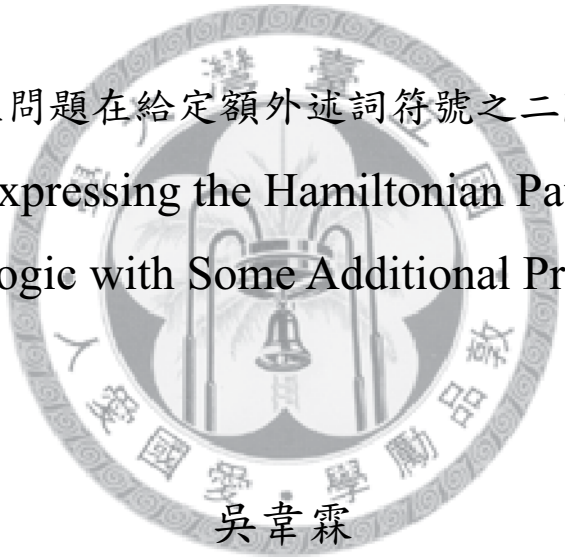
國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis



探討漢彌爾頓路徑問題在給定額外述詞符號之二階邏輯下的表示法

A Study on Expressing the Hamiltonian Path Problem in Second-Order Logic with Some Additional Predicate Symbols

吳韋霖

Wei-Lin Wu

指導教授：趙坤茂 博士

Advisor: Kun-Mao Chao, Ph.D.

中華民國 98 年 6 月

June, 2009

誌謝

本篇論文的誕生，以我一個非相關領域背景的學生來說，一個人獨力完成是不可能的任務。對於所有曾經陪伴我、幫助我的各位，我謹以這篇論文來表達對你們的感謝!

首先，我要感謝的是我的指導教授趙坤茂老師。趙老幽默風趣、親切和藹的態度，始終讓我印象深刻。猶記得我第一次上台報告的表現不甚理想，老師仍然給我許多正面評價，讓我能夠在接下來幾次報告的表現都信心十足並且表現得越來越好。在碩士生涯的兩年裡，我時常態度散漫、不知精益求精。然而老師卻胸襟寬大，以鼓勵代替責備。並且在我遭逢重大挫折、心情低潮的時候，時時給予關懷和溫暖。另一方面，我也感謝老師在我們的研究上賦予極大的自由，使我能夠在自己感興趣的領域---數理邏輯和計算理論---上，盡情地發揮，並給予指導和肯定。老師的恩情，我永生難忘!

接著，我也感謝其他兩位口試委員---徐熊健老師和張雅惠老師。謝謝兩位老師在口試時，給我許多建議和指導，使我的論文能夠修飾得更好，呈現得盡善盡美。同時也感謝您們，對於我的研究主題的讚賞與肯定，相信在接下來的研究當中，我能夠更有信心並且表現亮眼!

然後，我要感謝本系教授項潔老師，以及哲學系教授楊金穆老師。謝謝兩位老師時常撥空與我談話，推薦與我研究相關的書籍外，還不厭其煩地解答我的困惑，使我能夠在數理邏輯的認識上更進一步。這篇論文能夠完成，多虧兩位老師的協助!

另外，感謝本系宋彥朋同學，謝謝你推薦給我的相關書籍，這對我的研究方向有著關鍵性的影響! 還有林佳慶同學，謝謝你常與我分享討論演算法! 也感謝本實驗室已畢業成員洪智鐸學長，以及哲學系學弟黃煜翔同學，與兩位的談話時常激發出我一些新的想法，對我的研究有很大的幫助! 兩位的聰明機智、反應靈敏也令我印象深刻。

碩士生涯的這兩年，在 ACB 實驗室這個大家庭所有成員的協助下，我才能夠有今日的成果。謝謝 Roger 學長、大秉學姐還有明江學長，謝謝你們與我分享彼此的心事! 謝謝安強學長，謝謝你常常舉辦康樂活動，讓我在研究之餘能夠有健康的娛樂(爬山旅遊等); 也謝謝冠宇學長還有秋芸學妹，謝謝你們常常與我一同參與安強學長的活動! 謝謝 wasabe 學姐，謝謝妳常提供我許多協助並且邀請我一起品嘗美食! 謝謝偉哥還有琨哥兩位學長，跟兩位聊一些有趣話題的時候很歡樂! 謝謝蔚茵學妹，謝謝妳時常和我討論以及分享數學話題，妳的想法常常讓我覺得很特別! 謝謝容任學長、佑任學長還有稚穎學弟，雖然我和你們的接觸時間不多，不過我總是感受到你們的親切，謝謝你們平常給我的一些幫忙! 也謝謝小孟同學，你很 cool 也很有能力。我總是粗心大意，謝謝你時常替我處理許多大問題，很高興能夠跟你成為同學!

最後，我要感謝我的父母家人、我的忘年之交巧銀，還有好友于倫。謝謝你們一直以來對我的支持還有關心，並且在我遭遇重大挫折時，鼓勵我並且給我溫暖和照顧。你們是我背後最大的動力來源! 也感謝曾經陪伴我、幫助我還有愛我的朋友們!

中文摘要

邏輯是數學裡專門探討敘述的推理演繹的一個分支，被認定為推理的研究。由於數學的本質是由關於數學物件的敘述以及驗證這些敘述的證明所構成，因此，整個數學領域可以用邏輯加以分析。而理論電腦科學的核心部分---演算法，其觀念的基礎就是計算的觀念，也是個數學物件，可由邏輯分析之。在這篇論文裡，我們提供了邏輯的基本性質，並且利用這些性質來研究一些計算問題，特別是漢彌爾頓路徑(Hamiltonian Path)這個圖型理論的問題。

關鍵字：命題邏輯(propositional logic)，述詞邏輯(predicate logic)，一階邏輯(first-order logic)，存在性二階邏輯(existential second-order logic)，漢彌爾頓路徑。

Abstract

Logic is a branch of mathematics that investigates the deductions about statements and is recognized as the study of reasoning. Because of this, the whole mathematics can be investigated by logic and is even governed by it since the essentials of mathematics consist of statements about mathematical objects and the proofs that verify these statements. Since the underlying concept of algorithms, the critical part of theoretical computer science, is that of computation, which is also a mathematical object, it can aslo be analyzed by logic. In this thesis we provide the basic properties of logic, and then use them to investigate some computational problems, especially the graph-theoretic problem HAMILTONIAN PATH.

Key Words: Propositional logic, predicate logic, first-order logic, existential second-order logic, Hamiltonian path.

Contents

Chapter 1

Introduction

1.1 Prolog

Logic has long been considered the study of reasoning and thoughts, and the study of it can be traced back to the time of Aristotle, a philosopher of ancient Greece, who gave a collection of deduction rules for rationally thinking and governing knowledge. Among these deduction rules, the most well-known is *syllogism*.

It was not until the seventeenth century, however, that the investigations of logic grew mature, through the contributions of many excellent mathematicians and philosophers, among whom especially the German mathematician Leibnitz was the first one to investigate logic after the time of Aristotle, followed by the British mathematician Boole, whose work mainly concerned the so-called *Boolean operations* (which is now stated as *Boolean algebras*), then the American logician Peirce, who introduced *quantifiers*, and the German logician Frege and mathematician Hilbert, who resorted to laying the foundations of mathematics on logic (known as *Hilbert's program*), etc.

In 1930s, the famous Austrian logician Gödel proved an *incompleteness theorem* which put a devastating end to Hilbert's program. On the other hand, the English mathematician Turing proposed the *Halting problem*, which led to the undecidability of first-order logic. Both of these results best demonstrated the fatal flaws of the formal method.

Nevertheless, the progress of the researches in mathematics and logic continued on, and several marvelous results have been discovered in recent decades.

1.2 Categories inside Logic

Because of the wealth of the study of logic, it is often divided into two facets: *mathematical logic* and *philosophical logic*. The former is more relevant to our study of computer science and is further divided into a number of categories: *propostional logic* (or *sentential logic*, *Boolean logic*), *predicate logic* (including first-order, second-order, etc.), both of which are classical studies, and *intuitionistic logic* (whose most apparent feature is the absence of the *law of excluded-middle*, one of the constituents of classical logic), which is the modern trend. We shall always refer to the classical one in this thesis.

There are four theories intimately related to the study of logic: *proof theory*, *model theory*, *set theory* and *recursion theory*. Proof theory and model theory, respectively, represent the studies of the two most fundamental notions about logic — that of *syntax* and of *semantics*.

Set theory was proposed by the German mathematician Cantor. It is by and large considered the most fundamental theory in mathematics since the whole mathematics can be based on the notion of sets.

Finally, in the view point concerning theoretical computer science, recursion theory is usually considered part of *computation theory*, and hence is often called *computability theory*, with the other part of computation theory being *(computational) complexity theory*. The American mathematician Church proposed the so-called *λ-calculus*, which is one of the formal notions of computations, proved to be equivalent to other notions such as Turing machine.

1.2.1 Computational Problems

Logic is powerful in expressing statements in symbolic fashion since its expressibilty is its *raison d'être*. It has the ability to express all the mathematical statements and therefore it is natural to express computational problems in logic.

In computation theory, a problem is said to be in **P** if there is a deterministic Turing machine that decides it in polynomial time, whereas a problem is in **NP** if the same happens except for a nondeterministic Turing machine. The problem $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ is well-known in this field.

Essentially, every computational problem (which can be seen as a set of strings, i.e. a language) can be taken as one in *graph theory*, as the encoding of every string in it can be

regarded as the first row of the adjacency matrix of a graph.

According to Fagin [1], all the graph-theoretic problems in **NP** can be characterized as formulae in *existential second-order logic*. Moreover, Kolaitis and Vardi [2] showed the powerful *zero-one law*, which implies that the set of problems expressible in Horn existential second-order logic is only a restricted proper subset of **P**.

However, with an additional predicate symbol S which is interpreted as the *successor function* in advance, the problems that are precisely in **P** can be characterized as Horn formulae in existential second-order logic. This result was independently discovered in [3, 4, 5, 7]. We shall later give another illustration of the requirement that the predicate symbol of successor function should be given in advance.

Chapter 2

Topics on Propositional Logic

Generally speaking, in the investigations of logic we quite often concern ourselves to those objects called *logical statements*. Informally, a logical statement is one that is declarative, i.e. one that we can decide whether is *true* or *false* — the so-called *truth value*. For example,

1. "$1 + 1 = 2$,"
2. "J. K. Rowling is the author of the series of novels *Harry Potter*," and
3. "D. E. Knuth is one of the greatest computer scientists ever,"

etc.

But more importantly, the truth value of a logical statement or, equivalently, whether it holds or not, should be independent of one's own opinion, i.e. it should be clear and universally accepted. If we re-examine the three examples listed above, it should be clear that the first two are logical statements while the last is not in the sense that whether it is true depends on our own opinions (though almost all people of our time accept it with no doubt).

As mentioned earlier, we restrict our interest to classical logic, one feature of which is *binary* truth value (either true or false, or either 0 or 1 for another notation), whereas in modern logic there is one area called *multivalued logic*, in which truth values between 0 and 1 are allowed.

2.1 Preliminaries

Natural languages, such as everday English or Chinese, abound with plenty of connectives: 'and,' 'or,' 'not,' 'but,' 'if ... then,' 'if ... then ... else,' 'since,' 'unless,' 'while,' etc. Actually, some of them are redundant, for example:

1. "Fried chicken is delicious but unhealthy.": we could say "fried chicken is delicious and fried chicken is not healthy," though that seems somewhat garrulous.

2. "If $\mathbf{P} = \mathbf{NP}$, then there are efficient algorithms to solve those **NP**-complete problems; else they are considered to be intractable.": we could say "if $\mathbf{P} = \mathbf{NP}$, then there are efficient algorithms to solve **NP**-complete problems; else **NP**-complete problems are considered to be intractable."

And at the same time, there is often temporal ordering with the usage of 'and':

1. "The police came in and the thief ran away."

2. "The thief ran away and the police came in."

In such statements, 'and' is in place of 'and then.'

Moreover, we use 'or' sometimes in *inclusive* sense such as "that girl is slim or she is smart," in which one of both constituents of the statement is the case, and the case can be both; other times in *exclusive* sense such as "the manufacturer of this CPU is Intel or AMD," in which one of both constituents of the statement is the case but it cannot be both. Occasionally, we even use 'or' to describe the situation resulted from that some criterion fails to hold, e.g. "you should hurry up or you would miss the bus."

Moreover, some of them somewhat implicitly show our feelings about the events. For example, "that machine seems broken but it works," in which the use of 'but' shows our surprise that the machine in question still works.

As we have seen, these examples illustrate the ambiguities and vagueness that often arise in our daily languages. What we need as a tool in studying mathematics as well as logic, is a formal and artificial language that contains preciseness and exactness while excluding those uncertainties.

Fortunately, for our purpose it is sufficient to use only a restricted part of the connetives mentioned above: 'and,' 'or,' 'not,' 'if ... then,' and additionally 'if and only if.' In order to be precise, in logical statements we shall limit ourselves to the usage of the connectives just mentioned, and it is no loss with this restriction. In particular, we shall stipulate that

1. There is no temporal ordering with 'and';
2. There is no causal relation between the two statements connected by 'or,' it is merely used to indicate that one (or more) statement is the case;
3. 'If ... then' is always used for implications; and finally
4. 'If and only if' is always used for bi-implications.

2.2 Syntax

The language we shall adopt here is *the language of propositional logic*. We shall give precise definitions of terminologies in the following:

Definition 2.1: [10] The *alphabet* of the language of propositional logic is the set that consists of

(a) propositional variables: p_0, p_1, ...;

(b) propositional constants: **true**, **false**;

(c) logical connectives: $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$;

(d) parentheses: (,). $\square$

We shall use the terms "variables," "constants" and "connectives" for short when there are no ambiguities.

In the following we list the name of each connective and the way to pronounce them: [10]

SYMBOL	NAME	PRONUNCIATION
$\wedge$	conjunction	and
$\vee$	disjunction	or
$\neg$	negation	not
$\rightarrow$	implication	if ... then
$\leftrightarrow$	equivalence	... if and only if ...

The objects in this language, *propositions*, are defined recursively in the following:

Definition 2.2: [10]

(a) **true**, **false** and p_n for all $n \in \mathbb{N}$, are propostions;

(b) If φ is a proposition, then so is $\neg\varphi$;

(c) If φ and ψ are propositions, then so are $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ and $(\varphi \leftrightarrow \psi)$. $\square$

In fact, the symbols φ and ψ above are there merely to serve the purpose to denote *some* propositions, i.e. they are *metavariables*. They themselves are *not* propositions, actually. See [10]. In some viewpoint, the language we are investigating here (i.e. the language of propositional logic) is so-called *object language*, and the language we use to describe it (i.e. English) is so-called *metalanguage*. See [6, 10]. The notions of object language and of metalanguage should not be unfamiliar to us: for the study of programming, the object language is of course the programming language itself, while the metalanguage is our daily language; for linguistics such as the research on Latin, the object language is Latin, whereas our daily language (English or some other) plays the role of metalanguage. In fact, the famous *liar's paradox*

"This statement is false,"

which is an instance of *self reference*, arises from our confusion with object languages and metalanguages. This discrimination between them is essential to *theory of truth*, in which field the Polish logician Tarski contributed much. For a more thorough treatment about this, see [8].

On the other hand, the notation we adopt here is in *infix* form, i.e. we place the *binary* connectives (all except '¬,' which is *unary*) between two propositions. In fact, we could adopt the notation in *prefix* form, i.e. binary connetives precede their two arguments, the so-called *Polish notation*. For example, the proposition in our ordinary notation

$$((p_0 \wedge p_1) \rightarrow p_2)$$

would become

$$\rightarrow \wedge p_0 p_1 p_2.$$

Interestingly, in Polish notation parenthese are eliminated. See [8, 9].

Now we are ready to convert those colloquial statements to those in our formal language. For example,

"e is irrational and e is transcendental"

can be converted to

$$(p_0 \wedge p_1),$$

if we use p_0 to represent "e is irrational" and p_1 to represent "e is transcendental."

Actually, "propositions" are just one of the synonyms for "logical statements" in propostional logic, others being "sentence," "Boolean expressions" (named after Boole), etc. Similarly, "propositional logic" is sometimes refered to as "sentential logic" or "Boolean logic." We shall use "propositions" and "propositional logic" in the sequel.

Definition 2.3: [6] A proposition φ is in *conjunctive normal form* iff it is of the form [1]

$$\bigwedge_i C_i,$$

where each C_i is a proposition, called a *clause*, that consists of only (perhaps negated) variables and '$\vee$.' $\square$

[1] By the big '$\bigwedge$' we mean '$\wedge$' applied several times, and there is no ambiguity since '$\wedge$' is associative. (See the table of laws for equivalent propositions listed in subsection 2.3.1.)

Note that in our alphabet the variables are indexed on natural numbers. It is clear that our alphabet is a countable set. And since the set of all propositions are defined as strings over the alphabet according to the formation mentioned above, we have the following lemma:

Lemma 2.1: The set of all propositions is countable. □

2.3 Semantics

So far, we have only defined the syntactic aspect of propositional logic, i.e. its superficial structure. We shall introduce its semantic aspect or its meaning.

Definition 2.4: [6] A *truth assignment* T is a mapping $X \to \{0, 1\}$, where X is a finite subset of the set of all proposition variables $\{p_0, p_1, \ldots\}$. (Recall that 1 and 0 are truth values that represents "true" and "false," respectively.) We say that T is appropriate to φ iff the domain X of T contains all the variables appearing in φ as its elements. □

Definition 2.5: [6] Let T be a truth assignment appropriate to φ. We define the notion of *satisfaction of* T *to* φ (written $T \models \varphi$) inductively as follows:

(a) $T \models$ **true**; $T \not\models$ **false**; [2]

(b) $T \models p_i$:iff $T(p_i) = 1$;

(c) $T \models \neg\psi$:iff $T \not\models \psi$;

(d) $T \models (\psi \wedge \chi)$:iff both $T \models \psi$ and $T \models \chi$;

(e) $T \models (\psi \vee \chi)$:iff $T \models \psi$ or $T \models \chi$ (inclusively);

(f) $T \models (\psi \to \chi)$:iff if $T \models \psi$ then $T \models \chi$;

(g) $T \models (\psi \leftrightarrow \chi)$:iff $T \models \psi$ if and only if $T \models \chi$,

where p_i, ψ, and χ are all consituents of φ. □

[2] Note that **true** and **false** always serve as logical *constants*.

For a more concise definition, we refer the reader to [6].

We are ready to define the *truth value of* φ, specifically, *under a particular truth assignment* T:

Definition 2.6: [6] The truth value of φ under T is 1 if $T \models \varphi$ and 0 otherwise. □

A *truth table* is a sysmatic way to exhaustively list the truth values of a proposition φ, where the truth assignments that involved are those of which the domain consists of precisely the variables appearing in φ.

Example 2.1: The following is the truth table for $((p \to q) \to (\neg p \to \neg q))$:

p	q	$\neg p$	$\neg q$	$(p \to q)$	$(\neg p \to \neg q)$	$((p \to q) \to (\neg p \to \neg q))$
0	0	1	1	1	1	1
0	1	1	0	1	0	0
1	0	0	1	0	1	1
1	1	0	0	1	1	1

□

Observe in the above example that, we have four rows in the truth table. Generally speaking, for a propostion that involves n variables, there are 2^n rows in its truth table. Since each variable takes the value either 0 or 1, there are 2^n possible combinations in total. Furthermore, if we regard a proposition with n variables as a mapping $\{0,1\}^n \to \{0,1\}$ or, formally speaking, *Boolean function*, then there are totally 2^{2^n} possible mappings from $\{0,1\}^n$ to $\{0,1\}$. Thus we have:

Lemma 2.2: There are 2^n rows in the truth table of a proposition with n variables. There are 2^{2^n} n-ary Boolean functions. □

In the following we list the truth tables for $\neg p_0$, $(p_0 \wedge p_1)$ and $(p_0 \vee p_1)$:

1. $\neg p_0$

p_0	$\neg p_0$
0	1
1	0

2. $(p_0 \wedge p_1)$

p_0	p_1	$(p_0 \wedge p_1)$
0	0	0
0	1	0
1	0	0
1	1	1

3. $(p_0 \vee p_1)$

p_0	p_1	$(p_0 \wedge p_1)$
0	0	0
0	1	1
1	0	1
1	1	1

Example 2.2: The following is the truth table of $(p_0 \vee \neg p_0)$:

p_0	$\neg p_0$	$(p_0 \vee \neg p_0)$
0	1	1
1	0	1

□

As shown in the above example, we have all 1's in the last column (which corresponds to the truth values of the proposition $(p_0 \vee \neg p_0)$). This kind of propositions have a special name — *tautologies*.

Definition 2.7: [6] A proposition is a tautology iff it maps all appropriate truth assignments to 1. □

For a tautology φ, we write $\models \varphi$ since $T \models \varphi$ for all T appropriate to it (See [6].). A proposition that maps all appropriate truth assignments to 0 is usually called a *contradiction*. (A contradiction is *equivalent* to the negation of a tautology, — actually, *any* tautology, — see the discussion below.)

2.3.1 Equvalence between Propositions

If we write down the truth tables for both $(p_0 \to p_1)$ and $(\neg p_0 \vee p_1)$ (as shown below:

p_0	p_1	$\neg p_0$	$(p_0 \to p_1)$	$(\neg p_0 \vee p_1)$	$((p_0 \to p_1) \leftrightarrow (\neg p_0 \vee p_1))$
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	1	0	1	1	1

where they share the same table), we find that they are essentially the same mappings (notice the fourth and fifth columns above). In such a case, we say that these propositions are *equivalent*. (We write $\varphi \equiv \psi$ to denote that φ and ψ are equivalent.) More interestingly, if we apply '$\leftrightarrow$' to two equivalent propositions (as we did in the above table, see the last column), then we find that it only has truth value 1 over all rows of the truth table, i.e. it is a tautology (hence the name "equivalence" of the connective $\leftrightarrow$).

Lemma 2.3: Two propositions φ and ψ are equivalent iff $(\varphi \leftrightarrow \psi)$ is a tautology. □

Hence we know that implication '$\to$' can be replaced with '$\neg$' and '$\vee$.' On the other hand, it is easy to verify that equlvalence '$\leftrightarrow$' itself can be replaced with other connectives. For example, $(p_0 \leftrightarrow p_1)$ is equivalent to

$$((p_0 \to p_1) \wedge (p_1 \to p_0)),$$

which in turn is equivalent to

$$((p_0 \wedge p_1) \vee (\neg p_0 \wedge \neg p_1)).$$

It is not hard to see that each Boolean function can be desribed by a proposition which involves only '$\neg$,' '$\wedge$' and '$\vee$,' with constants **true** and **false** replaced by, say, $(p_0 \vee \neg p_0)$ and $(p_0 \wedge \neg p_0)$, respectively. In this situation, we say the set of connectives $\{\neg, \wedge, \vee\}$ is *functionally complete*, i.e. it is sufficient to express all Boolean functions. For a more complete treatment of the notion of functionally complete. See [9].

Lemma 2.4: $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ are functionally complete.

Proof: '$\vee$' is redundant in the sense that $(\varphi \vee \psi)$ is equivalent to $(\neg\varphi \wedge \neg\psi)$. (cf. De Morgan's law.) The other is similar. □

There are two binary connectives (which we eliminate here) that are both functionally complete by themselves: $\mid$ (nand) and $\downarrow$ (nor), i.e. $(\varphi \mid \psi)$ and $(\varphi \downarrow \psi)$ are equivalent to $\neg(\varphi \wedge \psi)$ and $\neg(\varphi \vee \psi)$, respectively.

Laws for Equivalent Propositions

The method of truth table to decide whether two propositions are equivalent works well as we have seen earlier. However, with the growing of the number n of variables appearing in them, this method would become inefficient and physically impractical for lack of space since the *size* (i.e. the total number of rows) of the truth table for a proposition with n variables is in exponential of n (see Lemma 2.2).

If we look deeper into those equivalent propositions, we should find some useful rules. We list some laws for equivalent propositions below: [3]

	FORM	NAME
(1)	$\neg\neg\varphi \equiv \varphi$	law of double neqation
(2)	$\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$	De Morgan's laws
	$\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$	
(3)	$(\varphi \vee \psi) \equiv (\psi \vee \varphi)$	commutative laws
	$(\varphi \wedge \psi) \equiv (\psi \wedge \varphi)$	

[3] Those listed here are taken from [11]. And the name for item (11) (*negation laws*) is undetermined.

	FORM	NAME
(4)	$(\varphi \vee (\psi \vee \chi)) \equiv ((\varphi \vee \psi) \vee \chi)$	associative laws
	$(\varphi \wedge (\psi \wedge \chi)) \equiv ((\varphi \wedge \psi) \wedge \chi)$	
(5)	$(\varphi \vee (\psi \wedge \chi)) \equiv ((\varphi \vee \psi) \wedge (\varphi \vee \chi))$	distributive laws
	$(\varphi \wedge (\psi \vee \chi)) \equiv ((\varphi \wedge \psi) \vee (\varphi \wedge \chi))$	
(6)	$(\varphi \vee \varphi) \equiv \varphi$	idempotent laws
	$(\varphi \wedge \varphi) \equiv \varphi$	
(7)	$(\varphi \vee \mathbf{false}) \equiv \varphi$	identity laws
	$(\varphi \wedge \mathbf{true}) \equiv \varphi$	
(8)	$(\varphi \vee \neg\varphi) \equiv \mathbf{true}$	inverse laws
	$(\varphi \wedge \neg\varphi) \equiv \mathbf{false}$	
(9)	$(\varphi \vee \mathbf{true}) \equiv \mathbf{true}$	domination laws
	$(\varphi \wedge \mathbf{false}) \equiv \mathbf{false}$	
(10)	$(\varphi \vee (\varphi \wedge \psi)) \equiv \varphi$	absorption laws
	$(\varphi \wedge (\varphi \vee \psi)) \equiv \varphi$	
(11)	$\neg\mathbf{true} \equiv \mathbf{false}$	negation laws
	$\neg\mathbf{false} \equiv \mathbf{true}$	

(continued)

By the associative laws, we shall often write $(\varphi \wedge \psi \wedge \chi)$ and $(\varphi \vee \psi \vee \chi)$ instead of $((\varphi \wedge \psi) \wedge \chi)$ and $((\varphi \vee \psi) \vee \chi)$, as no ambiguities should arise.

Example 2.3: The inference rule *modus ponens* [4] is a tautology:

$$\begin{aligned} & ((\varphi \wedge (\varphi \rightarrow \psi)) \rightarrow \psi) \\ \equiv \quad & (\neg(\varphi \wedge (\neg\varphi \vee \psi)) \vee \psi) \\ \equiv \quad & (\neg((\varphi \wedge \neg\varphi) \vee (\varphi \wedge \psi)) \vee \psi) \quad \text{(distributive law)} \end{aligned}$$

[4] See section 2.4

$$
\begin{aligned}
&\equiv (\neg(\mathbf{false} \vee (\varphi \wedge \psi)) \vee \psi) && \text{(inverse law)}\\
&\equiv (\neg(\varphi \wedge \psi) \vee \psi) && \text{(identity law)}\\
&\equiv ((\neg\varphi \vee \neg\psi) \vee \psi) && \text{(DeMorgan's law)}\\
&\equiv (\neg\varphi \vee (\neg\psi \vee \psi)) && \text{(associative law)}\\
&\equiv (\neg\varphi \vee (\psi \vee \neg\psi)) && \text{(commutative law)}\\
&\equiv (\neg\varphi \vee \mathbf{true}) && \text{(inverse law)}\\
&\equiv (\neg\varphi \vee \neg\mathbf{false}) && \text{(negation law)}\\
&\equiv \neg(\varphi \wedge \mathbf{false}) && \text{(DeMorgan's law)}\\
&\equiv \neg\mathbf{false} && \text{(domination law)}\\
&\equiv \mathbf{true} && \text{(negation law) } \square
\end{aligned}
$$

On the other hand, every proposition can be transformed into an equivalent proposition in conjunctive normal form (Definition 2.3). See [6, 10].

2.4 Deduction Systems

As mentioned in the beginning, we recognize logic as a tool for analyzing thoughts and realizing truths. But *what is truth*? Philosophers, and even mathematicians, have sought the answer for centuries. For a satisfactory answer, at least in propositional logic, the concept of truth coincides with that of tautology. If we take a deeper look at tautologies, we conceive that their truth values are 1 (true), independent of any particular truth assignments. Hence it suits perfectly with our feelings about truth — always true.

More precisely, in the process of discovering truth or *deduction*, we usually assume that certain conditions (possibly none), called *premises*, hold. Then we successively get the result or *conclusion* that follows from such assumptions by those justified *inference rules*. Symbolically, it states

$$(\varphi_0 \wedge \ldots \wedge \varphi_{n-1}) \Rightarrow \varphi,$$

where those φ_i, $0 \leq i \leq n-1$, are premises, and φ is the conclusion. The conjuction [5] of those

[5] From the associative laws there is no ambiguity to write propositions in conjuntion without parentheses.

premises is called *hypothesis*. We use '$\Rightarrow$' to denote that φ is deduced from φ_i's.

For a set Δ of propositions, we write $\Delta \models \varphi$ to denote that φ is a conclusion given the propositions in Δ as premises and we say that φ is a *consequence* of Δ.

Actually, given φ_i's and φ, there is a simple method to decide whether the *argument*, i.e. the equation shown above, is correct (in which case we say the argument is *valid*). What we need to do is to show that the implication

$$((\varphi_0 \wedge \ldots \wedge \varphi_{n-1}) \rightarrow \varphi)$$

is a tautology. This method is justified by Tarski's deduction theorem. Intuitively, it is not hard to see the correctness of this method: Since we assume those premises to hold, if the conclusion holds as well, then the truth value of the implication evaluates to 1; if otherwise some premise fails to hold, then the truth value of the implication also evaluates to 1, no matter whether the conclusion holds or not. This reminds us of tautology in some way.

It is not hard to decide whether a given proposition is a tautology, as we have seen: the truth table method, which suffices to decide *all* tautologies. Thus in the viewpoint of computer science, concerning the language TAUTOLOGY that consists of all tautologies in propositional logic, we have

Lemma 2.5: TAUTOLOGY is decidable. □

However, the drawback of the method of truth table is apparent: it is a tedious work and is inefficient and impractical for large number of variables appearing in the proposition. (See Lemma 2.2.)

Fortunately, we are equipped with useful rules discussed earlier (the laws for equivalent propositions), which reduce a great deal of effort of constructing the truth table most of the time, since they can be used to transform a complicated proposition into a simplified equivalent one.

A subtlety is in order: Let an implication χ be equivalent to a proposition ψ. If we added ψ as an additional premise, then the new implication χ' would be a tautology, i.e. the argument would be valid. The reason is clear: Whether χ is valid depends on whether or not the truth value of ψ is 1, if we assume that to be 1 as our additional premise, then the augument is valid.

Similar for the case of disjunction.

Lemma 2.6: Let $((\varphi_0 \wedge \ldots \wedge \varphi_{n-1}) \rightarrow \varphi)$ be an implication equivalent to some proposition ψ. Then $((\varphi_0 \wedge \ldots \wedge \varphi_{n-1} \wedge \psi) \rightarrow \varphi)$ is a tautology.

Proof:

$$
\begin{aligned}
& (((\varphi_0 \wedge \ldots \wedge \varphi_{n-1}) \rightarrow \varphi) \vee \neg\psi) \\
\equiv\ & ((\neg\varphi_0 \vee \ldots \vee \neg\varphi_{n-1} \vee \varphi) \vee \neg\psi) && \text{(DeMorgan's law applied several times)} \\
\equiv\ & ((\neg\varphi_0 \vee \ldots \vee \neg\varphi_{n-1} \vee \neg\psi) \vee \varphi) && \text{(associative law and commutative law)} \\
\equiv\ & (\neg(\varphi_0 \wedge \ldots \wedge \varphi_{n-1} \wedge \psi) \vee \varphi) && \text{(DeMorgan's law applied several times)} \\
\equiv\ & ((\varphi_0 \wedge \ldots \wedge \varphi_{n-1} \wedge \psi) \rightarrow \varphi).
\end{aligned}
$$

□

On the other hand, there is an alternative method which provides 'formal' proofs that are reminiscent of mathematical proofs for valid arguments: *deduction system*. There are four common deduction systems: *axiom system*, *sequent calculus*, *natural deduction* and *analytic tableau*. The method of deduction system is highly syntactic and more machine-oriented, and is suitable for *valid arguments only*, i.e. it cannot determine those invalid arguments. We shall briefy introduce axiom system, which is purely syntactic.

First of all, the term "axiom" should not be strange to us: it is fundamental to the study of mathematics and even physics. Take Euclidean geometry for an example, one of its axioms states "given an (infinitely long) line on a plane, and a point on the same plane but not on this line, there is exactly one line on the same plane that passes through the given point and is parallel to the given line." Another example arises from physics: That "the speed of light in absolute vacuum is a constant, independent of any frame of inertia" is an axiom in Einstein's relativity theory.

Definition 2.8: [11] A proof for an argument

$$(\varphi_0 \wedge \ldots \wedge \varphi_{n-1}) \Rightarrow \varphi$$

is a sequence $\langle \psi_0, \ldots, \psi_m, \psi \rangle$ where

$$\psi := ((\varphi_0 \wedge \ldots \wedge \varphi_{n-1}) \rightarrow \varphi),$$

and each ψ_i, $0 \leq i \leq m$ is an axiom of the system or a proposition generated according to the inference rules of the system. $\square$

(Note that these axioms are just like premises of another argument where the conclusion is the original argument.)

If there is a proof for a proposition φ, then we say φ is a *theorem* and is *derivable* and we write $\vdash \varphi$.

Given a set Δ of propositions, if a proposition φ is derivable regarding those in Δ as additional axioms, then we say φ is a Δ-*theorem* and is *derivable from* Δ and we write $\Delta \vdash \varphi$. The notion $\Delta \vdash \varphi$ is the syntactic counterpart of $\Delta \models \varphi$.

Definition 2.9: [6] A set Δ is consistent iff $\Delta \nvdash$ **false**. $\square$

The following two theorems together state that the notion of consequence and that of derivability coincides in propositional logic:

Theorem 2.1: [soundness theorem], [10] If $\Delta \vdash \varphi$, then $\Delta \models \varphi$. $\square$

Theorem 2.2: [completeness theorem], [10] If $\Delta \models \varphi$, then $\Delta \vdash \varphi$. $\square$

For more on axiom systems, see [12]. For more on soundness and completeness theorems, see [10].

Chapter 3

Topics on Predicate Logic

All is well so far. Propositional logic possesses many good properties as we have seen in previous chapter. But let us examine the implication below, Aristotle's well-known syllogism:

1. (Premise) All humans are mortal.
2. (Premise) Socrates is a human.
3. (Conclusion) Socrates is mortal.

This implication is evidently correct, since the conclusion is just an *instantiation* of the first premise by the second premise. If we symbolize this implication as shown below, however, we will be surprised that it is *not* a tautology, which stands for truth in propositional logic:

$$((p_0 \wedge p_1) \rightarrow p_2),$$

where

(a) p_0: "All humans are mortal,"

(b) p_1: "Socrates is a human,"

(c) p_2: "Socrates is mortal."

We see from this example that there is a flaw in propositional logic: There is no way to express *single objects* (e.g. 'Socrates' in our previous implication), along with their attibutes

such as states and relations. Thus, we must enlarge our language with some additional elements to encompass such statements — hence *predicate logic*. In the next section, we shall introduce the simplest part, *first-order logic*.

3.1 First-order Logic

First-order logic is more complicated than propositional logic. Informally, First-order logical statements (so-called *formulae*) are similar to propositions: They can be decided to whether hold or not (just like propositions can be decided to be whether true or false) and can be connected by logical connectives just like propositions. But additionally, they can be preceded by the so-called *quantifiers* ('$\forall$' and '$\exists$,' hence we sometimes refer to predicate logic as *quantificational logic*) and, more importantly, the basic parts of them, i.e. the *atomic formulae* (analogous to propositional variables), consist of statements about single objects — a combination of *predicate symbol* and *terms*, — which in contrast are the key improvements on propositional logic. (Recall that propositional variables cannot be further divided into smaller parts.)

3.1.1 Syntax

Definition 3.1: [6, 8] The alphabet of the language of first-order logic is the set that consists of

(a) first-order variables: v_0, v_1, ...;

(b) (i) a (possibly empty) set K of constant symbols: c_0, c_1, ...;

(ii) a (possibly empty) set Φ_n of n-ary function symbols for each $n \in \mathbb{Z}^+$;

(iii) a (possibly empty) set Π_n of n-ary relation symbols for each $n \in \mathbb{Z}^+$;

(c) quantifiers: $\forall$, $\exists$;

(d) logical connectives: $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$;

(e) parentheses: (,). $\square$

We shall denote $\Phi := \bigcup_{n\in\mathbb{Z}^+} \Phi_n$ and $\Pi := \bigcup_{n\in\mathbb{Z}^+} \Pi_n$. Note that $\Pi \neq \emptyset$ as the equality symbol '=' (a binary relation symbol) is a member of Π. Relation symbols are often called predicate symbols, or predicate for short. And some regard constants as *nullary* functions. Also, the language of first-order logic as well as of others in predicate logic, may or may not contain the binary predicate '=,' depending on the topics we are speaking about. (Predicate logic without '=' is sometimes called *specialized predicate logic*, whereas that with it is sometimes callled *generalized logic*. For more topics on this, see [9, 10].) We shall include it in our language.

Definition 3.2: [8]

(a) Each variable (v_0, v_1, ..., or x, y, ...) is a term;

(b) Each constant symbol is a term;

(c) If $t_0, \ldots, t_{n-1}$ are terms and f is an n-ary function symbol, then so is $ft_0 \ldots t_{n-1}$. □

We shall often write $f(t_0, \ldots, t_{n-1})$ for $ft_0 \ldots t_{n-1}$.

Definition 3.3: [8]

(a) If $t_0, \ldots, t_{n-1}$ are terms and R is an n-ary relation symbol, then $Rt_0 \ldots t_{n-1}$ is an (atomic) formula;

(b) If φ is a formula, then so is $\neg\varphi$;

(c) If φ and ψ are formulae, then so are $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$ and $(\varphi \leftrightarrow \psi)$;

(d) If φ is a formula and x is a variable, then $\forall x\varphi$ and $\exists x\varphi$ are also formulae. □

In the following, we list the name of each quantifier and the way to pronounce them: [10]

SYMBOL	NAME	PRONUNCIATION
$\forall$	universal quantifier	for all, for any, ...
$\exists$	existential quantifier	there exists ... such that, there is ... such that, ...

We shall often write $R(t_0, \ldots, t_{n-1})$ for $Rt_0 \ldots t_{n-1}$. In particular, for binary predicate, we often write $t_0 R t_1$ (*infix form*) instead of $Rt_0 t_1$ (prefix form). Note that these alternative forms of terms and formulae are just for ease of reading, by definition they are not terms and formulae, respectively.

The polish notation applies to first-order logic (actually, predicate logic) as well, just regard each atomic formula or *quantified formula* (those that are preceded by quantiers) as a propositional variable.

Now we are ready to convert those colloquial statements to ones in our formal language. For example,

"For every pair x, y, if $x < y$ then $f(x) < f(y)$."

can be converted to

$$\forall x \forall y (x < y \rightarrow f(x) < f(y)).$$

We say an *occurrence* of a variable x is *bound* in φ iff it is *quantified*, i.e. there is '$\forall x$' or '$\exists x$' that applies to it; otherwise it is called *free*. More precisely, in the following formula:

$$(\forall x x = x \wedge x = f(y)),$$

the first two occurrences of x (in '$x = x$') are bound (we do not recognize the x in '$\forall x$' as an occurrence of x as it is part of the package '$\forall x$'), whereas the third occurrence of x (in '$x = f(y)$') is free. Note that quantifiers only apply to the formula that immediately follows. Likewise, the y in the above formula is also free. A variable is bound if it has no free occurrences, otherwise it is free. A formula without free variable is called a *sentence*. Since x and y are free in the above formula, i.e. they have free occurrences, this formula is not a sentence.

Definition 3.4: [6] A formula φ is in *prenex normal form* iff it is of the form

$$Q_0 \ldots Q_{n-1} \psi,$$

where Q_i's are packages of the form '$\forall x$' or '$\exists x$' and ψ is a quantifier-free formula, called the *matrix* of the formula φ. $\square$

It is clear that our new alphabet is also countable, though we add some elements to it. Since the set of all formulae are defined as strings over the alphabet according to the formation mentioned above, we have the following lemma, analogous to Lemma 2.1:

Lemma 3.1: The set of all formulae is countable. □

3.1.2 Semantics

Until now, terms and formulae are merely strings (of special kinds) over the alphabet. We shall introduce their meanings. The semantic aspect of predicate logic is somewhat complicated.

First, we are given a nonempty set D, called *domain* (or *universe*), of which we map each variable x in our alphabet to some element. It is similar to $\{0, 1\}$ given in propositional logic. Furthermore, we map each constant, function symbol and relation symbol to actual element, function and relation, respectively, over the domain D, except that '=' is always mapped to $\{(e, e)|e \in D\}$. We shall denote the mapping from variables to elements in D (called an *assignment*) as β, and denote the mapping from constants, function symbols and relation symbols to those actual objects over D (called a *structure*) as $\mathfrak{A}$. We shall often write $f^{\mathfrak{A}}$ (or f^D), $c^{\mathfrak{A}}$ (or c^D), and $R^{\mathfrak{A}}$ (or R^D) instead of $\mathfrak{A}(f)$, $\mathfrak{A}(c)$ and $\mathfrak{A}(R)$, respectively. We have the following definition:

Definition 3.5: [8] An *interpretation* $\mathfrak{I}$ for a given domain D is a pair $(\mathfrak{A}, \beta)$ that consists of a structure $\mathfrak{A}$ and an assignment β. □

Intuitively, an interpretation in predicate logic is the counterpart of a truth assignment in propositional logic. For the meanings of terms, given an interpretation $\mathfrak{I}$, we have the following definition:

Definition 3.6: [the meaning of a term], [6]

(a) For a variable x, $\mathfrak{I}(x) := \beta(x)$;

(b) For a constant c, $\mathfrak{I}(c) := c^{\mathfrak{A}}$;

(c) For an n-ary function symbol f applied to n terms $t_0, \ldots, t_{n-1}$,
$\mathfrak{I}(ft_0 \ldots t_{n-1}) := f^{\mathfrak{A}}(\mathfrak{I}(t_0), \ldots, \mathfrak{I}(t_{n-1}))$. □

Hence the meanings of terms and atomic formulae (for $Rt_0 \dots t_{n-1}$, its meaning is thus $R^{\mathfrak{A}}(\mathfrak{I}(t_0), \dots, \mathfrak{I}(t_{n-1}))$, given $\mathfrak{I}$) are well-defined. Just as in propositional logic, we shall define the notion of satisfaction of $\mathfrak{I}$ to a formula φ below:

Definition 3.7: [8]

(a) For atomic formula $Rt_0 \dots t_{n-1}$, $\mathfrak{I} \models Rt_0 \dots t_{n-1}$:iff $R^{\mathfrak{A}}(\mathfrak{I}(t_0), \dots, \mathfrak{I}(t_{n-1}))$;
(Note that we stipulate that "$=^{\mathfrak{A}} := \{(e, e) | e \in D\}$," so "$\mathfrak{I} \models t_0 = t_1$:iff $\mathfrak{I}(t_0) = \mathfrak{I}(t_1)$.")

(b) $\mathfrak{I} \models \neg\varphi$:iff $\mathfrak{I} \not\models \varphi$;

(c) $\mathfrak{I} \models (\varphi \wedge \psi)$:iff both $\mathfrak{I} \models \varphi$ and $\mathfrak{I} \models \psi$;

(d) $\mathfrak{I} \models (\varphi \vee \psi)$:iff $\mathfrak{I} \models \varphi$ or $\mathfrak{I} \models \psi$ (inclusively);

(e) $\mathfrak{I} \models (\varphi \rightarrow \psi)$:iff if $\mathfrak{I} \models \varphi$ then $\mathfrak{I} \models \psi$;

(f) $\mathfrak{I} \models (\varphi \leftrightarrow \psi)$:iff $\mathfrak{I} \models \varphi$ if and only if $\mathfrak{I} \models \psi$.

(g) $\mathfrak{I} \models \forall x\varphi$:iff for all $e \in D$, $\mathfrak{I}_{x \rightarrow e} \models \varphi$; [1]

(h) $\mathfrak{I} \models \exists x\varphi$:iff there exists $e \in D$ such that $\mathfrak{I}_{x \rightarrow e} \models \varphi$. □

An interpretation $\mathfrak{I}$ is said to be a *model* of a formula φ iff $\mathfrak{I} \models \varphi$. Moreover, for a set Δ of formulae, $\mathfrak{I}$ is a model of Δ (written $\mathfrak{I} \models \Delta$) iff $\mathfrak{I} \models \varphi$ for every $\varphi \in \Delta$.

If for Δ and φ, every interpretation $\mathfrak{I}$ which is a model of Δ ($\mathfrak{I} \models \varphi$) is also a model of φ ($\mathfrak{I} \models \varphi$), then we write $\Delta \models \varphi$ and say that φ is a *consequence* of Δ. Specifically, if $\emptyset \models \varphi$, i.e. φ is satisfied by all interpretations, then we write $\models \varphi$ and say φ is a *valid formulae* (a situation analogous to tautologies in propositional logic). If a formula φ is unsatisfiable, i.e. it has no models, then it is equivalent to the negation of a valid formula.

Example 3.1: Let $K := \emptyset$, $\Phi := \emptyset$ and $\Pi := \{R, =\}$, where R is binary. Let $\mathfrak{I}_0$ and $\mathfrak{I}_1$ be two interpretations such that:

[1] $\mathfrak{I}_{x \rightarrow e}$ is just $\mathfrak{I}$ with the variable x mapped to e

	$\mathfrak{I}_0$	$\mathfrak{I}_1$
domain	$\{0, 1, 2\}$	$\{1, 2, 3\}$
x	0	1
y	1	2
z	2	3
R	$\{(0,1), (0,2), (1,2)\}$	$\{(1,2), (1,3), (2,3), (3,2), (3,1), (2,1)\}$

(The mappings of other variables are irrelevant to this example.)

Let

$$\varphi_0 := R(x, y),$$

and

$$\varphi_1 := \forall x \forall y (\neg x = y \rightarrow R(x, y)),$$

then we have:

	Is $\mathfrak{I}_0$ a model?	Is $\mathfrak{I}_1$ a model?
φ_0	Yes	Yes
φ_1	No	Yes

□

Equivalent Formulae

Analogous to propositional logic, we have the following definition:

Definition 3.8: [8] Two formulae φ and ψ are equivalent (written: $\varphi \equiv \psi$) iff $\{\varphi\} \models \psi$ and $\{\psi\} \models \varphi$. □

The laws for equivalent formulae are the same to those for equivalent propositions (without, of course, those that involve **true** or **false**), regarding a formulae as a proposition. And additionally, [6, 10]

FORM	NOTE
$\neg\forall x\varphi \equiv \exists x\neg\varphi$	generalized DeMorgan's law for '$\forall$'
$\neg\exists x\varphi \equiv \forall x\neg\varphi$	generalized DeMorgan's law for '$\exists$'
$\forall x(\varphi \wedge \psi) \equiv (\forall x\varphi \wedge \forall x\psi)$	-
$\forall x(\varphi \wedge \psi) \equiv (\forall x\varphi \wedge \psi)$	x does not appear free in ψ
$\forall x(\varphi \vee \psi) \equiv (\forall x\varphi \vee \psi)$	x does not appear free in ψ
$\forall x\varphi \equiv \forall y\varphi[x \leftarrow y]$	y does not appear in φ

($\varphi[x \leftarrow y]$ is φ with those free occurrences of x replaced by occurrences of y) We see that '$\forall$' and '$\exists$' are similar to '$\wedge$' and '$\vee$,' respectively.

On the other hand, every formula can be transformed into an equivalent formula in prenex normal form (cf. Definition 3.4). See [6, 10].

3.1.3 Deduction Systems

As mentioned in section 2.4, there are four common methods to deduce conclusions given some premises. Note that the method of truth table does not work here, since there are infinitely many possible interpretations. (Recall that in propositional logic, there are only finitely many truth assignments that matter.)

The following definitions are all analogous to those in propositional logic:

Definition 3.9: A proof for an argument[2]

$$(\varphi_0 \wedge \ldots \wedge \varphi_{n-1}) \Rightarrow \varphi$$

is a sequence $\langle \psi_0, \ldots, \psi_m, \psi \rangle$ where

$$\psi := ((\varphi_0 \wedge \ldots \wedge \varphi_{n-1}) \rightarrow \varphi),$$

and each ψ_i, $0 \leq i \leq m$ is an axiom of the system or a formula generated according to the inference rules of the system. $\square$

[2]The argument for first-order logic is defined similarly.

(Note that these axioms are just like premises of another argument where the conclusion is the original argument.)

If there is a proof for a formula φ, then we say φ is a *first-order theorem* and φ is *derivable*, and we also write $\vdash \varphi$.

Given a set Δ of formulae, if a formula φ is derivable regarding those in Δ as additional axioms, then we say φ is a Δ-*first-order theorem* and φ is *derivable from* Δ, and we also write $\Delta \vdash \varphi$. Note that a theorem φ is also a Δ-first-order theorem by definition. The set Θ of all Δ-first-order theorems φ (which are sentences) is called a *theory*, given Δ. The notation $\Delta \vdash \varphi$ is the syntactic counterpart of $\Delta \models \varphi$.

There is an interesting application to the research of *artificial intelligence*: the *knowledge base* plays the role of Δ, while φ denotes the representation of some knowledge. The process for the knowledge base to deduce φ (reasoning) is indeed the same as that for $\Delta \vdash \varphi$. For more on this issue, see [20].

Definition 3.10: [8] A set Δ is consistent iff there is no contradiction φ such that $\Delta \vdash \varphi$. □

The axiom system given in [6] is shown below:

ITEM	FORM
AX0	Any formula whose propositional form is a tautology.
AX1	Any formula of the following forms:
	AX1a $t = t$, where t is a term.
	AX1b $((t_0 = t'_0 \wedge \ldots \wedge t_{n-1} = t'_{n-1}) \rightarrow ft_0 \ldots t_{n-1} = ft'_0 \ldots t'_{n-1})$, where t_i's are terms and f is an n-ary function symbol.
	AX1c $((t_0 = t'_0 \wedge \ldots \wedge t_{n-1} = t'_{n-1}) \rightarrow (Rt_0 \ldots t_{n-1} \rightarrow ft'_0 \ldots t'_{n-1}))$, where t_i's are terms and R is an n-ary relation symbol.
AX2	Any formula of the form $(\forall x\varphi \rightarrow \varphi[x \leftarrow t])$.
AX3	Any formula of the form $(\varphi \rightarrow \forall x\varphi)$, with x not free in φ.
AX4	Any formula of the form $(\forall x(\varphi \rightarrow \psi) \rightarrow (\forall x\varphi \rightarrow \forall x\psi))$.

($\varphi[x \leftarrow t]$ is φ with those free occurrences of x replaced by occurrences of the term t). The formulae shown above are basic axioms. The axiom system contains besides these all those formulae that are preceded by any number of prefixes of the form '$\forall x$.' It is not hard to see that the axioms are valid formulae. The only one inference rule is modus ponens. (See Example 2.3.) Schematically, it states:

$$\frac{\varphi \qquad (\varphi \rightarrow \psi)}{\psi}.$$

Here we give an example illustrating the proof under axiom systems:

Example 3.2: The formal proof of the theorem $\exists y(x = y)$ is:

$\psi_0 := x = x$

(an axiom from group **AX1a**),

$\psi_1 := (\forall y \neg x = y \rightarrow \neg x = x)$

(an axiom from group **AX2**),

$\psi_2 := ((\forall y \neg x = y \rightarrow \neg x = x) \rightarrow (x = x \rightarrow \exists y x = y))$

(an axiom from group **AX0**),

$\psi_3 := (x = x \rightarrow \exists y x = y)$

(from ψ_1 and ψ_2 by *modus ponens*),

$\psi_4 := \exists y x = y$

(from ψ_0 and ψ_3 by *modus ponens*). $\square$

In proving a theorem, we often divide it into two or more parts. The correctness of this technique can be justified by the following lemma. (Note that we prove it in the level of metalanguage.)

Lemma 3.2: Let φ and ψ be two formulae. Then

$$(\vdash \varphi \text{ and } \vdash \psi) \text{ if and only if } \vdash (\varphi \wedge \psi).$$

Proof: Suppose that $\vdash \varphi$ and $\vdash \psi$, i.e. there are proofs S_φ and S_ψ for them. Notice that φ and ψ are the last elements of S_φ and S_ψ, respectively. Then $\langle S_\varphi, S_\psi, (\psi \rightarrow (\varphi \rightarrow (\varphi \wedge \psi))), (\varphi \rightarrow (\varphi \wedge \psi)), (\varphi \wedge \psi)\rangle$ is a proof for $(\varphi \wedge \psi)$. Therefore $\vdash (\varphi \wedge \psi)$.

Conversely, suppose that there is a proof $S_{(\varphi \wedge \psi)}$ for $(\varphi \wedge \psi)$. Then $\langle S_{(\varphi \wedge \psi)}, ((\varphi \wedge \psi) \rightarrow \varphi), \varphi \rangle$ and $\langle S_{(\varphi \wedge \psi)}, ((\varphi \wedge \psi) \rightarrow \psi), \psi \rangle$ are proofs for φ and ψ, respectively. Hence $\vdash \varphi$ and $\vdash \psi$. $\square$

Analogous to propositional logic, the following two theorems together state that the notion of consequence and that of derivability coincides in first-order logic:

Theorem 3.1: [soundness theorem], [6] If $\Delta \vdash \varphi$, then $\Delta \models \varphi$. $\square$

Theorem 3.2: [Gödel's completeness theorem], [6] If $\Delta \models \varphi$, then $\Delta \vdash \varphi$. $\square$

The following theorem, which is a consequence of Gödel's completeness theorem, is critical to our main result (Theorem 4.3):

Theorem 3.3: [Löwenheim-Skolem theorem], [6] If a sentence φ has finite models of arbitrary large cardinality (i.e. the *size* of the domain), then it has an infinite model. $\square$

For more complete treatment about this, see [6, 8, 10].

For more on the axiom system, see [6]. [8, 10] are excellent references for other deduction systems. [6, 8, 10] all discuss the theorems above. (However, the proofs of Theorem 3.2 in all of them adopt the one by [13]. For Gödel's own proof, see [14].)

3.1.4 Weakness of First-Order Logic

Note that first-order variables are only mapped to the simplest or the individual objects (hence the name *first-order*) in a structure, whereas the concepts such as functions and relations are objects in second-order. Because of this, there are many (computational) problems that cannot be expressed in first-order logic (our result in the next chapter, for instance). It is the need to represent these complex objects that provokes second-order logic.

3.2 Second-Order Logic

In fact, second-order logic is very much similar to first-order logic, except for addtional variables — so-called the *second-order* variables, — which map to relations over the domain.

A formula in second-order is the same as in first-order, except that it can be formed using second-order variable P as ordinary relation symbol in the alphabet (e.g. $Pt_0 \dots t_{n-1}$, where P is an n-ary second-order variable) and that second-order variables can be quantified in it (e.g. $\forall P(t_0 = t_{n-1} \to \neg Pt_0 \dots t_{n-1})$ or $\exists X X t_0 t_1$). In this aspect, the relation symbols in the alphabet can be regarded as *second-order constants*.

A specialized part of second-order logic — *existential second-order logic* — concerns only those formulae of the form

$$\exists P \varphi,$$

where P is a second-order variable, and φ is a first-order formula regarding P as given in the alphabet. We shall focus our effort to this in the next chapter.

3.3 Remarks

In the view point of algebra, propositional logic is characterized by *Boolean algebras*, whereas predicate logic is characterized by *cylindric algebras*. [16]

Gödel's incompleteness theorem [15] states that there is some statement about elementary arithmetic that is neither provable nor refutable provided that our axiom system is consistent, which manifests the limit of our formal method. This is a devastating result to Hilbert's program, of which the ultimate goal is to totally axiomatize mathematics. [6, 8, 10] are all good references for this topic.

Chapter 4

Graph-Theoretic Problems as Expressed in Logical Formulae

4.1 Prolog

Graphs are a widely-used data structure in computer science, and they have many applications. The first use of graphs dates back to 1736, when the Swiss mathematician Euler used them to solve the problem of seven bridges of Königsberg, thereby built the foundations of *graph theory*.

Definition 4.1: [17] A *directed graph* G is a pair (V, E), where V is a finite set and E is a binary relation defined over V. V is called the *set of vertices* of G (its elements are called *vertices*) and E is called the *set of edges* of G (its elements are called *edges*).

Note that there is another kind of graphs called *undirected graphs*, in which E contains subsets of cardinality exactly 2 of V as its elements. However, we shall focus on directed graphs. In the sequel we use *graph* for short.

A *subgraph* $G' = (V', E')$ of $G = (V, E)$ is a graph (hence $E' \subset V' \times V'$) such that $V' \subset V$ and $E' \subset E$. If (u, v) is an edge of G, then we say that (u, v) is *incident from* u and it is *incident to* v and v is *adjacent to* u. The *in-degree* of a vertex v is the number of edges incident to v, while its *out-degree* is the number of edges incident from it.

A *path* from a vertex u to a vertex v is a sequence $\langle v_0, \ldots, v_n \rangle$ of vertices in V such that $v_0 = u$, $v_n = v$, and there is an edge in the graph for each pair of consecutive vertices in the sequence. A path is *simple* if each vertex in this path appears exactly once. If there is a path from u to v, we say that v is *reachable from* u. A directed graph is *strongly connected* if every vertex in the graph is reachable from others.

A *cycle* is a path such that the first and the last vertices in the path coincide and there is at least one vertex other than the first and the last ones in this path.

For more thorough treatment, see [11, 17, 18].

Hamiltonian Path

A problem "Does a graph contain a Hamiltonian path?" which is named after the Irish mathematician Hamiltonian is defined below:

Definition 4.2: [17] A *Hamiltonian path* of a graph $G = (V, E)$ is a simple path in which exactly all vertices in V appear. The decision problem HAMILTONIAN PATH asks, given a graph G, whether it contains a Hamiltonian path. □

HAMILTONIAN PATH is a well-known **NP**-complete problem. [6, 17] For more on the properties and examples of Hamiltonian path, see [11].

4.2 The Successor Function in Graph-Theoretic Problems

Let us return back to the perspectives of logic. In general, the alphabet concerning graph-theoretic problems is: $K_G := \emptyset$, $\Phi_G := \emptyset$ and $\Pi_G := \{G, =\}$, where G is a binary relation symbol which is intuitively interpreted as "edges" in a graph, and $=$ is a binary relation symbol which is "always" interpreted as the equality relation. (See Subsection 3.1.2.)

If we regard the set of vertices V of a graph $G = (V, E)$ with n vertices as the set of numbers $\{0, 1, \ldots, n-1\}$, then the problem HAMILTONIAN PATH can be expressed by the following existential second-order formula, where P is isomorphic to the usual *is-less-than* relation ('$<$')

over $\{0, 1, \ldots, n-1\}$: [6]

$$\exists P\varphi,$$

where φ is the conjuction of the following three formulae:

$$\forall x \forall y (P(x, y) \vee P(y, x) \vee x = y),$$

$$\forall x \forall y \forall z (\neg P(x, x) \wedge ((P(x, y) \wedge P(y, z)) \rightarrow P(x, z))),$$

and

$$\forall x \forall y ((P(x, y) \wedge \forall z (\neg P(x, z) \vee \neg P(z, y))) \rightarrow G(x, y)).$$

Moreover, if we added to φ in conjunction the additional formula

$$\forall x \forall y (\forall z (\neg P(x, z) \wedge \neg P(z, y)) \rightarrow G(x, y)),$$

then the resulting existential second-order formula would characterize the problem HAMILTONIAN CYCLE.

The following two theorems are critical:

Theorem 4.1: [Fagin's theorem], [6] The class of all graph-theoretic properties expressible in existential second-order logic is **NP**. □

A formula $\exists P\varphi$ is said to be in *Horn existential second-order logic* iff φ is in prenex normal form with only universal first-order quantifiers (i.e. '$\forall$' that only applies to first-order variables), and its matrix is in conjunctive normal form where each clause contains at most one unnegated atomic formula that involves the second-order variable P. *Horn existential second-order with successor* is that with an additional binary relation symbol S given in our alphabet, i.e. $\Pi_G := \{G, S, =\}$, and additionally S is interpreted in advance as the successor function, i.e. a linear ordering over the vertices in a graph. More precisely, $S(x, y)$ states that vertex y is the successor of vertex x, and S is isomorphic to $\{(0, 1), (1, 2), \cdots, (n-2, n-1)\}$ for a graph with n vertices.

Theorem 4.2: [6] The class of all graph-theoretic properties expressible in Horn existential second-order logic with successor is precisely **P**. □

The reason of the requirement that S should be given in advance is according to [2], and might be well illustrated by our main result in the following (the proof technique is very much similar to that adopted in Corollary 6 to Theorem 5.7 from [6]):

Theorem 4.3: Let $\exists S\varphi$ be an existential second-order formula describing that S is the successor function. Then the part φ cannot be in first-order logic.

Proof: Suppose, for the sake of contradiction, that there were such a first-order formula φ. Then, it must be a *sentence*, i.e. one that has no free (first-order) variables. The reason is clear: the subject of this formula is the relation S itself and this description applies to *all* pairs of vertices in the graph.

Next, consider the following first-order sentence:

$$\psi := \varphi \wedge \varphi_0 \wedge \varphi_1 \wedge \varphi_2,$$

where

$$\varphi_0 := \exists x \forall y \neg S(x, y)$$

states that there is a vertex x that has no successor, i.e. the "last" vertex exists,

$$\varphi_1 := \forall x \forall y((S(x, y) \vee \forall z(\neg S(x, z) \wedge \neg S(z, y))) \rightarrow G(x, y))$$

states that there is an edge from vertex x to vertex y if *either* y is the successor of x *or* y is the first vertex (vertex 0) and x is the last vertex (vertex $n - 1$), and

$$\varphi_2 := \forall x \forall y(G(x, y) \rightarrow (S(x, y) \vee \forall z(\neg S(x, z) \wedge \neg S(z, y))))$$

states the converse of φ_1. (φ_1 and φ_2 together states that there is an edge from vertex x to vertex y *if and only if* either y is the successor of x or y is the first vertex and x is the last vertex.)

ψ states that the graph G is a cycle itself, and ψ has arbitrarily large finite models. In other words, there are finite cycles that have as many vertices as desired. According to Löwenheim-Skolem (cf. Theorem 3.3), the sentence ψ has an infinite model, call it G_∞.

If we start from vertex 0 and follow the edge out of it, we reach vertex 1; and then follow the edge out of it we reach vertex 2; ..., in this way, we will eventually meet all vertices reachable from vertex 0.

But since the graph is strongly connected, those vertices previously described include all vertices in the graph. At the same time, the graph is a cycle, so there must be a vertex, numbered j, from which there is an edge to vertex 0. This is obviously a contradiction! The

asserted "infinite" cycle is indeed finite. □

Thus the formula expressing the successor function is

$$\exists S \exists P \varphi,$$

where φ is the conjuction of the following three formulae:

$$\forall x \forall y (P(x,y) \vee P(y,x) \vee x = y),$$

$$\forall x \forall y \forall z (\neg P(x,x) \wedge ((P(x,y) \wedge P(y,z)) \rightarrow P(x,z))),$$

and

$$\forall x \forall y ((P(x,y) \wedge \forall z (\neg P(x,z) \vee \neg P(z,y))) \leftrightarrow S(x,y)).$$

Note that it is similar to the formula that expresses HAMILTONIAN PATH.

On the other hand, HAMILTONIAN PATH can in turn be expressed in existential second-order logic *with successor*:

$$\exists \pi (\varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \psi),$$

where

$$\varphi_0 := \forall x \exists x' \pi(x, x'),$$

$$\varphi_1 := \forall x \forall x' \forall x'' ((\pi(x,x') \wedge \pi(x,x'')) \rightarrow x' = x''),$$

(φ_0 and φ_1 together state that π is a *function*.)

$$\varphi_2 := \forall x \exists x' \pi(x', x),$$

(Additionally, φ_2 states that the function π defined by φ_0 and φ_1 is surjective. Hence these three formulae together state that π is a *permutation*, since a surjective function defined over a finite set is a permutation over it.) and

$$\psi := \forall x \forall x' \forall y \forall y' ((\pi(x,x') \wedge \pi(y,y') \wedge S(x',y')) \rightarrow G(x,y)).$$

At this point, ψ says that after the renumbering of all vertices according to the permutation π defined by φ_0, φ_1 and φ_2, there is an edge between every pair of vertices with consecutive numbers, hence a Hamiltonian path exists.

Alternatively, φ_2 can be replaced by

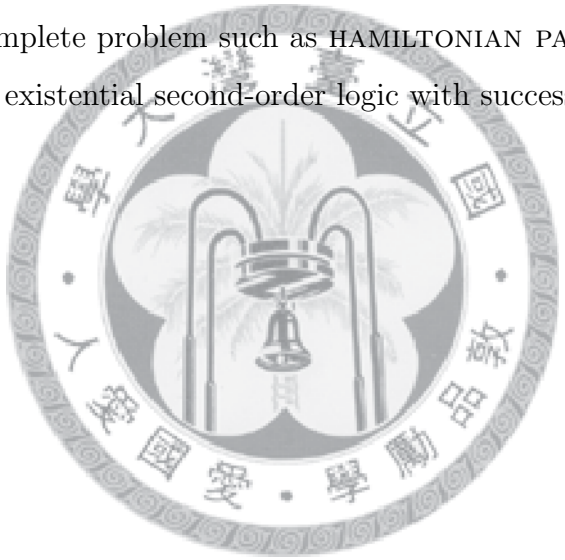
$$\varphi_2' := \forall x \forall x' \forall y \forall y' ((\pi(x, x') \wedge \pi(y, y') \wedge x' = y') \rightarrow x = y),$$

which states that the function π defined by φ_0 and φ_1 is injective. Again, all these three formulae together says that π is a permutation, since an injective function defined over a finite set is a permutation over it. Furthermore, if we added to ψ in conjunction the additional formula

$$\psi' := \forall x \forall x' \forall y \forall y' (\forall z (\pi(x, x') \wedge \pi(y, y') \wedge \neg S(x', z) \wedge \neg S(z, y')) \rightarrow G(x, y)),$$

then the resulting existential second-order formula would express HAMILTONIAN CYCLE.

Finally, for an **NP**-complete problem such as HAMILTONIAN PATH, it is least likely that it can be expressed in *Horn* existential second-order logic with successor, unless $\mathbf{P} = \mathbf{NP}$.



Chapter 5

Concluding Remarks

Logic is fundamental to mathematics in the sense that it governs the formulation of mathematical statements and its deductions. In this thesis we restrict ourselves to the classical part of logic, which underlies mathemtics. There are some interesting properties about nonclassical logic. For example, intuitionistic logic, which was developed by A. Heyting, L. Kronecker and L. E. J. Brouwer and which abandons the use of the law of excluded-middle, is even closer to our algorithmic approach, since for a proof it gives an algorithm to produce the required conclusion. Others find its applications to many fields.

With the fact that HALTING PROBLEM is undecidable, it can be shown that first-order logic is also undecidable, i.e. whether or not a given formula follows from axioms is undecidable. This in turn can be used to show that number theory cannot be axiomitized by a recursively enumerable set of axioms. [6] Therefore it is impossible to devise a machine that, given some premises, automatically proves or refutes each statement. On the other hand, Hilbert's program, of which the ambitious goal is to completely axiomatize number theory and even the whole mathematics, was defeated by Gödel's imcompleteness theorem.

Moreover, the famous *continuum hypothesis*, proposed by G. Cantor, which assumes that the cardinality of $\mathbb{R}$ is $|2^{\mathbb{N}}|$, was proved independent of our axioms for *set theory* by K. Gödel and P. Cohen. The "second" Gödel's incompleteness theorem (the one provided in the text is the "first" Gödel's incompleteness theorem) states that the consistency of set theory itself, provided that set theory is consistent, is not provable within (axiomatic) set theory itself. With this result, that the assertion "Mathematics is consistent, i.e. there is no contradiction that can

be derived within it" is also not provable, since the whole mathematics can be built upon the notion of set. This brings us to review the rudiments of the formal method, which has evolved since the time of Aristotle and Euclid.

On the other hand, besides logic there are many other characterizations of computational problems. The efforts to the problem $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ keep on, and the investigations of logic as well. (For more discussions on the logical characterization about this problem, see [19].)

Bibliography

[1] R. Fagin "Generalized first-order spectra and polynomial-time recognizable sets," pp. 43–73 in *Complexity of Computation*, edited by R. M. Karp SIAM-AMS Proceedings, vol. 7, 1974.

[2] P. Kolatis and M. Vardi "0 – 1 laws and decision problems for fragments of second-order logic," *Proc. 3rd IEEE Symp. on Logic In Comp. Sci*, pp. 2–11, 1988.

[3] N. Immerman "Relational queries computable in polynomial time," *Information and Control, 68*, pp. 86–104, 1986.

[4] M. Y. Vardi "The complexity of relational query languages," *Proceedings of the 14th ACM Symp. on the Theory of Computing*, pp. 137–146, 1982.

[5] C. H. Papadimitriou "A note on the expressive power of PROLOG," *Bull. of the EATCS, 26*, pp. 21–23, 1985.

[6] C. H. Papadimitriou *Computational complexity*, second edition, Addison-Wesley Longman.

[7] E. Grädel "The expressive power of second-order Horn logic," *Proc. 8th Symp. on Theor. Aspects of Comp. Sci.*, vol. 480 of Lecture Notes in Computer Science, pp. 466–477, 1991.

[8] H.-D. Ebbinghaus, J. Flum, and W. Thomas *Mathematical logic*, second edition, Springer.

[9] H. B. Enderton *A Mathematical Introduction to Logic*, Academic Press.

[10] D. van Dalen *Logic and Structure*, fourth edition, Springer.

[11] R. P. Grimaldi *Discrete and Combinatorial Mathematics, an Applied Introduction*, fifth edition, Pearson Addison-Wesley.

[12] E. Mendelson *Introduction to Mathematical Logic*, fourth edition, Chapman & Hall/CRC.

[13] L. Henkin "The completeness of first-order functional calculus," *J. Symb. Logic, 14*, pp. 159 – 166, 1949.

[14] K. Gödel "Die Vollständigkeit der Axiome der Logischen Funktionenkalküls" (The completeness of the axioms of the logical function calculus), *Monat, Math. Physik, 37*, pp. 349 – 360, 1930.

[15] K. Gödel "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme" ("On formally undecidable theorems in Principia Mathematica and related systems), *Monatschefte für Mathematik und Physik, 38*, pp. 173 – 198, 1931

[16] L. Henkin, J. D. Monk and A. Tarski *Cylindric Algebras*, North Holland, 1971 – 1985.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein *Introduction to Algorithms*, second edition, MIT Press, Cambridge, Massachusetts, 2001

[18] E. Horowitz, S. Sahni, D. Mehta *Fundamentals of Data Structures in C++*, Computer Science Press, An imprint of W. H. Freeman and Company, New York, 1995.

[19] S. Hedman *A first course in logic : an introduction to model theory, proof theory, computability, and complexity*, Oxford, New York, Oxford University Press, 2004.

[20] S. Russell, P. Norvig *Artificial Intelligence, a Modern Approach*, second edition, Prentice Hall Series in Artificial Intelligence, 2003.