Introduction to Logic
Handout 6

Natthapong Jungteerapanich

International College
King Mongkut's Institute of Technology of Ladkrabang

Inductive Definitions

- In propositional logic, a formula is a finite sequence of logical symbols (such as $\wedge$, $\neg$, ...) and non-logical symbols (i.e. proposition letters).
- But not all finite sequences of these symbols are formulas. Only the sequences which conform to certain rules (or grammar) are counted as formulas.
- In mathematics, there is a systematic way to define an infinite set by describing how the set can be constructed from some base elements and then applying some rules to generate more elements. This kind of definitions is called **inductive definitions**.

Inductive Definitions

one way to define a infinite set

- The set of formulas in propositional logic can be defined inductively as follow:
- Suppose Prop is the set of proposition letters. The set Form of formulas contains all finite sequences which can be constructed by repeatedly applying these rules: consists of all formula in propos— logic.

Rule 1. Prop $\subseteq$ Form → any prop symbol is in set form

" 2. $\top \in$ Form and $\bot \in$ Form → $\top$ and $\bot$ are formula in propositon logic.

" 3. If $\phi \in$ Form, then $(\neg\phi) \in$ Form. → if ϕ is a formula then $\neg\phi$ is a formula.

" 4. If $\phi \in$ Form and $\psi \in$ Form, then $(\phi \wedge \psi) \in$ Form.

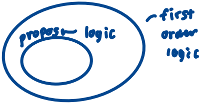
" 5. If $\phi \in$ Form and $\psi \in$ Form, then $(\phi \vee \psi) \in$ Form.

" 6. If $\phi \in$ Form and $\psi \in$ Form, then $(\phi \to \psi) \in$ Form.

" 7. If $\phi \in$ Form and $\psi \in$ Form, then $(\phi \leftrightarrow \psi) \in$ Form.

↳ (rule to defined a set of formula)

↳ mean ϕ ψ need to be a formula.



To define a set

↳ there are 3 ways:

def 1.) $A = \{0, 2, 4, 6, 8, \ldots\}$ → define by listing element

def 2.) $A = \{x \in \mathbb{N} \mid x \text{ is even}\}$ → ⟶ using the set comprehension notation

def 3.) A is defined inductively as follow → define by inductive definition.

→ which can define infinite set.

there are 2 types of rule

base case

↳ R1) $0 \in A$

inductive case → R2) $\forall x \; (x \in A \to x+2 \in A.)$

Show from def 3 that $8 \in A$

1.) $0 \in A$	R1
2.) $2 \in A$	R2, 1
3.) $4 \in A$	R2, 2
4.) $6 \in A$	R2, 3
5.) $8 \in A$	R2, 4

Propositional = $\{p, q, r\}$

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

1.) $p \in$ Form	R1
2.) $q \in$ Form	R2
3.) $(\neg q) \in$ Form	R3, 2
4.) $(p \wedge (\neg q)) \in$ Form	R4, 1, 3

Inductive Definitions

- Suppose A is a non-empty set of letters (also called an **alphabet**). A **string** over A is any finite sequence of elements of A.
- The set of strings over A is typically denoted by A^*.
- A^* contains a unique empty string (i.e. the empty sequence). This is often denoted by ϵ.
- A^* can also be defined inductively as follows:
  1. $\epsilon \in A^*$,
  2. If $x \in A$ and $w \in A^*$, then $w \cdot x \in A^*$.

  Here $w \cdot x$ denotes the concatenation of w with x.

Inductive Definitions

- A **palindrome** is a string whose reverse and itself are equal. For example, *ABBA* and *EBCBE* are palindromes whereas *ABAB* and *EECBB* are not.
- Suppose A is an alphabet. Can we define the set of palindromes over A inductively?
- Let Pal be the set of all strings which can be constructed by repeatedly applying these rules:
  1. $\epsilon \in \text{Pal}$
  2. $A \subseteq \text{Pal}$
  3. If $x \in A$ and $w \in \text{Pal}$, then $x \cdot w \cdot x \in \text{Pal}$.

The set Pal is defined inductively as follows

R1 $\varepsilon \in$ Pal

R2 $a \in$ Pal for any character $a \in A$

R3 $x \in$ Pal $\to$ $axa \in$ Pal for any string x and charater a

1001 $\in$ Pal

$A = \{0,1\}$

1.) $\varepsilon \in$ Pal R1

2.) $00 \in$ Pal R3, 1

3.) $1001 \in$ Pal R3, 2

B = set of string with odd length.

$011 \in B$	Inductive def of B
$1101 \notin B$	R1: $a \in B$ for all $a \in A$
$1 \in B$	R2: $x \in B \to abx \in B$ for any string x
$\varepsilon \notin B$	and character $a, b \in A$

length = 0

Suppose The alphabet $A = \{0,1\}$ + $011 \in B$

1.) $1 \in B$ R1

2.) $011 \in B$ R2, 1

Constructing a Set Inductively

- For simplicity, assume that the set A contains two letters: 0 and 1.
- To construct the set Pal inductively, we successively construct the sets $\mathsf{Pal}_1, \mathsf{Pal}_2, \ldots$.

$$\begin{aligned}
\mathsf{Pal}_1 =& \{\epsilon, 0, 1\} \\
\mathsf{Pal}_2 =& \mathsf{Pal}_1 \cup \{x \cdot w \cdot x \,|\, x \in A \text{ and } w \in \mathsf{Pal}_1\} \\
=& \mathsf{Pal}_1 \cup \{00, 11, 000, 101, 010, 111\} \\
\mathsf{Pal}_3 =& \mathsf{Pal}_2 \cup \{x \cdot w \cdot x \,|\, x \in A \text{ and } w \in \mathsf{Pal}_2\} \\
=& \mathsf{Pal}_2 \cup \{0000, 1001, 0110, 1111, 00000, 10001, \\
& 01010, 11011, 00100, 10101, 01110, 11111\} \\
& \ldots
\end{aligned}$$

- Pal contains all strings that can be constructed this way, i.e.

$$\mathsf{Pal} = \bigcup_{i \geq 1} \mathsf{Pal}_i$$

Proving Properties on Inductively-Defined Sets

- How do we know that the set Pal defined this way contains precisely all the palindromes over alphabet A?
- First, we show that every string in Pal is a palindrome.
- Recall the sets $\mathsf{Pal}_1, \mathsf{Pal}_2, \ldots$:

$$\begin{aligned}
\mathsf{Pal}_1 =& \{\epsilon, 0, 1\} \\
\mathsf{Pal}_2 =& \mathsf{Pal}_1 \cup \{x \cdot w \cdot x \mid x \in A \text{ and } w \in \mathsf{Pal}_1\} \\
\mathsf{Pal}_3 =& \mathsf{Pal}_2 \cup \{x \cdot w \cdot x \mid x \in A \text{ and } w \in \mathsf{Pal}_2\} \\
\ldots =& \ldots
\end{aligned}$$

and

$$\mathsf{Pal} = \bigcup_{i \geq 1} \mathsf{Pal}_i$$

Proving Properties on Inductively-Defined Sets

(1) Clearly, every string in Pal_1 is a palindrome.

(2) Suppose v is a string in Pal_2. There are two possibilities:

- v is in Pal_1. From (1), v is a palindrome.
- $v = x \cdot w \cdot x$, where $x \in A$ and $w \in \mathrm{Pal}_1$. From (1), w is a palindrome. Then, $x \cdot w \cdot x$ is also a palindrome.

Therefore, every string in Pal_2 is a palindrome.

Proving Properties on Inductively-Defined Sets

(3) Suppose v is a string in Pal_3. There are two possibilities:
- v is in Pal_2. From (2), v is a palindrome.
- $v = x \cdot w \cdot x$, where $x \in A$ and $w \in \mathrm{Pal}_2$. From (2), w is a palindrome. Then, $x \cdot w \cdot x$ is also a palindrome.

Therefore, every string in Pal_3 is a palindrome.

(4) ...

Since we can repeat this argument indefinitely for $\mathrm{Pal}_4, \mathrm{Pal}_5, \ldots$, we conclude that every string in Pal is a palindrome.

Proving Properties on Inductively-Defined Sets

- So we have shown that every string in Pal is a palindrome over alphabet A. Next we will show the converse: every palindrome over alphabet A is in Pal.
- Suppose this is not the case, i.e. there are palindromes over A which are not in Pal. Let w be a shortest palindrome over A which is not in Pal.
- Clearly, w must have length 2 or more, because every string of length 0 (i.e. the empty string) and every string of length 1 is in Pal. Thus w must be of the form $x \cdot v \cdot x$, where $x \in A$ and v is a string over A (v is possibly empty).
- Since w is a palindrome, v must also be a palindrome. This means that v is in Pal because w is a shortest palindrome which is not in Pal.

Proving Properties on Inductively-Defined Sets

- But since v is in Pal, by the inductive definition of Pal, $x \cdot v \cdot x$ must also be in Pal. But since $x \cdot v \cdot x$ equals to w, this contradicts our assumption that w is not in Pal.
- By this contradiction, we can conclude that there cannot be a palindrome over alphabet A which is not a member of Pal.
- Therefore, Pal is precisely the set of all palindromes over A.

Recursive Definitions of Functions

- An inductive definition enables us to construct an element from some **smaller** elements in the set.
- Similarly, a **recursive definition** of a function enables us to find the value of the function at some element from the values of the function at some **smaller** elements.
- For example, consider the following recursive definition of function len from A^* to $\mathbb{N}$:

  (LEN1) $\text{len}(\epsilon) = 0$
  (LEN2) $\text{len}(w \cdot x) = \text{len}(w) + 1$, for any $w \in A^*$ and $x \in A$

Recursive Definitions of Functions

Bottom up evaluation

1.) len (ε) = 0 len1

2.) len (a) = 1 len2

3.) len(aa) = 2 len2

4.) len (aaa) = 3 len2

5.) len(aaaa) = 4 len2

Based

larger element.

Top-down evaluation.

- Suppose $A = \{a, b\}$.
- Let's see how we can find the value of len($aaaa$).

aaa · a

w x

len (w·x) = len(w)+1

→ len(aaa · a)

→ (len(aa) + 1) + 1

$$\begin{aligned}
\text{len}(aaaa) &= \text{len}(aaa) + 1 && \text{by (LEN2)}\\
&= \text{len}(aa) + 2 && \text{by (LEN2)}\\
&= \text{len}(a) + 3 && \text{by (LEN2)}\\
&= \text{len}(\epsilon) + 4 && \text{by (LEN2)}\\
&= 0 + 4 = 4 && \text{by (LEN1)}
\end{aligned}$$

→ len(a) = len (ε · a)

= len(ε) + 1

TOP

base

[len (a) + 3 = len (ε) + 1 + 3

= len(ε) + 4]

len (ε) + 4 = 0 + 4

= 4

#

Recursive Definitions of Functions

- Let's look at the recursive definition of len again:

  (LEN1) $\text{len}(\epsilon) = 0$

  (LEN2) $\text{len}(w \cdot x) = \text{len}(w) + 1$, for any $w \in A^*$ and $x \in A$
- A recursive definition typically consists of two parts.
- The first part contains one or more conditions on the values of the function at some most basic elements. These conditions are called the **basis clauses** (or the **basis cases**)
- The second part contains one or more conditions which relate the value of the function at some larger element on the value of the function at some smaller elements. These conditions are called the **recursive clauses** (or the **recursive cases**).
- In the definition above, (LEN1) is the basis clause and (LEN2) is the recursive clause.

R0: $f(0) = 0$ } based case

R1: $f(1) = 1$

R2: $f(n) = f(n-1) + f(n-2)$ for any integer $n \geq 2$ → Recursive case

TOP → DOWN

$$
\begin{aligned}
f(6) &= f(5) + f(4) \\
&= f(4) + f(3) + f(4) \\
&= f(3) + f(2) + f(3) + f(4) \\
&= f(2) + f(1) + f(2) + f(3) + f(4) \\
&= f(1) + f(0) + f(1) + f(2) + f(3) + f(4) \\
&= \underbrace{1 + 0}_{2} + 1 + f(1) + f(0) + f(3) + f(4) \\
&= \underbrace{2 + 1 + 0}_{3} + f(2) + f(1) + f(4) \\
&= 3 + f(1) + f(0) + f(1) + f(4) \\
&= 3 + 1 + 0 + 1 + f(3) + f(2) \\
&= 3 + 1 + 0 + 1 + f(2) + f(1) + f(2) \\
&= 3 + 1 + 0 + 1 + f(1) + f(0) + f(1) + f(2) \\
&= 5 + 1 + 0 + 1 + f(1) + f(0) \\
&= 5 + 1 + 0 + 1 + 1 + 0 \\
&= 8 \ \#
\end{aligned}
$$

Smartly way:

$$
\begin{aligned}
f(6) &= f(5) + f(4) \\
&= f(4) + f(3) + f(4) \\
&= 2 \cdot f(4) + f(3) \\
&= 2(f(3) + f(2)) + f(3) \\
&= 3f(3) + 2f(2)
\end{aligned}
$$

$f(6)$: BOTTUM evaluation

1.) $f(0) = 0$ R0

2.) $f(1) = 1$ R1

3.) $f(2) = f(1) + f(0) = 1 + 0 = 1$ R2

4.) $f(3) = f(2) + f(1) = 1 + 1 = 2$ R2

5.) $f(4) = f(3) + f(2) = 2 + 1$ R2

6.) $f(5) = f(4) + f(3) = 3 + 2 = 5$ R2

7.) $f(6) = f(5) + f(4) = 5 + 3 = 8$ R2

Recursive Definitions of Functions

- Consider another example: let double be the function from A^* to A^* defined recursively as follows:

  $A = \{0, 1\}$

  A^* all strings on A

  (DBL1) double$(\epsilon) = \epsilon$
  (DBL2) double$(w \cdot x) =$ double$(w) \cdot xx$, for any $w \in A^*$ and $x \in A$

  Z

- Let's see how we can find the value of double(ab).

$$\begin{aligned} \text{double}(\overset{w\,x}{ab}) &= \text{double}(\overset{w}{a}) \cdot \overset{x\,x}{bb} && \text{by (DBL2)} \\ &= \text{double}(\epsilon) \cdot aabb && \text{by (DBL2)} \\ &= \epsilon \cdot aabb = aabb && \text{by (DBL1)} \end{aligned}$$

$a = \varepsilon \cdot a$

$$\begin{aligned} \text{double}(a) = \text{double}(\varepsilon \cdot a) &= \text{double}(\varepsilon) \cdot aa \\ &= \varepsilon \cdot aa \\ &= aa \end{aligned}$$

Recursive Definitions of Functions

factorial func

$$\text{fac}(n) = n \cdot (n-1) \cdot n(n-2) \cdot \ldots 2 \cdot 1$$

- In general, a recursive definition is suitable for a function whose domain is a set which can be well-ordered (such as natural numbers and strings).
- Let fac be the function from $\mathbb{N}$ to $\mathbb{N}$ defined recursively as follows:

  (FAC1) $\text{fac}(0) = 1$
  (FAC2) $\text{fac}(n+1) = (n+1) * \text{fac}(n)$, for any $n \in \mathbb{N}$

  } factorial func through recusive func.

Recursive Definitions of Functions

- A recursive definition for a function with more than one argument can be given similarly.
- Let super be the function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ defined recursively as follows:

  (SUP1) $\text{super}(0, n) = 1$, for any $n \in \mathbb{N}$
  (SUP2) $\text{super}(m, 0) = 1$, for any $m \in \mathbb{N}$ where $m > 0$
  (SUP3) $\text{super}(m+1, n+1) = (m+1) * (n+1) * \text{super}(m, n)$, for any $m, n \in \mathbb{N}$
- Can you find the value of $\text{super}(5, 3)$?

Recursive Definitions of Functions

- A good recursive definition should enable us to compute unambiguously the value of the function at every point in the domain that the function is intended to define. A recursive definition with this property is said to be **well defined**.
- If we are not careful, we may come up with a recursive definition which is not well-defined. The following are some common examples.
- Circular definitions:

  (CIR1) $f(0,0) = 1$

  (CIR2) $f(\underbrace{m}_{1} + 1, \underbrace{n}_{0} + 1) = 2 * f(\underbrace{n}_{0} + 1, \underbrace{m}_{1} + 1)$, for any $m, n \in \mathbb{N}$

→ bad recursive func

↳ if good always return base case.

$f(1,0) = 2 \cdot f(0,1)$

$= 2 \cdot (2 \cdot f(1,0))$

$= 2 \cdot (2 \cdot (2 \cdot f(0,1)))$

$= \ldots$ → infinite

Recursive Definitions of Functions

$g(2) = g(1+1) = 10 \cdot g(1)$
$= 10 \cdot 10 \cdot g(0) = 100$

$g(2) = g(0+2) = 20 \cdot g(0)$
$= 20 \cdot 1$
$= 20$

- Ambiguous definitions:

  (AM1) $g(0) = 1$
  (AM2) $g(n+1) = 10 * g(n)$, for any $n \in \mathbb{N}$
  (AM3) $g(n+2) = 20 * g(n)$, for any $n \in \mathbb{N}$

  Can you see that you can compute the value of $g(2)$ in more than one way and obtain different results? → both rule can be applied but return different result.

- Incomplete definition:

  (IN1) $h(0) = 1$
  (IN2) $h(n+2) = 2 * h(n)$, for any $n \in \mathbb{N}$ → 0, 1, ...

  It is not possible the compute the value of $h(m)$ where m is odd.

$h(1)$?!!

Recursive Definitions of Functions

- Diverging definition: → similar to circular but worse

  (DI1) $p(0) = 0$

  (DI2) $p(n+1) = (n+1) * p(n+2)$, for any $n \in \mathbb{N}$

Can you see that any attempt to compute the value of $p(1)$ will lead us to compute $p(2), p(3), p(4), \ldots$ and never terminate?

มันเพิ่มขึ้นเรื่อยๆ จน we are not reaching based case.

$p(1) = p(0+1) = (0+1) \cdot p(0+2)$

$= 1 \cdot p(2)$

$= (1+1) \cdot p(1+2)$

$= 2 \cdot p(3)$

$= 2 \cdot p(3) = 2 \cdot 3 \cdot p(4) = 2 \cdot 3 \cdot 4 \cdot p(5)$