Functions - Supplementary

COMP9020 Tutorial

JIAPENG WANG

24T3 Week5, H18B & F15A

1 Tutorial Outline

1.1 Definition

Ask yourself these questions below to test your understanding.

- What is the definition of the *functions*?
  - How to prove a binary relation is (Fun)? (Tut)?
  - How to prove a function is an *Injection*? A *Surjection*?
- What *operations* can be performed on functions?
  - How to compute them?
  - What are the differences between *Converse* and *Inverse* of a function?
- What special functions do we have? What are the *Boolean Functions*?
  - What is the meaning of !, ||, and &&?
  - What *laws* do these operators have?
  - What are the *Normal Forms* for a Boolean Functions?
- How to compare two functions?
  - What is the *Big-O notation* ranking of common functions?

1.2 Brainstorming

The following questions are open and have no standard answers.

- What advantages are there in defining functions through binary relations?
- What are the applications of Boolean functions in computer science?
- Why is the idea of comparing two functions so important in CS?
- ......

2 Preview of Next Week's Content

Since our tutorial sessions are on Thursday or Friday, there isn't much time left after our lectures for a recap. Therefore, I'm sharing the questions we'll cover in our next week's tutorial ahead of time, so you can start thinking about them.

Next week will be a **Problem-set Session**, and I'll also prepare a PowerPoint to guide you through both analysis and writing. These topics are:[1]

Topic 1 **Convert the Notations**

1.1 Set Notation: Roster, Builder (/Interval)

1.2 CS/Math Language Notation: Formal Language

1.3 Relation Notation: Set (/Class), Matrix, Graph (/Hasse diagram)

1.4 Function Notation

1.5 Boolean Notation: Disjunctive Normal Form (/`CNF`)

1.6 Logic Notation: Language, Proposition (/`wffs`)

Topic 2 **Check the Property**

2.1 Elements: (`R`), (`AR`), (`S`), (`AS`), (`T`)

2.2 Domain and Co-domain: (`Fun`), (`Tot`), (`Inj`), (`Sur`) (/(`Bij`))

2.3 Laws: (`Comm.`), (`Asso.`), (`Dist.`), (`Iden.`), (`Comp.`)

Topic 3 **Compute the Operations**

3.1 Number Operations: $\lfloor x \rfloor$ and $\lceil x \rceil$; `div` and `%`; `gcd` and `lcm`

3.2 Set Operations: $\#$; ^{c}, $\cup$, $\cap$ $(, \setminus, \oplus)$; `Pow`

3.3 Relation Operations: $R^{\leftarrow}$, $R(A)$, $R_1; R_2$; `lub` and `glb` (/Lattice)

3.4 Function Operations: $f^{-1}, f(A), T^S, f \circ g$

3.5 Boolean Operations: !, ||, &&

3.6 Logic Operations: $\neg$, $\vee$, $\wedge$ (, $\rightarrow$, $\leftrightarrow$)

Topic 4 **Prove the Relations**

4.1 Number Relations: $|$; $=_k$

4.2 Set/ Bool./ Prop. Relations: $=$ and $\equiv$ (/$\subseteq$)

4.3 Function Relation: $O(f(x))$

4.4 Proposition Relation: $\models$

[1] This is only based on my experience and not the official scope of examination, so there may be omissions.

3 Explanation for DNF

`DNF` (**Disjunctive Normal Form**) is a canonical normal form of a logical formula consisting of a disjunction ($\vee$) of conjunctions ($\wedge$), i.e., an `OR` ($\|$) of `AND`s (&&); it is also called **Sum-of-Products (`SoP`) Form**.

3.1. Expression from Definition:

- An $\vee$ of $\wedge$s:

$$\bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{n} l_{ij} \right)$$

★ An `OR` of `AND`s:

$$(l_{11} \ \&\& \ \dots \ \&\& \ l_{1n}) \ \| \ (l_{21} \ \&\& \ \dots \ \&\& \ l_{2n}) \ \| \cdots \| \ (l_{m1} \ \&\& \ \dots \ \&\& \ l_{mn})$$

- A sum of products:

$$\sum_{i=1}^{m} \left(\prod_{j=1}^{n} l_{ij} \right)$$

3.2. Expression for Canonical `DNF` (`CDNF`):

In Boolean algebra, any Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ can be expressed in the **Canonical Disjunctive Normal Form** (`CDNF`, or $f_{\texttt{DNF}}$), also known as the `DNF` which is an `OR` of minterms[2] where **each of the n variables appears EXACTLY once**.

This form can be useful for the simplification of Boolean functions, which is of great importance in the optimization of Boolean formulas in general and *digital circuits* in particular.

We're able to do this thanks to the favorable properties of this specific midterm. For any $\mathbf{b} = (b_1, \dots, b_n) \in \mathbb{B}^n$, we can define this kind of minterm as:

$$m_{\mathbf{b}} = l_1(x_1) \ \&\& \ l_2(x_2) \ \&\& \ \dots \ \&\& \ l_n(x_n)$$

where $(1 \le i \le n)$

$$l_i(x_i) = \begin{cases} x_i & \text{if } b_i = 1 \\ !x_i & \text{if } b_i = 0 \end{cases}$$

By the definition of the `AND` operation, we know that for m_b to equal 1, each $l_i(x_i)$ must also be 1. Then by the definition of $l_i(x_i)$, we have all $l_i(x_i)$ are equal to 1 if and only if $(x_1, x_2, \dots, x_n) = (b_1, b_2, \dots, b_n) = \mathbf{b}$. Therefore, if we consider the truth table for this function, **ONLY** the row $(b_1, b_2, \dots, b_n)$ will have an output of 1, while all other rows will be 0.

[2] In our course, the definition of **minterms** is:
A Boolean function of the form $\texttt{AND}(l_1(x_1), l_2(x_2), \dots, l_n(x_n))$ where the l_i are literals.

x_1	x_2	$\cdots$	x_n	$m_{\mathbf{b}}(x_1, x_2, \cdots, x_n)$
0	0	$\cdots$	0	0
0	0	$\cdots$	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
b_1	b_2	$\cdots$	b_n	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	$\cdots$	1	0

Therefore, this function represents only the specific case of $\mathbf{b}$.

For any Boolean function, we can always find its truth table. According to the definition of *equality* in Boolean function, we can break it down into an `OR` function combined by these special minterms corresponding to the rows where the output is 1 (because of the definition of `OR`). This allows us to convert any Boolean function into its Disjunctive Normal Form (`DNF`).

Exercise 10. Convert $(x \parallel y)$ && $(x$ && $(!y \parallel z))$ into `DNF`.

Analysis:

Let $f(x, y, z) = (x \parallel y)$ && $(x$ && $(!y \parallel z))$. Let's convert it directly into `CDNF`.

First, write out its truth table.

x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

We find that this function is equal to 1 only in rows 5, 6, and 8, allowing us to rewrite the function using `OR` as follows:

$$f(x, y, z) = m_{(1,0,0)} \parallel m_{(1,0,1)} \parallel m_{(1,1,1)}$$

Next, based on the earlier definition of $m_{\mathbf{b}}$, we express them by x, y, z:

- $m_{(1,0,0)}(x, y, z) = (x \text{ \&\& } !y \text{ \&\& } !z)$
- $m_{(1,0,1)}(x, y, z) = (x \text{ \&\& } !y \text{ \&\& } z)$
- $m_{(1,1,1)}(x, y, z) = (x \text{ \&\& } y \text{ \&\& } z)$

Finally, substituting them back into the expression for $f(x, y, z)$, we obtain the `CDNF`:

$$f(x, y, z) = (x \text{ \&\& } !y \text{ \&\& } !z) \parallel (x \text{ \&\& } !y \text{ \&\& } z) \parallel (x \text{ \&\& } y \text{ \&\& } z). \quad \square$$

For `CNF`, we similarly have `CCNF`. However, obtaining `CCNF` in a general way requires applying the complement operation on truth table and then using *De Morgan's laws*, which is relatively more complex. We won't go into detail here.

It's also worth to note that the `DNF` form is not necessarily unique! For example, in this **Exercise 11**, using a *Karnaugh map* or applying certain laws can yield other `DNF` forms.

$$
\begin{aligned}
f(x, y, z) &= \Big(\big((x \,\&\&\, !y) \,\&\&\, !z\big) \parallel \big((x \,\&\&\, !y) \,\&\&\, z\big)\Big) \parallel (x \,\&\&\, y \,\&\&\, z) \\
&= \Big((x \,\&\&\, !y) \,\&\&\, (!z \parallel z)\Big) \;\parallel\; (x \,\&\&\, y \,\&\&\, z) && (\texttt{Dist.}) \\
&= \Big((x \,\&\&\, !y) \,\&\&\, (z \parallel !z)\Big) \;\parallel\; (x \,\&\&\, y \,\&\&\, z) && (\texttt{Comm.}) \\
&= \Big((x \,\&\&\, !y) \,\&\&\, 1\Big) \;\parallel\; (x \,\&\&\, y \,\&\&\, z) && (\texttt{Comp.}) \\
&= (x \,\&\&\, !y) \;\parallel\; (x \,\&\&\, y \,\&\&\, z) && (\texttt{Iden.})
\end{aligned}
$$

This result is also in `DNF`. Additionally, we observe that, in this example, the `CDNF` is not the `DNF` which have the *minimal* number of minterms.