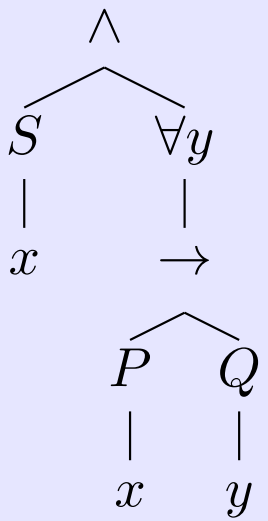
Capturing in $[x \Rightarrow t]\phi$

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

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!

Nguyen An Khuong,
Huynh Tuong Nguyen

Example of Renaming

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

$$\Downarrow$$

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

$$\Downarrow$$

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

Nguyen An Khuong, Huynh Tuong Nguyen

Nguyen An Khuong,
Huynh Tuong Nguyen

Natural Deduction for Predicate Logic

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

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

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

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

Nguyen An Khuong,
Huynh Tuong Nguyen

Example

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

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

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

Rules for Universal Quantification

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

$$\dfrac{\begin{array}{|c|}\hline x_0 \\ \vdots \\ [x \Rightarrow x_0]\phi \\ \hline \end{array}}{\forall x \phi}\,[\forall x\ i]$$

Nguyen An Khuong, Huynh Tuong Nguyen

Example

$$\frac{\begin{array}{|l} x_0 \\ \vdots \\ [x \Rightarrow x_0]\phi \end{array}}{\forall x \phi}$$

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

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

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

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

Nguyen An Khuong,
Huynh Tuong Nguyen

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

Nguyen An Khuong, Huynh Tuong Nguyen

Examples of Quantifier Equivalences

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

Assume x is not free in ψ:

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

Nguyen An Khuong, Huynh Tuong Nguyen

Nguyen An Khuong,
Huynh Tuong Nguyen

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
- $000 \cdot^{\mathcal{M}} \epsilon = 000$

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)

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 $\mathcal{M}$

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

Nguyen An Khuong, Huynh Tuong Nguyen

Another Example

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.

Nguyen An Khuong, Huynh Tuong Nguyen

How To Handle Free Variables?

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 $l(y)$.

Nguyen An Khuong, Huynh Tuong Nguyen

Satisfaction Relation

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.

Nguyen An Khuong, Huynh Tuong Nguyen

Satisfaction of Closed Formulas

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 $\phi \in \Gamma$.

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 $\mathcal{M} \models_l \psi$.

Nguyen An Khuong, Huynh Tuong Nguyen

The Problem with Predicate Logic

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?

Nguyen An Khuong, Huynh Tuong Nguyen

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

Nguyen An Khuong,
Huynh Tuong Nguyen

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.

Nguyen An Khuong, Huynh Tuong Nguyen

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

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

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

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 $b_1 b_2 \dots b_l$ corresponds to the term

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

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 $t_{i_1} t_{i_2} \dots t_{i_m}$.

Nguyen An Khuong, Huynh Tuong Nguyen

The Formula ϕ

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

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

Nguyen An Khuong, Huynh Tuong Nguyen

Undecidability of Predicate Logic

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.

Nguyen An Khuong, Huynh Tuong Nguyen

Summary of Undecidability Proof

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.

Nguyen An Khuong,
Huynh Tuong Nguyen

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.

Nguyen An Khuong,
Huynh Tuong Nguyen

Reachability not Expressible in Predicate Logic

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.

Nguyen An Khuong, Huynh Tuong Nguyen

Löwenheim-Skolem Theorem

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.

Nguyen An Khuong, Huynh Tuong Nguyen

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.