Discrete Mathematics (II) Spring 2019

Lecture 8: Application of Propositional Logic

Lecturer: Yi Li

1 Overview

Compactness is a very important property in mathematics. It has occurred in Calculus. It is also a key role in logic.

Mathematical Logic is a corner stone of modern computer science. In this lecture, we also show you how to apply propositional logic to other problems.

2 Circuit and Proposition

Let's take $0, 1$ as F, T respectively. We can use circuit gate to represent $\wedge, \vee$ and $\neg$ as *and,or,not* respectively. So given any proposition, we can design a circuit to compute the truth value with the specific input.

Example 1. *Consider the circuit*[1] *for the following propositions:*

1. $(A_1 \wedge A_2) \vee (\neg A_3))$
2. $(A \wedge B \wedge D) \vee (A \wedge B \wedge \neg C)$

The complexity of circuits depends on the complexity of proposition. We usually call the depth of proposition as *delay* of circuit and the number of gates as *power consumption*. To design a good circuit, we try to minimize delay and power consumption.

Example 2. *Consider the boolean function majority of* $\{A, B, C\}$*. It means that the value of function depends on the majority of input.*

Solution: We first consider its truth value table, we can simple connectives to represent majority.

$$\begin{aligned} m(A, B, C) &= (A \wedge B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge C) \\ &= (B \wedge C) \vee (A \wedge C) \vee (A \wedge B) \\ &= (A \wedge (B \vee C)) \vee (B \wedge C) \end{aligned}$$

In another way, we can depict the boolean function with a state diagram and then design the circuit according to it. This method will be introduced in the successor course *Digital Component Design*.

[1] We need special packages to draw the circuit. It could be drawn in the future.

3 Formalize Problems with Propositions

Example 3. *Suppose there is a murder case with three suspects. The police queried them about murder case.*

A said,"I didn't do it. The victim is a friend of B. And C hates him." B said, "I didn't do it. Even I don't know him. And I am not present." C said, "I didn't do it. I saw A and B stayed with the victim in that day. The murder must be one of them."

Solution: Suppose only murder would lie. We try to formalize it with the following propositions:

1. A: A killed victim.
2. BKV: B knows the victim.
3. AP: A is present.
4. CHV: C hates the victim.
5. $(A \wedge \neg B) \vee (\neg A \wedge B)$: murder is either A or B.

Now we can represent the statement of each suspects as following:

1. A: $\neg A \wedge BKV \wedge CHV$.
2. B: $\neg B \wedge \neg BKV \wedge \neg BP$.
3. C: $\neg C \wedge AP \wedge BP \wedge ((A \wedge \neg B) \vee (\neg A \wedge B))$.

It is easy to know that the maximal satisfiable set of propositions can only contain A's and C's statement. Then we can imply that B would be the murder.

Example 4. *Consider the pigeonhole principle:* $f : n^+ \to n, \exists i, j, f(i) = f(j),$ *where* $0 \leq i < j \leq n$.

Solution: let p_{ij} means $f(i) = j$. Then we can describe everywhere defined property as

$$\alpha_1 = \wedge_{0 \leq i \leq n} \vee_{0 \leq j < n} p_{ij}$$

and we can describe single value as

$$\alpha_1 = \wedge_{0 \leq i \leq n} \wedge_{0 \leq j \neq k < n} \neg(p_{ij} \wedge p_{ik})$$

Now we can describe pigeonhole principle as

$$\varphi = (\alpha_1 \wedge \alpha_2) \wedge (\vee_{0 \leq i < j \leq n} \vee_{0 \leq k < n} (p_{ik} \wedge p_{j,k}))$$

Remark: This form of pigeonhole is much harder to recognize than its original version. However, it is described by a much more rigid language compared with our natural language, which could result in ambiguities.

In fact, mathematical logic can be applied into program verification, model checking, and such other applications. European Space Agency only uses software which has gotten through program verification.

Given a piece of code segment as:

Listing 1: A piece of C segment

```
y = 1;
z = 0;

while (z !=x) {
    z = z + 1;
    y = y * z;
}
```

It can be formalized as R P S, where P is the piece of code segment. R and S are two propositions describe status before and after code segment respectively. The code segment P can also be described by set of propositions. Then we just verify $\{R\} \cup P \vdash S$ holds. R and S can be determined via requirement analysis. But the trouble is that P could be hard to represent with logic.

4 Application of Compactness Theorem

Compactness theorem is a very important result in mathematical logic. It establishes a connection between infinity and finitude.

Example 5. *Given an infinite planar graph. If its every finite subgraph is k-colorable, then the graph itself is also k-colorable.*

A graph is $G =< V, E >$, where V is the set of vertices and $E \subseteq V^2$ is the set of all edges. If $(a, b) \in E$, it is denoted as aEb sometimes. G is k-colorable if V can be decomposed into k-color classes $V = C_1 \cup C_2 \cup \cdots \cup C_k$, where $C_i \neq \emptyset$ and $C_i \cap C_j = \emptyset$ if $i \neq j$. It is obvious $(a, b) \notin E$ if $a, b \in C_i$.

A finite subgraph is k-colorable. It means there is a k-colorable graph $G_0 =< V_0, E_0 >$, where $V_0 \subset V$ is a finite set and $E_0 \subset E$ is the set determined by V_0.

Proof. Let $p_{a,i}$ represent vertex a is colored with i. We can formulate a graph which is k-colorable with the following propositions.

1. $p_{a,1} \vee p_{a,2} \vee \cdots \vee p_{a,k}$, for every $a \in V$. It means every vertex could be colored with at least one of k colors.
2. $\neg(p_{a,i} \wedge p_{a,j}), 1 \leq i < j \leq k$ for all $a \in V$. It means $C_i \cap C_j = \emptyset$.
3. $\neg(p_{a,i} \wedge p_{b,i}), i = 1, \ldots, k$ for all aEb. It means no neighbors have the same color .

Then we get a set S with infinite propositions. For any finite subset $S_0 \subset X$, we can extract vertices V_0 from it and construct a set S_1 which describe the graph G_0 generated by V_0. For every finite subgraph is k-colorable, S_1 is satisfiable. S_0 must be satisfiable.

According to compactness theorem, S is satisfiable which means the graph is k-colorable. □

Remark: Generally, this theorem is not easy to prove via graph approach. Because there is no effective way to find it possible that a big graph is still k-colorable merged by two k-colorable graph. However, Compactness Theorem does not need this requirement. Here, we should be aware that a set of propositions is constructed and we try to prove that every finite subset of it is satisfiable, which is the essence of Compactness Theorem.

Example 6. *Every set S can be (totally) ordered.*

Similarly, it can be proved like k-colorable infinite graph. The point is to represent our problem with a set of propositions. This is left as an exercise.

Hints: A set is partial order at least. If you can change a partially ordered set into a linear order set, you successfully complete the proof.

5 König Lemma and Compactness Theorem

In our textbook, Compactness Theorem is proved by König lemma. Now we will show you that it can be proved by Compactness Theorem.

Lemma 1 (König)**.** *A infinite tree with finite branch has a infinite path.*

Actually, the problem can be represented as following. If every $a \in T$ has only finitely many immediate successor and T contains arbitrarily long finite paths, then there is a infinite path in T starts at root.

Proof. Tree is a hierarchical structure. It means that the vertices of a tree could be divided into many sets which corresponds to vertices in some level. So we can define $S_0 = \{c | c \text{ is the root}\}$ and $S_k = \{b \in T | \text{ there is a } a \in S_k \text{ and } b \text{ is a immediate successor of } a\}$. For every k, S_k is finite and no S_k is empty because of infinity of given tree.

Denote p_a as that vertex a is in path P. We now represent a tree with a set of propositions, Σ, as following:

1. $\vee_{a \in S_k} p_a$: there is at least one vertex of level k in a path;
2. $\wedge_{a,b \in S_k} \neg(p_a \wedge p_b)$: there is only one vertex of level k in path p.
3. $p_a \to p_b$: b is a immediate successor of a.

For there are infinite vertices, we have infinite propositions. If they are all satisfiable. We do know there is a infinite path. Given a subset Σ_0, it is just a part of subtree with height k. Then we just add missed propositions into Σ_0 and obtain the subtree represented by Σ_0'. As the tree has

arbitrarily long finite paths. We know Σ_0' is satisfiable and also Σ_0. Now we just apply compactness theorem to get final result. □

In textbook, compactness theorem of proposition logic is proved based on König lemma. Here we prove inversely. It means that they are equivalent.

Exercises

1. Design a circuit for multiply with two two bits input and four bits output. For example, we have $1 * 1 = 1, 10 * 11 = 110$.
2. Brown, Jones, and Smith are suspected of a crime. They testify as follows:
   (a) Brown: Jones is guilty and Smith is innocent.
   (b) Jones: If Brown is guilty then so is Smith.
   (c) Smith: I'm innocent, but at least one of the others is guilty.

   Represent their testimonies with propositions and show who would be the criminal.
3. Every set S can be (totally) ordered. (*Hint*: Use proposition to represent partial order and dichotomy, then try to apply compactness theorem.)
4. Ex 7/p46.