Study Guide for Phil279 L01 F17

Teppei Hayashi
Department of Philosophy
University of Calgary

Email: thayashi@ucalgary.ca

December 15, 2017

If you find any mistakes in this document, please let me know IMMEDIATELY.

Contents

Chapter 1

Language L_1

We are dealing with a language called L_1. So let's start with very basic aspects of this language.

L_1 can be seen from two different angles: *syntax* and *semantics*. In short, the syntax and semantics of L_1 tell us:

> **Syntax:** What kind of expression is considered as a sentence of L_1 (vocabulary and formation rules of a sentence of L_1).
>
> **Semantics:** How to assign the truth values {true, false}[1] to a sentence of L_1 (truth table).

Let's take a bit closer look at these two aspects.

1.1 Syntax of L_1

As we've seen in the above, the syntactical aspect of L_1 tells us what kind of expression is considered as a sentence of L_1. As in ordinary languages, the syntax of L_1 comprises of the vocabulary and formation rules (namely, grammar) of L_1-sentences. Let's start with the vocabulary of L_1.

[1] True and false are sometimes abbreviated as: T and F, t and f, T and $\perp$ (upside-down T), or 1 and 0 respectively. I'm going to use the first abbreviation (namely, T and F) throughout this study guide

1.1.1 The Vocabulary of L_1

The vocabulary of L_1 comprises of

Logical Symbols: $(,\ ),\ -, \wedge, \vee, \rightarrow, \leftrightarrow$

Non-Logical Symbols: Sentence letters $A, B, C, \ldots$ with or without subscript (e.g., A_1, B_2, Z_{1028}).

Auxiliary Symbols: Sentence variables $\alpha, \beta, \gamma, \ldots$ (the lower Greek letters) with or without subscript.

Sentence letters are sometimes called *atomic sentences*; if a sentence is not a sentence letter, it's called a *compound sentence*.

The logical symbols $-, \wedge, \vee, \rightarrow, \leftrightarrow$ are called *logical connectives*, and called "denial" (or "negation"), "conjunction", "disjunction", "conditional", and "bi-conditional" respectively. A sentence variable is used for referring to an arbitrary sentence of L_1. So, the sentence variable α may refer to a sentence letter like A, or a rather complicated sentence like $-(A \wedge (B \leftrightarrow (C \vee D)))$.

1.1.2 The Formation Rules of L_1-Sentences

The formation rules of L_1-sentences are as follows.

1. Every sentence letter is a sentence of L_1.
2. If α and β are sentences of L_1, so are $-\alpha, (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), (\alpha \leftrightarrow \beta)$.[2]
3. Nothing else is a sentence of L_1.

Thus, $-(A \wedge (B \leftrightarrow C))$ is a sentence of L_1 but $((\rightarrow A$ is not.

[2]Officially, the formation rules for a conjunction and a disjunction are as follows:

1. If $\alpha_1, \alpha_2, \ldots, \alpha_n$ are sentences of L_1, so is $(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n)$.
2. If $\alpha_1, \alpha_2, \ldots, \alpha_n$ are sentences of L_1, so is $(\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n)$.

However, for several reasons (especially in writing truth tables or truth trees), it would be better to adopt the above non-official way.

The components of a conjunction (α and β in the above case) are called *conjuncts*; the components of a disjunction are called *disjuncts*; and the first component of a conditional (α in the above case) is called an*antecedent* while the second component (β in the above case) is called a *consequent*.[3]

Some authors call sentence letters and their denials *literals*; so A, $-B$, C_{256} etc. are all literals.

1.1.3 The Main Connective

In writing truth tables or truth trees,[4] first you need to be able to answer the following question: "What kind of a sentence is this?" In the case of a denial sentence, it's easy to answer because every denial sentence has a form "$-(\ldots)$". In the case of a simple sentences like $(A \wedge B)$ or $(A \rightarrow B)$, it's easy as well; the former is a conjunctive sentence, the latter conditional. However, in the case of a more complicated sentence, figuring out what sentence we're thinking of can be a substantial task. Let's take $(((B \vee C) \rightarrow A) \leftrightarrow ((B \rightarrow A) \wedge (-A \rightarrow -C)))$ as an example.

First, note that a sentence of L_1 is built in a bottom-up way; in other words, a bigger sentence should be built from smaller ones. For example, in order to build $((A \wedge B) \rightarrow -C)$, we need to build $(A \wedge B)$ and $-C$ first, and then, connect them by $\rightarrow$ (and of course enclose it with brackets). In the case of our example, we need to build $-A$ and $-C$ first, and then $(B \vee C)$, $(B \rightarrow A)$ and $(-A \rightarrow -C)$, and then $((B \vee C) \rightarrow A)$ and $((B \rightarrow A) \wedge (-A \rightarrow -C))$, and lastly connect them by $\leftrightarrow$. This building process can be shown graphically as follows.

$$\overbrace{\overbrace{(\underbrace{(B \vee C)}_{(2)} \rightarrow A)}^{(3)} \leftrightarrow \overbrace{(\underbrace{(B \rightarrow A)}_{(2)} \wedge \overbrace{(\underbrace{-A}_{(1)} \rightarrow \underbrace{-C}_{(1)})}^{(2)})}^{(3)}}^{(4)}$$

In the stage 4 of the above building process, we connect the two sentences built in the stage 3 by $\leftrightarrow$; in other words, the last connective we

[3] There's no special way to call the components of a bi-conditional; we simply call them "the left-hand (right-hand) side of a bi-conditional".

[4] We're gonna learn what truth tables and truth trees are later in the course.

add in building the sentence is $\leftrightarrow$. We call this last connective the *main connective*, and say, "the sentence is a bi-conditional".

1.1.4 Problems

Determine the main connectives of the following sentences.

1. $((--A \wedge B) \rightarrow (-A \leftrightarrow B))$
2. $(-D \wedge (-H \vee (D \wedge E)))$
3. $(-(D \leftrightarrow (-A \wedge B)) \vee (-D \vee -B))$
4. $((J \wedge ((E \vee F) \wedge (-E \wedge -F))) \rightarrow -J)$
5. $-(-A \leftrightarrow -(B \leftrightarrow -(A \leftrightarrow (B \wedge C))))$

1.2 Semantics of L_1

The semantics of L_1 tells us how to assign the truth values {T,F} to a sentence of L_1. Such assignments of the truth values to a sentence are called *interpretations* or *truth-value assignments*. Let's start with the simplest one.

1.2.1 Interpretations of a Sentence Letter of L_1

A sentence letter α is going to be true in an interpretation I if and only if[5] I assigns true to α; otherwise, α is going to be false. The truth table for the interpretations of α is

α
T
F

Each row in the above truth table represents an interpretation; thus, there are two interpretations for each sentence letter.

[5]In the course, we use "just in case" (abbreviated as "jic") instead of "if and only if" (abbreviated as "iff"). Since "if and only if" is more popular (especially in mathematics), I adopt "if and only if" (iff) in this study guide.

1.2.1.1 Interpretations of a Denial

A denial $-\alpha$ is going to be true in I if and only if I assigns false to α; otherwise, $-\alpha$ is going to be false. Thus, the truth table for a denial $-\alpha$ is

α	$-\alpha$
T	F
F	T

1.2.2 Interpretations of a Conjunction

A conjunction $(\alpha \wedge \beta)$ is going to be true in I if and only if I assign true to both conjuncts; in other words, if I assigns false to at least one of its conjuncts, $(\alpha \wedge \beta)$ is going to be false in I.

α	β	$(\alpha \wedge \beta)$
T	T	T
T	F	F
F	T	F
F	F	F

Again, each row represents an interpretation. If a sentence in consideration contains two components, there should be four rows (interpretations).

1.2.3 Interpretations of a Disjunction

A disjunction $(\alpha \vee \beta)$ is going to be true in I if and only if I assigns true to at least one of the sentence's disjuncts; in other words, if I assigns false to both of its disjuncts, $(\alpha \vee \beta)$ is going to be false in I.

α	β	$(\alpha \vee \beta)$
T	T	T
T	F	T
F	T	T
F	F	F

Note that our disjunction is going to be true when I assigns true to both disjuncts; namely, our disjunction is *not* exclusive.

1.2.4 Interpretations of a Conditional

A conditional $(\alpha \to \beta)$ is going to be true in I if and only if I assigns false to the antecedent (α in this case) *or* true to the consequent (β in this case); in other words, if I assigns true to the antecedent *and* false to the consequent, $(\alpha \to \beta)$ is going to be false in I.

α	β	$(\alpha \to \beta)$
T	T	T
T	F	F
F	T	T
F	F	T

1.2.5 Interpretations of a Bi-Conditional

A bi-conditional $(\alpha \leftrightarrow \beta)$ is going to be true in I if and only if I assigns the same truth value to both components; in other words, if I assigns the different truth value to the components, $(\alpha \leftrightarrow \beta)$ is going to be false in I.

α	β	$(\alpha \leftrightarrow \beta)$
T	T	T
T	F	F
F	T	F
F	F	T

1.2.6 Examples

Let's construct the truth table for $(((B \vee C) \to A) \leftrightarrow ((B \to A) \wedge (-A \to -C)))$. In constructing it, you need to proceed from smaller sentences to bigger ones. So, you need to fill out the truth values for $-A$ and $-C$ first. (If you're not so sure why you need to do so, read Section 1.1.3 again.)

A	B	C	$(((B$	$\vee$	$C)$	$\rightarrow$	$A)$	$\leftrightarrow$	$((B$	$\rightarrow$	$A)$	$\wedge$	$($	$-$	A	$\rightarrow$	$-$	$C)))$
T	T	T												F			F	
T	T	F												F			T	
T	F	T												F			F	
T	F	F												F			T	
F	T	T												T			F	
F	T	F												T			T	
F	F	T												T			F	
F	F	F												T			T	

And then, $(B \vee C)$, $(B \rightarrow A)$, and $(-A \rightarrow -C)$.

A	B	C	$(((B$	$\vee$	$C)$	$\rightarrow$	$A)$	$\leftrightarrow$	$((B$	$\rightarrow$	$A)$	$\wedge$	$($	$-$	A	$\rightarrow$	$-$	$C)))$
T	T	T		T						T				F		T	F	
T	T	F		T						T				F		T	T	
T	F	T		T						T				F		T	F	
T	F	F		F						T				F		T	T	
F	T	T		T						F				T		F	F	
F	T	F		T						F				T		T	T	
F	F	T		T						T				T		F	F	
F	F	F		F						T				T		T	T	

And then, $((B \vee C) \rightarrow A)$ and $((B \rightarrow A) \wedge (-A \rightarrow -C))$.

A	B	C	$(((B$	$\vee$	$C)$	$\rightarrow$	$A)$	$\leftrightarrow$	$((B$	$\rightarrow$	$A)$	$\wedge$	$($	$-$	A	$\rightarrow$	$-$	$C)))$
T	T	T		T		T				T		T		F		T	F	
T	T	F		T		T				T		T		F		T	T	
T	F	T		T		T				T		T		F		T	F	
T	F	F		F		T				T		T		F		T	T	
F	T	T		T		F				F		F		T		F	F	
F	T	F		T		F				F		F		T		T	T	
F	F	T		T		F				T		F		T		F	F	
F	F	F		F		T				T		T		T		T	T	

And finally, based on what you've got in the above, you're gonna fill out the truth values for the entire sentence.

A	B	C	$(((B$	$\vee$	$C)$	$\rightarrow$	$A)$	$\leftrightarrow$	$((B$	$\rightarrow$	$A)$	$\wedge$	$($	$-$	A	$\rightarrow$	$-$	$C)))$
T	T	T		T		T		**T**		T		T		F		T	F	
T	T	F		T		T		**T**		T		T		F		T	T	
T	F	T		T		T		**T**		T		T		F		T	F	
T	F	F		F		T		**T**		T		T		F		T	T	
F	T	T		T		F		**T**		F		F		T		F	F	
F	T	F		T		F		**T**		F		F		T		T	T	
F	F	T		T		F		**T**		T		F		T		F	F	
F	F	F		F		T		**T**		T		T		T		T	T	

As you see, it turned out that $(((B \vee C) \rightarrow A) \leftrightarrow ((B \rightarrow A) \wedge (-A \rightarrow -C)))$ is a valid sentence. (For what is a valid sentence, see Section 1.3.3.)

1.2.7 Problems

Construct truth tables for the sentences in Problems 1.1.4.

1.3 Semantical Properties of (a Set of) L_1-Sentences

There are some properties of L_1-sentences which concern a semantical aspect of sentences. In short, those properties tell us how to classify a sentence according to what kind of truth-value assignments the sentence has. Let's take a look at them one by one.

1.3.1 Satisfiability

> A sentence (or a set of sentences) is called *satisfiable* if there is at least one interpretation where the sentence (or the set of sentences) is going to be true.

Here, some clarification as to when a set of sentences is considered as true (or false) would be in order.

A set of sentences is going to be true if and only if an interpretation I assigns true to all sentences in the set; otherwise, it's going to be false. In other words, the truth value for a set of sentences can be identified with that of the conjunction of the sentences in the set.

For example, the set of L_1-sentences $\{(A \to B), A\}$[6] is satisfiable because, in the following truth table,

A	B	$(A \to B)$	
T	T	T	$\Leftarrow$
T	F	F	
F	T	T	
F	F	T	

you find an interpretation (which is the line 1) where $(A \to B)$ and A are both true.

Note that any sentence letter is satisfiable. (See Section 1.2.1 for the truth table.)

Sometimes a satisfiable sentence (resp. a satisfiable set of sentences) is called a *consistent* sentence (resp. a consistent set of sentences).

1.3.2 Unsatisfiability

A sentence (or a set of sentences) is called *unsatisfiable* if there is no interpretation where the sentence (or the set of sentences) is going to be true. In other words, an unsatisfiable sentence (or an unsatisfiable set of sentences) is always false no matter what truth value we assign to sentence letters.

A paradigmatic example of unsatisfiable sentences would be $(A \wedge -A)$.

(	A	$\wedge$	$-$	$A)$
	T	**F**	F	
	F	**F**	T	

[6] We use the curly brackets {} for constructing a set.

What we find in the truth value assignments for $(A \wedge -A)$ is nothing but F.

Sometimes an unsatisfiable sentence (resp. an unsatisfiable set of sentences) is called an *inconsistent* sentence (resp. an inconsistent set of sentences).

1.3.3 Validity

Validity is the exact opposite of unsatisfiability.

> A sentence (or set of sentences) is called *valid* if and only if there's no interpretation where the sentence is going to be false; in other words, a valid sentence (or a valid set of sentences) is always true in all interpretation.

A paradigmatic example of valid sentences would be $(A \vee -A)$.

(	A	$\vee$	$-$	A)
	T	**T**	F	
	F	**T**	T	

What we find in the truth value assignments for $(A \vee -A)$ is nothing but T.

Sometimes a valid sentence is called a *tautology*.

1.3.4 Invalidity

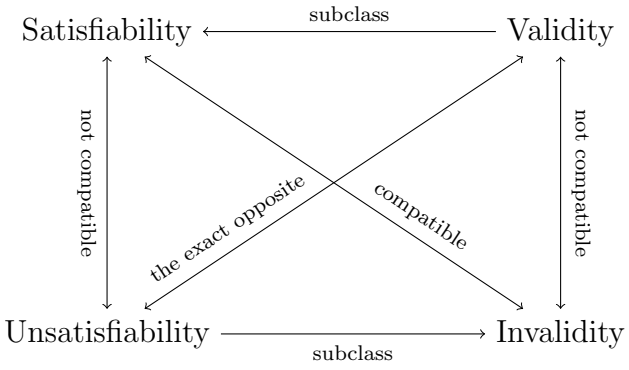
Invalidity is a kind of relatives of satisfiability.

> A sentence (or a set of sentences) is called *invalid* if there is at least one interpretation where the sentence (or the set of sentences) is going to be false.

For example, the set of L_1-sentences $\{(A \rightarrow B), A\}$ is invalid because, in its truth table (see Section 1.3.1), we find Fs in the lines 2–4.

1.3.5 Interrelations among the Properties

As you may notice from the wordings of their definitions, there are some interrelations among satisfiability/unsatisfiability/validity/invalidity as follows.



Note that the directions of the arrows matter. For example, there is a single-headed arrow from "Validity" to "Satisfiability"; this means that, together with the description "subclass" at the above of the arrow, validity is a subclass of satisfiability (meaning, every valid sentence is satisfiable but not every satisfiable sentence is valid). On the other hand, there is a double-headed arrow between "Satisfiability" and "Invalidity"; this means that, together with the description "compatible", a sentence can be both satisfiable and invalid at the same time.

1.3.6 Problems

Classify the following sentences into satisfiable/unsatisfiable/valid/invalid. (Note that a sentence can be classified into more than one category.)

1. $(J \to (K \to J))$
2. $((E \leftrightarrow H) \to (-E \to -H))$
3. $((((C \to D) \wedge (D \to E)) \wedge C) \wedge -E)$
4. $-(((A \vee B) \wedge (B \vee B)) \wedge (-A \wedge -B))$
5. $(-B \to ((B \vee D) \to D))$

1.4 Logical Implication and Logical Equivalence

1.4.1 Logical Implication

α *logically implies* β if and only if there's no interpretation where α is true but β is false. In other words, α implies β if and only if a conditional $\alpha \to \beta$ is valid.

We usually omit "logically" and simply say "α implies β". In the below, we'll use the notation "$\models$" for denoting logical implication.

The left-hand side of an implication can be a set of sentences; in such cases, we write $\{\alpha, \beta, \ldots, \gamma\} \models \delta$ or $\Gamma \models \delta$ (we use the upper Greek letters to denote a set of sentences). Obviously, here, you need to be able to assign truth values to a set of sentences.[7]

In order to figure out whether α implies β or not, you can use the truth table method. For example, you can figure out whether or not $-(A \to B)$ implies A just by taking a look at the following truth table.

A	B	$(A \to B)$	$-(A \to B)$
T	T	T	F
T	F	F	T
F	T	T	F
F	F	T	F

In the above table, there's no interpretation where $-(A \to B)$ is true but A is false; thus, you conclude that $-(A \to B)$ implies A.

Although the truth-table method comes in handy, writing a truth table can be quite tedious. If there's a handier way to figure out whether $\alpha \models \beta$ or not, we want to use it. Is there any such method? Actually there is.

Let's think about $((B \to C) \to (A \to B)) \models (A \to B)$ and try to show that $((B \to C) \to (A \to B))$ doesn't imply $(A \to B)$. For the implication to fail, $(A \to B)$ must be false; and for $(A \to B)$ to be false, A must be true and B must be false. Thus, we get the following truth-value assignment.

$$((\underset{\text{F}}{B} \to C) \to \underbrace{(\underset{\text{T}}{A} \to \underset{\text{F}}{B})}_{\text{F}}) \models \underbrace{(\underset{\text{T}}{A} \to \underset{\text{F}}{B})}_{\text{F}}$$

On the other hand, we want the left-hand side of the implication to be true; thus, since $(A \to B)$ has been already assigned the truth value F, $(B \to C)$

[7] If you're not so sure how to assign truth values to a set of sentences, see Section 1.3.1.

must be false in order for $((B \to C) \to (A \to B))$ to be true. However, we've already assigned F to B; consequently, it's impossible to make $(B \to C)$ false no matter what truth value we assign to C because a conditional with a false antecedent never be false. Therefore, it's impossible to assign T to the left-hand side of the implication and F to the right side; in other words, it's impossible to make the implication fail; $((B \to C) \to (A \to B))$ actually implies $(A \to B)$.

How about $(A \vee -(B \wedge C)) \models ((A \leftrightarrow C) \vee B)$? Let's see if we can make this implication fail. For the implication to fail, the right-hand side $((A \leftrightarrow C) \vee B)$ must be false; and for $((A \leftrightarrow C) \vee B)$ to be false, both of its conjuncts, namely $(A \leftrightarrow C)$ and B, must be false.

$$(A \vee -(\underset{\text{F}}{B} \wedge C)) \models \underbrace{((\underset{?}{A} \leftrightarrow \underset{?}{C}) \vee \underset{\text{F}}{B})}_{\text{F}}$$

In the above, for $(A \leftrightarrow C)$ to be false, there are two options; A is true and C is false, or A is false and C is true. If we choose the first option, we instantly know that the left-hand side of the disjunction is going to be true. Therefore, in this case, the implication fails and the counterexample is given by the truth values TFF for A, B, and C.

$$\underbrace{(\underset{\text{T}}{A} \vee -(\underset{\text{F}}{B} \wedge \underset{\text{F}}{C}))}_{\text{T}} \models \underbrace{((\underset{\text{T}}{A} \leftrightarrow \underset{\text{F}}{C}) \vee \underset{\text{F}}{B})}_{\text{F}}$$

On the other hand, if we choose the second option (A is false, C is true), the truth value of the disjunction on the left-hand side is determined by $-(B \wedge C)$.

$$\underbrace{(\underset{\text{F}}{A} \vee \underbrace{-\underbrace{(\underset{\text{F}}{B} \wedge \underset{\text{T}}{C})}_{\text{F}}}_{\text{T}})}_{\text{T}} \models \underbrace{((\underset{\text{F}}{A} \leftrightarrow \underset{\text{T}}{C}) \vee \underset{\text{F}}{B})}_{\text{F}}$$

In either way, the implication fails.

1.4.1.1 Problems

1. Prove:

1. $\alpha \models \alpha$
2. If $\Gamma \models \alpha$, then $\Gamma, \beta \models \alpha$ (Here, "Γ, β" on the left-hand side of the second implication means $\{\gamma_1, \gamma_2, \ldots, \gamma_n, \beta\}$ with $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$.
3. If $\Gamma \models \alpha$ and $\alpha, \Delta \models \beta$, then $\Gamma, \Delta \models \beta$
4. $\Gamma, \alpha \models \beta$ if and only if $\Gamma \models (\alpha \rightarrow \beta)$
5. If $\alpha \models \beta$ and $\beta \models \gamma$, then $\alpha \models \gamma$

2. Determine whether the following implications hold. If the implication fails, provide a counterexample.

1. $\{A, (B \wedge C)\} \models B$
2. $\{A, (B \vee C)\} \models B$
3. $\{(K \vee J), -(K \vee J)\} \models K$
4. $\{(W \vee J), ((W \rightarrow Z) \vee (J \rightarrow Z)), -Z\} \models -(W \wedge J)$

1.4.2 Logical Equivalence

> α *logically equivalent* to β if and only if $\alpha \models \beta$ and $\beta \models \alpha$; in other words, α is equivalent to β if and only if $\alpha \leftrightarrow \beta$ is valid.

We usually omit "logically" and simply say "α is equivalent to β". In the below, I use the notation "$\equiv$" for denoting logical equivalence. (Note that this is *my* notation; if you want to use this notation in the exam, please explicitly state that $\equiv$ denotes logical equivalence.)

As an example, let's prove that, if $S_1 \models S_2$, S_1 is equivalent to $(S_1 \wedge S_2)$ and S_2 is equivalent to $(S_1 \vee S_2)$. In order to prove this, we have to consider the following four implications.

1. $S_1 \models (S_1 \wedge S_2)$
2. $(S_1 \wedge S_2) \models S_1$
3. $S_2 \models (S_1 \vee S_2)$
4. $(S_1 \vee S_2) \models S_2$

In order for the first implication to fail, $(S_1 \wedge S_2)$ must be false; and in order for the conjunction to be false, at least one of its conjuncts (namely, either S_1 or S_2) must be false. However, if S_1 is false, the implication is going to hold because, in order for the implication to fail, S_1 on the left-hand side must be true. On the other hand, if S_2 is false, S_1 cannot be true because

we're assuming the truth of $S_1 \models S_2$. In either case, there's no interpretation where the left-hand side is true but the right-hand side is false.

In a similar fashion, you can make sure that there's no interpretation where the left-hand sides of the items 2–4 are true but the right-hand sides of the items are false. Therefore, S_1 is equivalent to $(S_1 \wedge S_2)$ and S_2 is equivalent to $(S_1 \vee S_2)$.

In this particular case, the truth-table method might be easier.

S_1	S_2	$(S_1 \wedge S_2)$	$(S_1 \vee S_2)$
T	T	T	T
~~T~~	~~F~~	~~F~~	~~T~~
F	T	F	T
F	F	F	F

Note that the situation depicted in the line 2 never happens because $S_1 \models S_2$ (there's no interpretation where S_1 is true but S_2 is false). Thus, we can safely ignore the line 2. Obviously, the turth-value assignments of S_1 (resp. S_2) match up with those of $(S_1 \wedge S_2)$ (resp. $(S_1 \vee S_2)$). Therefore, S_1 (resp. S_2) is equivalent to $(S_1 \wedge S_2)$ (resp. $(S_1 \vee S_2)$).

Let's do one more example by constructing a truth table. This time we're gonna check whether $(A \rightarrow B)$ and $(-A \vee B)$ are logically equivalent or not.

A	B	$(A \rightarrow B)$	$(-A \vee B)$	$((A \rightarrow B) \leftrightarrow (-A \vee B))$
T	T	T	T	T
T	F	F	F	T
F	T	T	T	T
F	F	T	T	T

The truth-value assignments for $((A \rightarrow B) \leftrightarrow (-A \vee B))$ are all true; namely, $((A \rightarrow B) \leftrightarrow (-A \vee B))$ is valid. This shows that $(A \rightarrow B)$ and $(-A \vee B)$ are logically equivalent.

1.4.2.1 Problems

1. Prove:

1. $--\alpha \equiv \alpha$ (Remember: "$\equiv$" is read as "is equivalent to".)
2. $\alpha \equiv -\beta$ if and only if $-\alpha \equiv \beta$

3. $-(\alpha \wedge \beta) \equiv (-\alpha \vee -\beta)$
4. $-(\alpha \vee \beta) \equiv (-\alpha \wedge -\beta)$
5. $(\alpha \to \beta) \equiv (-\alpha \vee \beta)$
6. $(\alpha \to \beta) \equiv (-\beta \to -\alpha)$
7. $(\alpha \leftrightarrow -\beta) \equiv (-\alpha \leftrightarrow \beta)$

(The above equivalences are all useful. I recommend you to memorize them.)

2. Determine whether or not the following equivalences are correct.

1. $(A \to (B \to A)) \equiv ((C \wedge -C) \vee (A \to A))$
2. $(C \wedge (B \vee A)) \equiv ((C \wedge B) \vee A)$
3. $(-(D \vee B) \to (C \to B)) \equiv (C \to (D \wedge B))$
4. $(A \to (B \to (A \to B))) \equiv (B \to (A \to (B \to A)))$
5. $(F \vee -(G \vee -H)) \equiv ((H \leftrightarrow -F) \vee G)$

1.5 Argument

An *argument* is comprised of two parts: a set of premises and one conclusion. An argument is called *valid* if there's no interpretation where a set of premises is true[8] but the conclusion is false. If there's such an interpretation, an argument is called *invalid* and the interpretation is called a *counterexample*.

We express an argument in the following form.

Premise 1
Premise 2
⋮
Premise n

Conclusion

As is easily seen, the definitions of (valid) arguments and implications are almost identical; so, we sometimes write an argument in the form of $\{\text{Premise 1}, \text{Premise 2}, \ldots, \text{Premise } n\} \models \text{Conclusion}$.

Let's think of the following simple argument.

[8] If you're not sure when a set of sentences (which are premises of an argument) is going to be true, see Section 1.3.1.

$$\frac{\begin{array}{c}(A \to B)\\ A\end{array}}{B}$$

You can show its validity (or invalidity) either by the truth-table method or by the one explained above (let's call this method the *try-to-refute* method).

Truth-Table Method:

A	B	$(A \to B)$
T	T	T
T	F	F
F	T	T
F	F	T

As you see, there's no interpretation where B (your conclusion) is false but both A and $(A \to B)$ (your premises) are true; therefore, this argument is valid.

Try-to-Refute Method:

$$\underbrace{\{\underbrace{(\underset{\text{F}}{A} \to \underset{\text{F}}{B})}_{\text{T}}, \underset{\text{F}}{A}\}}_{\text{F}} \models \underset{\text{F}}{B}$$

In order for the above argument to be invalid, B must be false; and in order for $(A \to B)$ to be true with the truth value F of B, A must be false. However, such a truth-value assignment makes the set of premises false (why?); and consequently, the argument is going to be valid. In this way, we know that there's no way to assign T to the premises and F to the conclusion, which means the argument is valid.[9]

Let's do another example: $\{(A \to B), B\} \models A$.

Truth-Table Method:

A	B	$(A \to B)$	
T	T	T	
T	F	F	
F	T	T	$\Leftarrow$
F	F	T	

[9] An argument with the form $\{(\alpha \to \beta), \alpha\} \models \beta$ is called *modus ponens*.

Try-to-Refute Method:

$$\underbrace{\{\underbrace{(\underset{\text{F}}{A} \rightarrow \underset{\text{T}}{B})}_{\text{T}}, \underset{\text{T}}{B}\}}_{\text{T}} \models \underset{\text{F}}{A}$$

In both ways, we find the interpretation which assigns T to the premises and F to the conclusion. Further, we know that it happens when A is false and B is true; that truth-value assignment would be your counterexample to the argument.

Here's a more complicated example: $\{(A \leftrightarrow (-B \vee C)), (C \leftrightarrow -A)\} \models -A$.

Truth-Table Method:

A	B	C	$-A$	$-B$	$(-B \vee C)$	$(A \leftrightarrow (-B \vee C))$	$(C \leftrightarrow -A)$	
T	T	T	F	F	T	T	F	
T	T	F	F	F	F	F	T	
T	F	T	F	T	T	T	F	
T	F	F	**F**	T	T	**T**	**T**	$\Leftarrow$
F	T	T	T	F	T	F	T	
F	T	F	T	F	F	T	F	
F	F	T	T	T	T	F	F	
F	F	F	T	T	T	F	T	

In the line 4, we find a counterexample (A: T, B: F, C: F) which makes all the premises true but the conclusion false.

Try-to-Refute Method:

$$\underbrace{\{\underbrace{(\underset{\text{T}}{A} \leftrightarrow \underbrace{(\underbrace{-\underset{\text{F}}{B}}_{\text{T}} \vee \underset{\text{F}}{C})}_{\text{T}}}_{\text{T}}, \underbrace{(\underset{\text{F}}{C} \leftrightarrow \underbrace{-\underset{\text{F}}{A}}_{\text{F}})}_{\text{T}}\}}_{\text{T}} \models \underbrace{-\underset{\text{T}}{A}}_{\text{F}}$$

In order for the argument to be invalid, $-A$ must be false; thus, A must be true. On the other hand, all the premises must be true. In order for the premises to be all true, both $(A \leftrightarrow (B \vee C))$ and $(C \leftrightarrow -A)$ must be

true. In order for $(C \leftrightarrow -A)$ to be true, the truth value of C must match up with that of $-A$; thus, C must be false. In order for $(A \leftrightarrow (B \vee C))$ to be true, the truth values of the both sides of $\leftrightarrow$ must match up; and since the one side of the bi-conditional, namely A, is true, $(-B \vee C)$ must be true as well. Since C is false, in order for $(-B \vee C)$ to be true, $-B$ must be true; thus, B must be false. In this truth-value assignment (A: T, B: F, C: F), we successfully assigned T to the premises and F to the conclusion, which means the argument is invalid.

1.5.1 Problems

1. Prove:

1. $\{(\alpha \vee \beta), -\alpha\} \models \beta$
2. $\{(\alpha \rightarrow \beta), -\beta\} \models -\alpha$
3. $\{(\alpha \rightarrow \gamma), (\beta \rightarrow \gamma), (\alpha \vee \beta)\} \models \gamma$

(The above forms of arguments are called *disjunctive syllogism*, *modus tollens*, and *constructive dillema* respectively.)

2. Determine whether or not the following arguments are valid.

1. $\{(B \vee (A \wedge -C)), ((C \rightarrow A) \leftrightarrow B), (-B \vee A)\} \models -(A \vee C)$
2. $\{-(Y \leftrightarrow A), -Y, -A\} \models (W \wedge -W)$
3. $\{(B \vee B), ((-B \rightarrow (-D \vee -C)) \wedge ((-D \vee C) \vee (B \vee C)))\} \models C$
4. $\{(((J \wedge T) \wedge Y) \vee (-J \rightarrow -Y)), (J \rightarrow T), (T \rightarrow Y)\} \models (Y \leftrightarrow T)$
5. $\{((A \wedge (B \vee C)) \leftrightarrow (A \vee B)), (B \rightarrow -B)\} \models (C \vee A)$

1.6 Some Words on Implication/Argument

Some students may find the following aspects troubling.

1. The difference between the validities of a sentence and of an argument.
2. The condition in which an implication (resp. argument) holds.

In this section, I like to arouse such students' attention.

1.6.1 The Difference between the Validities of a Sentence and of an Argument

Let's replicate the definitions where the same word "valid" appears.

> A sentence (or set of sentences) is called *valid* if and only if there's no interpretation where the sentence is going to be false; in other words, a valid sentence is always true in all interpretation.

> An argument is called *valid* if there's no interpretation where a set of premises is true but the conclusion is false.

Some students seem to confuse the definition of the validity of a sentence with that of the validity of an argument and think that an argument is going to be valid if and only if both the premises and the conclusion are true in all interpretations. Surely, if its premises and conclusion are always true, an argument is going to be valid; however, this is not the only case where the argument is going to be valid. Let's see when an argument is going to be valid in the next section.

1.6.2 The Condition in Which an Implication and an Argument Hold

First, let's take a look again at the definitions of implication and argument.

> α *logically implies* β if and only if there's no interpretation where α is true but β is false.

> An argument is called *valid* if there's no interpretation where a set of premises is true but the conclusion is false.

Note that they don't explicitly tell us when an implication or an argument holds. However, they *implicitly* tell us when an implication or an argument is going to be valid; as long as there's no interpretation where α (resp. a set of premises) is true but β (resp. the conclusion) is false, an implication (resp. argument) holds. More explicitly, α implies β if the configuration of the truth values for α and β take one of the following forms.

$$\mathrm{T} \models \mathrm{T},\ \mathrm{F} \models \mathrm{T},\ \mathrm{F} \models \mathrm{F}$$

Likewise, an argument is going to be valid if the configuration of the truth values for a set of premises and the conclusion take one of the following forms.

$$\frac{\mathrm{T}}{\mathrm{T}} \quad \frac{\mathrm{F}}{\mathrm{T}} \quad \frac{\mathrm{F}}{\mathrm{F}}$$

When you think about the situations where α implies β or the situations where an argument is valid, please don't forget the cases $\mathrm{F} \models \mathrm{T}$, $\mathrm{F} \models \mathrm{F}$, $\frac{\mathrm{F}}{\mathrm{T}}$, and $\frac{\mathrm{F}}{\mathrm{F}}$.

1.7 Complete Disjunctive Normal Form

1.7.1 Truth-Functional Connectives

As we know perfectly well by now, there are four combinations of the truth values for a pair of sentence letters α, β: (T, T), (T, F), (F, T), and (F, F). Those combinations of the truth values give us different truth-value assignments according to which connective we're thinking of. For example, if we're thinking of $\wedge$, the truth-value combination (T, F) gives us the truth value F; and if we're thinking of $\vee$, it gives us T.

From the above consideration, we notice that we can think of a connective in terms of a *mathematical function*. A mathematical function receive some inputs and returns an output. For example, the conjunction $\wedge$ can be thought of as a function which receive two inputs and return an output: $f_{\wedge}(\alpha, \beta)$. This function gives us T when we input T to both α and β; in other cases, it gives us F.

Is there any mechanical (i.e. algorithmic) way to express an *arbitrary* function in our language L_1? In other words, if we're given some sentence letters and truth-value assignments to each combination of those letters, can we find an L_1-sentence which meet those specifications? In simple cases, we may be able to find such a sentence by trial and error. For example, for the following truth table, we can manage to figure out that it can be expressed as $(A \vee -(A \vee B))$ (or in a simpler form, $(B \rightarrow A)$).

A	B	γ
T	T	T
T	F	T
F	T	F
F	F	T

But what if we're given the following truth table and asked to construct an L_1-sentence which has the truth-value assignments given in the table?

A	B	C	δ
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	T
F	T	T	F
F	T	F	T
F	F	T	F
F	F	F	T

We're at a loss.

1.7.2 Complete Disjunctive Normal Form

Luckily, we have a mechanical method for finding an L_1-sentence which meets a given specification. In order to explain what that method is (and how it works), let's go back to the above example.

The sentence δ defined in the above truth table is going to be true if all the sentence letters in it are true. Thus, this situation can be expressed as $(A \wedge B \wedge C)$ because this sentence is going to be true only when we assign T to all A, B, C. From the line 4, we also notice that the sentence δ is going to be true when $(A \wedge -B \wedge -C)$ is true (why?). In this way, we know that δ is going to be true if at least one of the following L_1-sentences is true: $(A \wedge B \wedge C), (A \wedge -B \wedge -C), (-A \wedge B \wedge -C), (-A \wedge -B \wedge -C)$. Thus, we can express δ as $((A \wedge B \wedge C) \vee (A \wedge -B \wedge -C) \vee (-A \wedge B \wedge -C) \vee (-A \wedge -B \wedge -C))$. This is exactly what is called a *complete disjunctive normal form.*

A *complete disjunctive normal form* of a sentence is

1. a disjunction
2. whose disjuncts are conjunctions
3. whose conjuncts are comprised of all the sentence letters (possibly with the denial) of the target sentence
4. each of which appears once and only once in each conjunction.

Let's compare what we've got in the above with this definition. $((A\wedge B\wedge C)\vee(A\wedge -B\wedge -C)\vee(-A\wedge B\wedge -C)\vee(-A\wedge -B\wedge -C))$ is obviously a disjunction; also, its disjuncts are all conjunctions; and these conjunctions are comprised of the sentence letters and/or their denials of the target sentence δ; therefore, $((A\wedge B\wedge C)\vee(A\wedge -B\wedge -C)\vee(-A\wedge B\wedge -C)\vee(-A\wedge -B\wedge -C))$ has a complete disjunctive normal form.

In fact, given any sentence (except one special case. We're gonna look at such a special case below), we can transform it into a complete disjunctive normal form in a mechanical way. Let's take $((A \wedge B) \vee -A)$ as an example. First, you need to construct its truth table.

A	B	$-A$	$(A \wedge B)$	$((A \wedge B) \vee -A)$
T	T	F	T	T
T	F	F	F	F
F	T	T	F	T
F	F	T	F	T

And then, look at the interpretations where the sentence $((A \wedge B) \vee -A)$ is going to be true. In our case, the sentence is going to be true in the lines 1, 3, and 4. In the line 1, the sentence letters A, B take the truth value T; and this situation can be expressed by $(A \wedge B)$. Similarly, the situations depicted in the lines 3–4, namely where the sentence letters A, B take the truth value (F, T) and (F, F) respectively, can be expressed as $(-A \wedge B)$ and $(-A \wedge -B)$ respectively ($(-A \wedge B)$ is going to be true if and only if A is false and B is true, and $(-A \wedge -B)$ is going to be true if and only if both A and B are false). Since $((A \wedge B) \vee -A)$ is going to be true in one of those situations, the sentence can be expressed as $((A \wedge B) \vee (-A \wedge B) \vee (-A \wedge -B))$ which is a complete disjunctive normal form.

To recap:

If you're asked to transform a sentence α into its complete disjunctive normal form,

1. Construct a truth table for the target sentence α.
2. Find rows where the target sentence is true.
3. In each row where the target sentence is true, look at the truth-value assignments for the sentence letters, and create a conjunction based on those truth-value assignments as follows.
   - α. If the truth-value assignment for the sentence letter is true, the sentence letter itself is one of your conjuncts.
   - β. If the truth-value assignment for the sentence letter is false, the denial of the sentence letter is one of your conjuncts.
   - γ. Connect what you've got above with $\wedge$ (and of course enclose it with brackets).
4. Connect the conjunctions you've got above with $\vee$ (and don't forget brackets).

Now, let's consider the following truth table for $(A \wedge -A)$ and construct a complete disjunctive normal form out of it.

A	B	$(A \wedge -A)$
T	T	F
T	F	F
F	T	F
F	F	F

As is easily seen, there's no row where $(A \wedge -A)$ is going to be true; $(A \wedge -A)$ is an unsatisfiable sentence. However, according to the above instruction, we have to take a look at the rows where the target sentence (in this case, $(A \wedge -A)$) is going to be true; obviously, there's no row to look at. In fact, there's no *complete* disjunctive normal form corresponding to any unsatisfiable sentence. What should we do? Well, if you're asked to construct a complete disjunctive normal form of an unsatisfiable sentence, simply reply, "There's no complete disjunctive normal form of it".[10]

[10]However, there are disjunctive normal forms of an unsatisfiable sentence. First, note that we dropped the qualification "complete" here; your conjunctions (which will be part of your disjunction) can have the same sentence letter more than once, and can lack

1.7.2.1 Problems

Find complete disjunctive normal forms for the following sentences.

1. $(-(A \to B) \vee (-A \wedge C))$
2. $((A \vee B) \wedge (-B \vee C))$
3. $((A \wedge -B) \vee (A \wedge C))$
4. $(-A \vee (B \to -C))$
5. $((A \vee B) \leftrightarrow -C)$

1.8 Expressive Completeness

Now we know that any sentence can be expressed in a (complete) disjunctive normal form. From this, we can conclude that the connectives $-, \wedge, \vee$ would suffice to express any sentences. We call such a set of connectives *expressively complete*.

> A set of connectives is called *expressively complete* if you can express any sentence (i.e. any possible truth-value assignments) with the connectives in the set.

From Problems 1.4.2.1, we know that $\wedge$ can be defined in terms of $-$ and $\vee$. This means that $-$ and $\vee$ are sufficient for expressing any sentence; the set $\{-, \vee\}$ is expressively complete.

> Given an expressively complete set Γ of connectives; if you can express each connective in Γ in terms of connectives in another set Δ of connectives, Δ is also expressively complete.

Is $\{-, \vee\}$ the smallest expressively complete set? There are actually two expressively complete sets each of which is comprised of just one connective: $|$ (the *Sheffer stroke*, or *NAND*) and $\downarrow$ (the *Peirce arrow*, or *NOR*).

---

one or more sentence letters. For example, with a target sentence with three sentence letter A, B, C, all the following are legitimate candidates for conjunctions for a disjunctive normal form: $(A \wedge A), (B \wedge -C), (A, B, C)$. Now, let's go back to our main business; find a disjunctive normal form of an unsatisfiable sentence. Let's cut to the chase; the target sentence $(A \wedge -A)$ itself would do the job. "Wait. This is a conjunction, not a disjunction. Why is this considered as a *disjunctive* normal form?", you may ask. Answer: it's a disjunction of $(A \wedge -A)$ and the empty sentence.

α	β	$(\alpha \mid \beta)$
T	T	F
T	F	T
F	T	T
F	F	T

α	β	$(\alpha \downarrow \beta)$
T	T	F
T	F	F
F	T	F
F	F	T

Let's focus on the Peirce arrow. As you may notice, the truth-value assignments for $(\alpha \downarrow \beta)$ are exactly the opposite of those of $(\alpha \vee \beta)$ (that's why it's sometimes called "NOR"!).[11] With this, we know that $(\alpha \vee \beta)$ can be expressed as $-(\alpha \downarrow \beta)$. Thus, if we figure out how to express "$-$" (denial) with the Peirce arrow alone, we can express "$\vee$" with the arrow alone as well. So let's figure it out.

From the above truth table, we know that $(\alpha \downarrow \beta)$ is going to be true if α and β are both false and it's gonna be false if α and β are both true. Now let's replace β in the above table with α and see what's gonna happen.

α	$(\alpha \downarrow \alpha)$
T	F
F	T

$(\alpha \downarrow \alpha)$ is going to be false if α is true, true if α is false. This is exactly what we want for the denial. Thus, we can define $-\alpha$ as $(\alpha \downarrow \alpha)$.

Since we now know how to express "$-$" with the Peirce arrow alone, we also know how to express "$\vee$" with the arrow alone; $(\alpha \downarrow \beta)$ being defined as $-(\alpha \vee \beta)$, its denial should give you $(\alpha \vee \beta)$.

α	β	$((\alpha \downarrow \beta)\mid(\alpha \downarrow \beta))$	$(\equiv (\alpha \vee \beta))$
T	T	T	
T	T	F	
F	T	F	
F	F	F	

And we now know how to express "$\wedge$" with $\{-, \vee\}$ (recall that $(\alpha \wedge \beta) \equiv -(-\alpha \vee -\beta)$ and $(\alpha \vee \beta) \equiv -(\alpha \downarrow \beta)$), we also know how to express "$\wedge$" with the Peirce arrow alone.

[11] And the truth-value assignments for the Sheffer stroke $\mid$ are exactly the opposite of those of $\wedge$; that's why it's called "NAND".

In a similar fashion, you can show that the Sheffer arrow $|$ is expressively complete.

Now let f be a zero-place connective (that is, a connective which connects nothing). You can think of this as a shorthand for $(A \wedge -A)$ (or any other unsatisfiable sentences). Then, the set $\{f, \rightarrow\}$ is expressively complete (that is, you can express any sentences in which other connectives $-, \wedge, \vee, \leftrightarrow$ might appear). Let's show this.

In order to show that $\{f, \rightarrow\}$ is expressively complete, we have to show that some other expressively complete set can be expressed in terms of $\{f, \rightarrow\}$. Let's take $\{-, \wedge\}$ as an expressively complete set. Our first task is to express $-$ only with $\{f, \rightarrow\}$.

Now take a look at the following truth table for $\rightarrow$.

α	β	$(\alpha \rightarrow \beta)$
T	T	T
T	F	F
F	T	T
F	F	T

We notice that, when the consequent (β) is false, the true antecedent gives us false, the false antecedent true; namely, when the consequent is false, $(\alpha \rightarrow \beta)$ gives us the truth value for the denial of the antecedent. So, with f and $\rightarrow$, we can express $-$ as follows.

α	$(\alpha \rightarrow f)$	$(\equiv -\alpha)$
T	F	
F	T	

Next, we have to express $\wedge$ with $\{f, \rightarrow\}$. Note that $(\alpha \rightarrow \beta)$ can be expressed as (sometimes even defined by) $(-\alpha \vee \beta)$ and $(\alpha \wedge \beta)$ can be expressed as $-(-\alpha \vee -\beta)$ (de Morgan's law). Therefore, $\alpha \wedge \beta$ can be expressed as $((\alpha \rightarrow (\beta \rightarrow f)) \rightarrow f)$. Since we've just shown how to express $\{-, \wedge\}$ in terms of $\{f, \rightarrow\}$, we've also shown that $\{f, \rightarrow\}$ is expressively complete.

1.8.1 Problems

1. Express $\vee, \rightarrow, \downarrow$ in terms of $|$.
2. Express $-, \wedge, \vee, \rightarrow, |$ in terms of $\downarrow$.
3. Express $\vee, \rightarrow, |, \downarrow$ in terms of $\{-, \wedge\}$.

4. Express $\wedge, \rightarrow, |, \downarrow$ in terms of $\{-, \vee\}$.
5. Express $\wedge, \vee, |, \downarrow$ in terms of $\{-, \rightarrow\}$.

1.9 Truth-Tree Method for L_1

To figure out the satisfiability/unsatisfiability/validity/invalidity of a sentence, the validity of an argument, or the truth of a logical implication (resp. a logical equivalence), the truth-table method comes in handy. However, as the number of sentence letters in a sentence increases, constructing a truth table becomes a substantial task. It would be nice if there's a handier way to figure out the above properties of a sentence or an argument. The *truth-tree method* provides such a handier way.

To construct a *truth tree* is, basically, to decompose sentences into literals. (Remember: literals refer to sentence letters and their denials. See 1.1.2.) In constructing it, your tree grows, maybe branching into paths, and eventually stops growing when every sentences in a set are decomposed into literals.

By decomposing sentences into literals, it enables us to easily find whether a set of sentences is satisfiable (consistent) or unsatisfiable (inconsistent). If you find at least one contradiction (a contradictory pair of sentences like A and $-A$; namely, a pair of a sentence letter and its own denial) in each path of your tree, your set of sentences is unsatisfiable. On the other hand, after finish constructing your tree, if there's at least one path immune from contradiction, your set of sentences is satisfiable. The truth-tree method tells us how to decompose a sentence into its components for that purpose.

1.9.1 10 Rules of the Truth-Tree Method

There are 10 rules you need to memorize. Each rule is comprised of two parts: the premise and the conclusions.

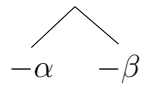
2 Rules for Denial

1. α
   $-\alpha$
   $\times$

2. ✓ $--\alpha$
   α

2 Rules for Conjunction

3. ✓ $(\alpha \wedge \beta)$
α
β

4. ✓ $-(\alpha \wedge \beta)$


2 Rules for Disjunction

5. ✓ $(\alpha \vee \beta)$

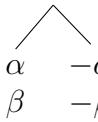

6. ✓ $-(\alpha \vee \beta)$
$-\alpha$
$-\beta$

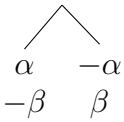
2 Rules for Conditional

7. ✓ $(\alpha \rightarrow \beta)$


8. ✓ $-(\alpha \rightarrow \beta)$
α
$-\beta$

2 Rules for Bi-Conditional

9. ✓ $(\alpha \leftrightarrow \beta)$


10. ✓ $-(\alpha \leftrightarrow \beta)$


Memorizing these 10 rules may not sound quite lovely; however, if you have a good understanding of the truth conditions of the logical connectives $(-, \wedge, \vee, \rightarrow, \leftrightarrow)$, memorizing them is not so hard.

Let's take the rules for $\wedge$ as an example. The result of decomposing $(\alpha \wedge \beta)$ is, according to Rule 3, vertically-aligned α and β (in this case, the premise of the rule is $(\alpha \wedge \beta)$, and the conclusions are α and β); and when two sentences are vertically aligned, it means that, if there's an interpretation which assigns T to those two sentence, that interpretation also assigns T to a sentence from which the two sentences come. Thus, in the case of vertically-aligned α and β, it means that, if there's an interpretation which assigns T to α and β, that interpretation also assigns T to $(\alpha \wedge \beta)$; this is nothing but the truth condition of $(\alpha \wedge \beta)$. On the other hand, Rule 5 tells us that we have two branches after decomposing $(\alpha \vee \beta)$; and this means that, if there's an interpretation which assigns T to either α or β (maybe both), that interpretation also assigns T to $(\alpha \vee \beta)$; again, this is nothing but the truth condition of $(\alpha \vee \beta)$.

In the above, we described the relation between a rule and the truth condition of its premise with starting from the truth of conclusions (the existence of an interpretation which makes both, or at least one, of conclusions). We can of course describe the relation the other way around; namely, from the assumption of the truth of the premise, we can conclude the truth of conclusions. When this relation is seen from the first point of view (from the truth of the conclusions to that of the premise), it's called the *completeness* of the rules; on the other hand, when the relation is seen from the second point of view (from the truth of the premise to that of the conclusions), it's called the *soundness* of the rules.

1.9.1.1 Problem

Convince yourself that the rules are closely connected to the truth conditions of the logical connectives; namely, verify that each rule is sound and complete.

In doing the above problem, you may have noticed that two of the rules don't seem to be connected to the truth conditions of the connectives in a straightforward way: those for the denial. (Still, there are some connections between the rules and the truth condition of the denial of course).

The plain version of the rule for the denial (Rule 1) tells you that you can close a path if you find a contradictory pair of sentences. (The meaning of "closing a path" will become clear eventually.) Its negated version (Rule 2) tells you that you can take the double negation $(--)$ away.

Now it's time to see how these rules work.

1.9.2 Testing the Satisfiability of sentences

To test the satisfiability of a set of sentences, start constructing a tree with the sentences themselves. If all paths of the tree are closed, the set is unsatisfiable; if there's at least one open path in the finished tree (namely, the tree which stops growing), the set is satisfiable; and such an open path tells you an interpretation which makes the set true.

Let's start with a simple example: $\{(A \wedge -B), C, (-A \vee -B)\}$.

You start constructing your tree by listing your sentences up vertically. We call this list of sentences an *initial list*.

1. $(A \wedge -B)$
2. C
3. $(-A \vee -B)$

And then, apply one of the rules to one of the sentences. Here, we have two options: apply the rule 3 to the line 1, or apply the rule 5 to the line 3. In general, if you can apply a non-branching rule (like the rules 3, 6, 8), apply it.

Rule of Thumb 1: Apply a non-branching rule when you can.

Let's apply the non-branching rule 3 to the line 1. (In the below I wrote down which rule is applied to which line. This is purely for pedagogical purpose. You don't have to do it in the exam or in the assignments.)

1.	✓ $(A \wedge -B)$	
2.	C	
3.	$(-A \vee -C)$	
4.	A	Rule 3 to Line 1
5.	$-B$	Rule 3 to Line 1

After applying a rule to a sentence, you should put a check mark to the left or right of the sentence in order to avoid applying a rule again to it.

Then, there's only one option left: apply the rule 5 to the line 3.

1.	✓ $(A \wedge -B)$	
2.	C	
3.	✓ $(-A \vee -C)$	
4.	A	Rule 3 to Line 1
5.	$-B$	Rule 3 to Line 1
6.	$-A \quad -C$	Rule 5 to Line 3

At this point, we have two paths; one with sentences $(A \wedge -B), C, (-A \vee -C), A, -B, -A$, the other with $(A \wedge -B), C, (-A \vee -C), A, -B, -C$.

All sentences except literals are check-marked; there's no sentence left to which our rules can be apply. We finish drawing the tree. The next task is to find a contradiction (a sentence and its own denial) in the paths. In this example, it's obvious; we can easily find two contradictions: A in the line 4 and $-A$ in the line 6, and C in the line 2 and $-C$ in the line 6. And the following is what to do if you come across such a path.

> If you find a contradiction, you put $\times$ under the path where you find it. When you do this, we say that you close the path. And if all the paths are closed, we say that the tree is closed.

1.	✓ $(A \wedge -B)$	
2.	C	
3.	✓ $(-A \vee -C)$	
4.	A	Rule 3 to Line 1
5.	$-B$	Rule 3 to Line 1
6.	$-A$ $\quad$ $-C$ $\times$ $\quad$ $\times$ 4,6 $\quad$ 2,6	Rule 5 to Line 3

(In the above I wrote down which line and which line are contradictory. Again, this is just for pedagogical purpose. You don't have to do it in the exam or in the assignments.)

All the paths in the above tree are closed. What does this mean? Well, let's look at the left path and make a list from the sentences in the path: $\{(A \wedge -B), C, (-A \vee -C), A, -B, -A\}$. Recall that the results of the decomposition are the truth conditions for sentences in our initial list. If there are some contradictory sentences (A and $-A$ in this case), some of the truth conditions for sentences are conflicting; in other words, there's no interpretation which assigns T to all the sentences in the list at the same time. And if there are conflicting truth conditions in all the paths, there's no way to assign T to the sentences in the initial list at the same time; in short, a set of the sentences in the initial list is unsatisfiable.

Let's do another example: the satisfiability check for $\{(A \to B), (B \to A), -A\}$. As before, we start drawing our tree by listing up the sentences vertically.

1. $(A \to B)$
2. $(B \to A)$
3. $-A$

This time, unfortunately, there's no non-branching rule we can apply. Thus, it seems it doesn't matter which rule we apply first. However, if we apply the rule 7 to the line 2, we immediately come across a contradiction; and we can close the path.

Rule of Thumb 2: Apply the rule which leads to a contradiction.

1. $(A \to B)$
2. ✓ $(B \to A)$
3. $-A$
4. $-B$ | A ×3,4 — Rule 7 to Line 2

There's one more sentence to decompose in the tree.

1. ✓ $(A \to B)$
2. ✓ $(B \to A)$
3. $-A$
4. $-B$ | A ×3,4 — Rule 7 to Line 2
5. $-A$ | B ×4,5 — Rule 7 to Line 1

We decomposed all the sentences except literals; we finish constructing the tree. We call this tree *finished*, and because the left-most path remains open, we also call the tree *open*. And if there's at least one open path in your finished tree, the set of sentences in the initial list is satisfiable.[12]

[12] In order to know whether or not the set of sentences in the initial list of a tree is satisfiable, you don't necessarily have to finish your entire tree. We'll be back to this point later.

1.9.2.1 Problem

Explain why the existence of at least one open path means the satisfiability of the set of sentences in the initial list.

Now we know that the set of sentences $(A \to B), (B \to A), -A$ is satisfiable; and we want to know under what truth-value assignments it's going to be true. To figure this out, we need to go back the path to its root, and assign T to each sentence letter which appear as a line itself. From $-A$ to $(A \to B)$, we find no sentence letter which appears as a line itself; in this case, we're gonna assign F to A, B (which still appear as a part of other compound sentences).

Yet another example: $\{(A \to (B \leftrightarrow C)), -(C \to A)\}$. Since the rules for $-(C \to A)$ is non-branching, so we apply the rule 8 first.

1.	✓ $(A \to (B \leftrightarrow C))$	
2.	✓ $-(C \to A)$	
3.	C	Rule 8 to Line 2
4.	$-A$	Rule 8 to Line 2
5.	$-A$ $\quad$ $(B \leftrightarrow C)$	Rule 7 to Line 1

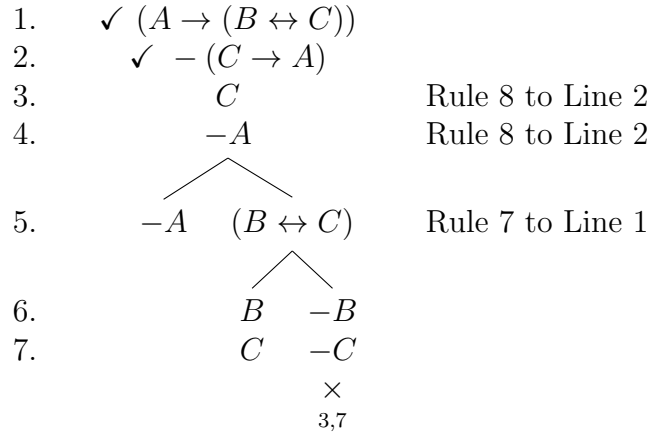
At this point, we notice that there's nothing to do for the left path because we decomposed all the compound sentences which are in the "root" positions as to the left path. It remains open no matter what happens on the right path. We call such an path (in which there's no contradiction and no compound sentence left) a *finished open path*. (Note that $(B \leftrightarrow C)$ belongs to the different path; we can't decompose it under $-A$ on the left path.) Thus, we can stop here (unless you're required to find all the open paths).

> **Rule of Thumb 3:** You can stop decomposing the tree once you find an open path (unless you're required to find all the open paths).

As in the previous example, we assign T to sentence letters which appear as a line itself on the open path (in this case, C) and assign F to those which don't appear as a line itself but as a component of other sentences (in

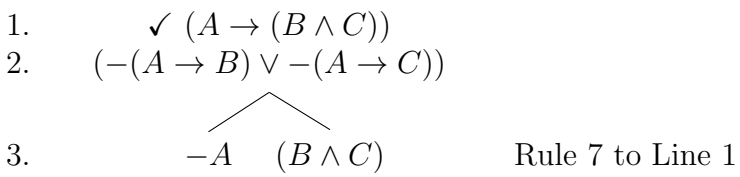
this case, A and B); thus, on this open path, we have truth-value assignments $A : F, B : F, C : T$ which make all the sentences in the initial list true.

According to Rule of Thumb 3 above, we can stop decomposing the tree here because we already know the set of sentences is satisfiable. Even so, let's keep decomposing (yet again, for pedagogical purpose).



It turns out that the tree has one more open path and it tells us that the set is going to be true when A is false, B is true, and C is true.

Last example: $\{(A \to (B \wedge C)), (-(A \to B) \wedge -(A \to C))\}$.



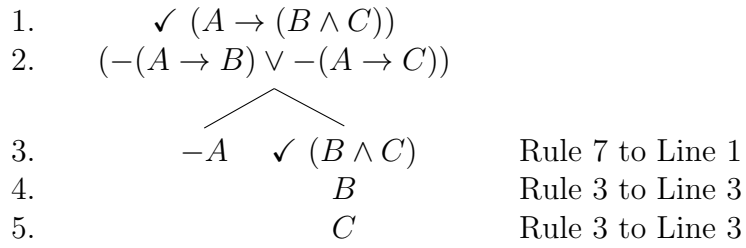
Here, we have two options: decompose $(-(A \to B) \vee -(A \to C))$ which is at the root position to both branches, or decompose $(B \wedge C)$ which is in a branch. Which one should we decompose first? The following is a rule of thumb.

Rule of Thumb 4:

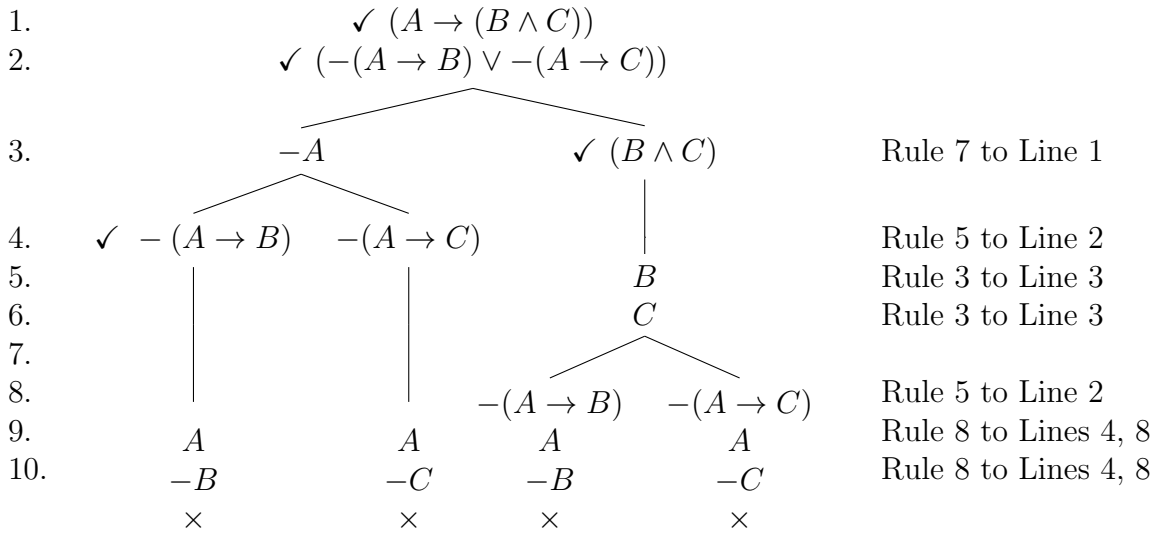
1. If a sentence in a branch is decomposed by a non-branching rule, decompose it first.

2. If a sentence in a branch is decomposed by a branching rule, decompose a sentence at the root position.

In this case, the sentence in the right branch is decomposed by a non-branching rule; so decompose it first.



Now the only decomposable sentence is $(-(A \to B) \vee -(A \to C))$. Since it is at the root position of two branches, make sure that you decompose it in both branches.



All paths are closed; thus, the set of the sentences are unsatisfiable.

1.9.3 Testing the Validity of a Sentence

To test the validity of a sentence, test the satisfiability of the denial of the sentence. If the tree is closed, the sentence is valid; if not, it's not. In the latter case, an open path gives you an interpretation in which the sentence is going to be false.

Example: Check the validity of $((C \to R) \to (-R \to -(C \wedge J)))$.

According to the above instruction, we should start our tree with the *denial* of the sentence.

1.	✓ $-((C \to R) \to (-R \to -(C \wedge J)))$	
2.	✓ $(C \to R)$	Rule 8 to Line 1
3.	✓ $-(-R \to -(C \wedge J))$	Rule 8 to Line 1
4.	$-R$	Rule 8 to Line 3
5.	✓ $--(C \wedge J)$	Rule 8 to Line 3
6.	✓ $(C \wedge J)$	Rule 2 to Line 6
7.	C	Rule 3 to Line 6
8.	J	Rule 3 to Line 6
9.	$-C$ $\quad$ R × $\quad$ × 7,9 $\quad$ 4,9	Rule 7 to Line 2

All the paths are closed; $((C \to R) \to (-R \to -(C \wedge J)))$ is valid.

Now, let's think about why you need to start with the *denial* of your sentence whose validity you're gonna check. First, recall the case of satisfiability. In checking the satisfiability of sentence(s), you start with sentence(s) itself (themselves). If your tree has at least one finished open path, your sentence is satisfiable; if not, it's not (namely, unsatisfiable). And if a tree for the denial of a sentence is closed, the denial of a sentence is unsatisfiable; and because the denial of any valid sentence is unsatisfiable, a sentence itself has to be valid. On the other hand, if a tree for the denial of a sentence has at least one finished open path, the denial of a sentence is satisfiable; namely, there's at least one interpretation which assigns F to a sentence. In such a case, a sentence cannot be valid of course.

One more example: Check the validity of $((L \vee (J \vee -K)) \wedge (K \wedge ((J \vee L) \to -K)))$.

1.	✓ $-((L \vee (J \vee -K)) \wedge (K \wedge ((J \vee L) \rightarrow -K)))$		
2.	✓ $-(L \vee (J \vee -K))$	$-(K \wedge ((J \vee L) \rightarrow -K))$	Rule 4 to Line 1
3.	$-L$		Rule 6 to Line 2
4.	✓ $-(J \vee -K)$		Rule 6 to Line 2
5.	$-J$		Rule 6 to Line 4
6.	✓ $--K$		Rule 6 to Line 4
7.	K		Rule 2 to Line 6

At this point, we notice that there's nothing to do for the left path because we decomposed all the sentences which are in the "root" positions as to the left path. The left path remains open no matter what happens to the rest. $((L \vee (J \vee -K)) \wedge (K \wedge ((J \vee L) \rightarrow -K)))$ is going to be false when J is false, K is true, and L is false because the only sentence letter which appears as a line itself on this open path is K.

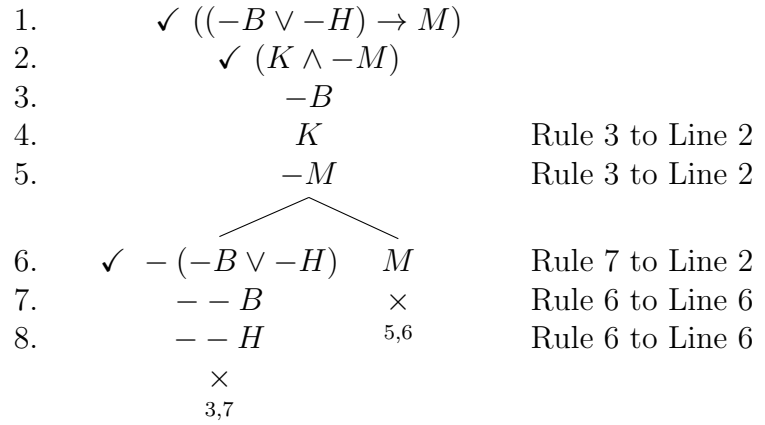
1.9.3.1 Problems

Do Problems 1.3.6 by the truth-tree method.

1.9.4 Testing the Validity of an Argument

> To test the validity of an argument, test the satisfiability of the premises and the denial of the conclusion. If the tree is closed, the argument is valid; if not, it's not. In the latter case, an open path gives you a counterexample to the argument.

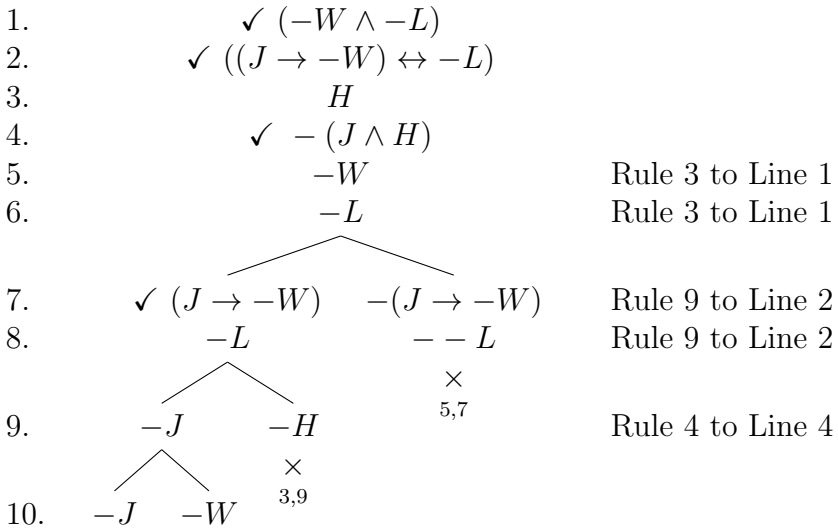
Example: Check the validity of the argument $\{((-B \vee -H) \rightarrow M), (K \wedge -M)\} \models B$.



It's closed; the argument is valid.

Let's think about why this works (especially why your initial list should have the *denial* of the conclusion of the argument) as its member. First, suppose your argument is $\{P_1, \ldots, P_n\} \models C$; in this case, your tree starts with the initial list $\{P_1, \ldots, P_n, -C\}$. If all the paths in your tree are closed, the initial list $\{P_1, \ldots, P_n, -C\}$ is unsatisfiable; in other words, there's no interpretation which assigns T to all the sentences in the list. Because the impossibility of assigning T to the denial of a sentence is the same as the impossibility of assigning F to a sentence itself, the above conclusion amounts to saying that there's no interpretation which assigns T to all the premises and F to the conclusion; this is exactly what an argument needs in order to be valid.

Another example: Check the validity of $\{(-W \wedge -L), ((J \rightarrow -W) \leftrightarrow -L), H\} \models (J \wedge H)$



We have two open paths according to both of which the arguments is invalid when H is true and J, L, W are false; and we say these truth-value assignments (H: T, $J.L.K$: F) is (or provides) a *counter-example* to the argument.

This works because the satisfiability of the sentences in the list means there's at least one interpretation which assigns T to the premises and F to the conclusion; this is exactly what we need for confirming the argument is invalid.

1.9.4.1 Problems

Do Problems 1.5.1 (2) by the truth-tree method.

1.9.5 Testing the Implication

Recall that an implication $\alpha \models \beta$ holds if and only if $\alpha \to \beta$ is valid (see Section 1.4.1). Thus, checking the truth of the implication $\alpha \models \beta$ amounts to that of the validity of $\alpha \to \beta$.

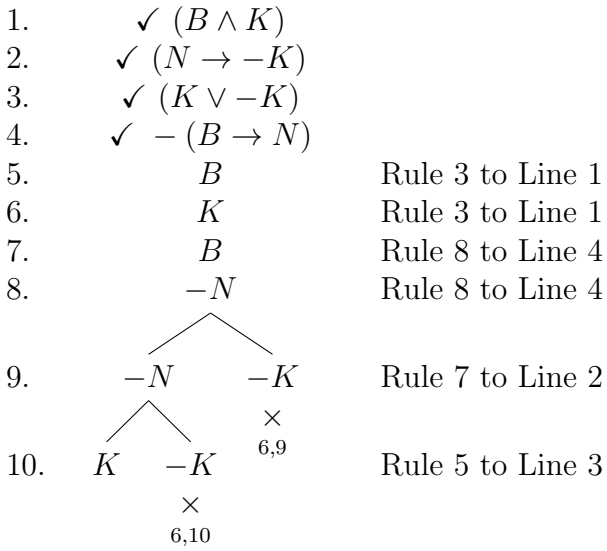
> To test the truth of an implication $\alpha \models \beta$, test the validity of $\alpha \to \beta$. If the tree is closed, the implication holds; if not, it's not. In the latter case, an open path gives us an instance where the implication fails.

Example: Check the truth of the implication $\{(-J \vee S), (S \rightarrow E)\} \models (J \rightarrow E)$. (Remember: the truth of a set of sentences amounts to that of the conjunction of the sentences.)

1.	✓ $-(((-J \vee S) \wedge (S \rightarrow E)) \rightarrow (J \rightarrow E))$	
2.	✓ $((-J \vee S) \wedge (S \rightarrow E))$	Rule 8 to Line 1
3.	✓ $-(J \rightarrow E)$	Rule 8 to Line 1
4.	✓ $(-J \vee S)$	Rule 3 to Line 2
5.	✓ $(S \rightarrow E)$	Rule 3 to Line 2
6.	J	Rule 8 to Line 3
7.	$-E$	Rule 8 to Line 3
8.	$-J$	Rule 5 to Line 4
	× 6,8	
9.	$-S$	Rule 7 to Line 5
	× 8,9	
	E	
	× 7,9	

The tree is close; the implication $\{(-J \vee S), (S \rightarrow E)\} \models (J \rightarrow E)$ actually holds.

Another example: $\{(B \wedge K), (N \rightarrow -K), (K \vee -K)\} \models (B \rightarrow N)$ (This time I start the tree by vertically aligning the sentences of the left side and the denial of the sentence of the right side. You might not want to adopt this way in the exam unless you can justify this based on the definitions of implication and argument.)



We have an open path according to which the implication fails when B, K are true and N is false.

1.9.5.1 Problems

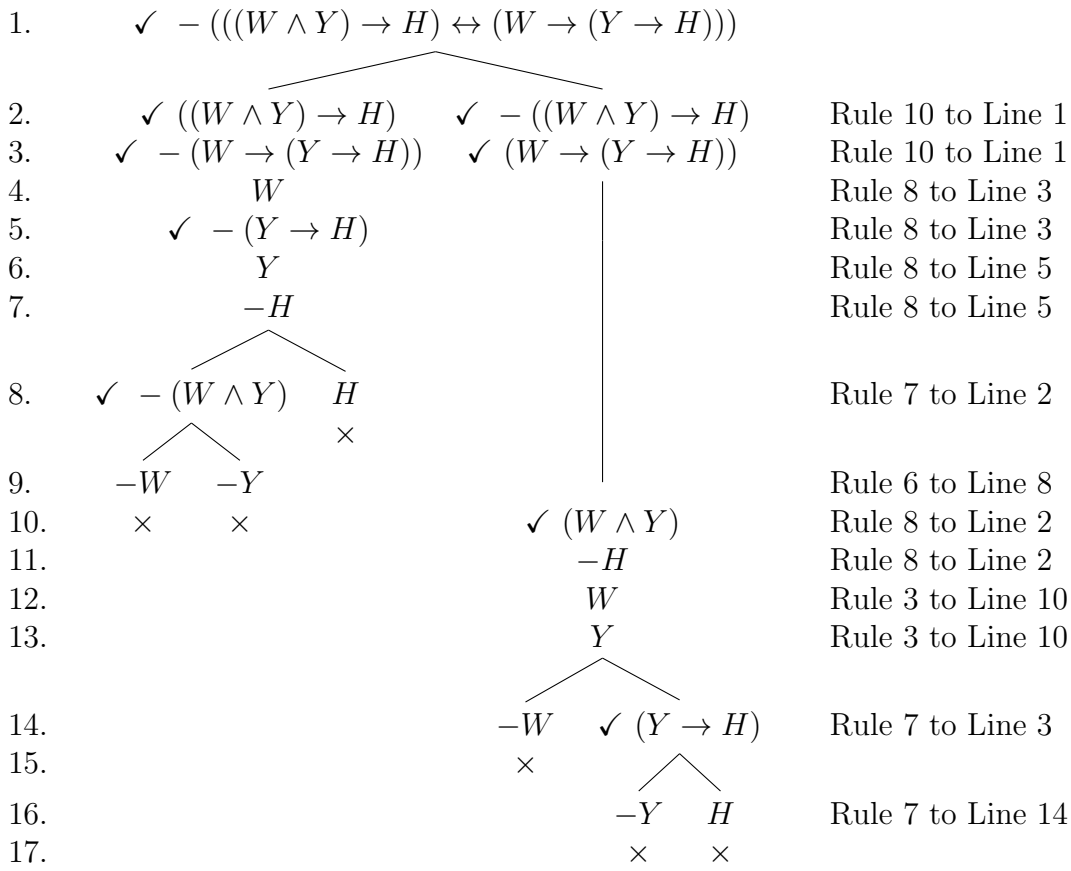
Do Problems 1.4.1.1 (2) by the truth-tree method.

1.9.6 Testing the Equivalence

Recall that an equivalence $\alpha \equiv \beta$ holds if and only if $\alpha \leftrightarrow \beta$ is valid (see Section 1.4.2). Thus, checking the truth of the equivalence $\alpha \equiv \beta$ amounts to that of the validity of $\alpha \leftrightarrow \beta$.

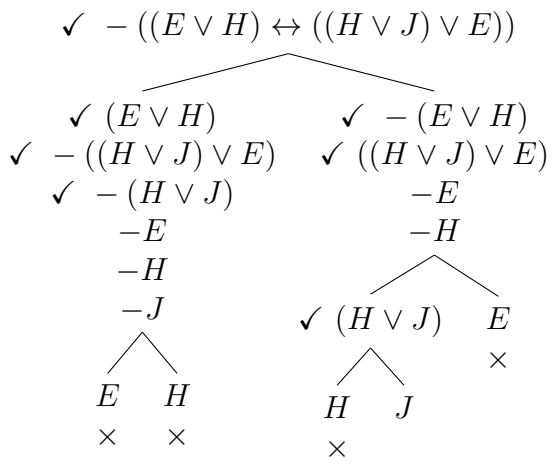
> To test the truth of an equivalence $\alpha \equiv \beta$, test the validity of $\alpha \leftrightarrow \beta$. If the tree is closed, the equivalence holds; if not, it's not. In the latter case, an open path gives us an instance where the equivalence fails.

Example: Check the truth of $((W \wedge Y) \rightarrow H) \equiv (W \rightarrow (Y \rightarrow H))$.



All paths are closed; the implication holds.

Another (and last) example: $\{(E \vee H) \equiv ((H \vee J) \vee E)\}$. This time I omit the justifications and other annotations.



One path remains open; the equivalence doesn't hold when E, H are false, and J is true.

1.9.6.1 Problems

Do Problems 1.4.2.1 (2) by the truth-tree method.

Chapter 2

Language L_2

We've studied the language L_1 which has as its vocabulary some logical connectives ($-$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$), sentence letters (the upper case alphabets with or without subscripts), and some auxiliary symbols (brackets (and), sentence variables α, β, γ ... (the lower Greek letters) with or without subscripts). Although it's powerful enough to capture (or formalize) a variety of situations, it's quite powerless to formalize an argument like the following.

All humans are mortal.
Socrates is a human.

Socrates is mortal.

Of course, we can still formalize the above argument in L_1 by introducing some sentence letters each of which refers to each statement in the argument (e.g. H means all humans are mortal, S means Socrates is a human, etc.). However, in order to judge whether an argument is valid or not, we have to capture the internal structures of sentences of the argument in our formal language and relate these structures to show the validity of an argument. If we simply formalized the above argument with sentence letters, there would be no way to make connection between them.

Then, the questions is this: How to capture the internal structures of sentences? First, you immediately notice that these sentences above share some vocabulary: human, mortal, and Socrates. Thus, if we have some ways to express these in our formal language, we'd be able to express the relation between sentences. Second, more importantly, there's an occurrence of a determiner "all" in the first premise. The first premise is not talking about

this or *that* human; it's talking about *humans in general*. Thus, if we have some way to express this kind of generality in our formal language, we'd be able to express a *general* situation, not *this* or *that* situation. In order to implement these in our formal language, we're gonna introduce the following new vocabulary: *names*, *predicates*, *variables*, and *quantifiers*.

2.1 Syntax of L_2

2.1.1 The Vocabulary of L_2

Our new language L_2 is an extension of our familiar language L_1. In L_2, we have all we have in L_1. We're just gonna add a small number of new vocabulary to L_1. However, this addition will make a huge difference.

Logical Symbols

Brackets: (,).

Connectives: $-, \wedge, \vee, \rightarrow, \leftrightarrow$.

Quantifiers: $\forall$ (universal quantifier), $\exists$ (existential quantifier).

Variable: x with or without subscripts.

Non-Logical Symbols

Sentence Letters: $A, B, C, \ldots$ with or without subscript. E.g., A_1, B_2, Z_{1028}.

Names: $a, b, c, \ldots, w$ with or without subscripts.

Predicate Letters: $A, B, C, \ldots$ with or without super- and subscripts.

Auxiliary Symbols

Sentence variables $\alpha, \beta, \gamma, \ldots$ (the lower Greek letters) with or without subscript.

Let's take a look at the new elements one by one.

2.1.1.1 Names

A *Name* is used for referring to an object, it must refer to one and only one object; but different names may refer to the same object.

In real life, one name can refer to many different objects; the name "David" can refer to millions of Davids. In this sense, the reference of a name in real life is vague. In order to avoid this kind of vagueness, we stipulate that in L_2 a name must refer to one and only one object; a cannot refer to the natural number 1 and Socrates at the same time. Moreover, a name must refer to something; an empty name (a name without reference) is not allowed. On the other hand, different names can refer to the same object. Different names a and b may refer to the same object, say, Socrates. (You may have a nickname besides your real name.)

2.1.1.2 Predicates

A *predicate* is used for expressing a property of an object or a relation between objects. An n-place predicate followed by n names (not variables!) forms a sentence.

Sometimes a predicate has a superscript, like H^1 or L^2, for making clear how many names are required to form a sentence. Thus, for example, L^2ab counts as a legitimate L_2-sentence whereas L^2a doesn't.

Let s be a name for Socrates, H^1 be a predicate expressing a property of being a human. Thus, Hs means "Socrates is a human". If a predicate is followed by one name or variable, it's called a *one-place* predicate.

We can think of n-place predicates in general for some natural number n. For example, a two-place predicate L^2 may mean, with names a and b, "a loves b" (strictly speaking, the names a and b in "a loves b" should be written as "an object referred to by a" and "an object referred to by b" respectively, but I take the liberty of using this kind of omissions), and a three-place predicate B^3 may mean, with names a, b, c, "a is between b and c".

What if we take 0 for n? In this case, in order for a 0-place predicate to be a sentence, 0 names must follow the predicate letter. What does this mean? Yes, a 0-place predicate itself is a sentence; in other words, a 0-place

predicate is nothing but a sentence letter.

n-place predicates followed by n names are called *atomic sentences*. Together with their denials, they are called *literals*. E.g. A, $-B$, Fa, $-Lbc$ are all literals.

2.1.1.3 Quantifiers

Here comes our main dish in this section: the quantifiers $\forall$ and $\exists$.

Conceptually, these are not hard to understand; $\forall$ means "'all', $\exists$ means "some". That's it. However, some words of caution are in order, especially for $\exists$. Let's take a look at these with some examples.

First, a general remark on both quantifiers; the quantifiers are *always* used with variables, and in general, followed by predicates with those variables. For example, with some predicate H^1, we write $\forall x H^1 x$ or $\exists x H^1 x$.

Now, let H^1 in the above be a predicate which means "...is a human"; then, $\forall x Hx$ means "all are humans", $\exists x Hx$ means "some are humans".

Here, in our language L_2, "some" doesn't necessarily means/refers to plural objects; it can just refer to one object. However, as in the case of names, it has to refer to at least one object. Thus, the stricter translation of $\exists x \ldots x \ldots$ would be "there is at least one object which is ...".

To recap:

$\forall x \ldots x \ldots$: All objects are .... $\exists x \ldots x \ldots$: There is at least object which is ....[1]

Lastly, considering that "all are ..." can be paraphrased as "there's nothing which is not ...", and "some are ..." as "not all are not ...", we have the following equivalences.

$\forall x$ can be written as $-\exists x-$. $\exists x$ can be written as $-\forall x-$.

[1] Of course, there are more than one way to express (or translate) $\forall$ and $\exists$ in our ordinary language (in this case, in English). We'll take a look at a few more ways to express or translate $\forall$ and $\exists$ in English in Section 2.2.

2.1.2 The Formation Rules of L_2-Sentences

As was mentioned earlier, the language L_2 is an extension of the language L_1. Thus, all we have to do here is to add a few rules to the formation rules of L_1.

1. Every sentence letter is a sentence of L_2.[2]
2. If P^n is an n-place predicate, and $a_1, \ldots, a_n$ are names, then $P^n a_1 \ldots a_n$ is a sentence of L_2.
3. If α and β are sentences of L_2, so are $-\alpha, (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), (\alpha \leftrightarrow \beta)$. Sentences of these forms are called *compound sentences*.
4. If S is an L_2-sentence with a name n in it, and a variable x doesn't appear in it, the result of replacing n with x and prefixing $\exists x$ or $\forall x$ is a sentence of L_2.
5. Nothing else is a sentence of L_2.

In forming quantified sentences (namely, sentences with the quantifier(s)), a special care must be taken in some cases. In the cases of simple sentences like Fa or Lab, the formation of the quantified sentences from them would cause no trouble; just replace a name with a variable and put one of the quantifiers in front of it. For example, by replacing a in Fa (or in Lab) with x and putting $\forall x$ in front of it, we get $\forall x Fx$ (or $\forall x Lxb$). However, if we want to make a quantified sentence from a compound sentence like $(Fa \rightarrow Lab)$, there are several options as follows.

1. $\forall x(Fx \rightarrow Lab)$
2. $\forall x(Fx \rightarrow Lxb)$
3. $(\exists x Fx \rightarrow Lab)$
4. $(\exists x Fx \rightarrow \exists x Lxb)$[3][4]

[2]Note that sentence letters can be thought of as 0-place predicates. Thus, we can do without sentence letters at all. However, as a convention, we keep using the term "sentence letter(s)".

[3]This list is not meant to be exhaustive; there are lot more options.

[4]The formation processes of the sentences are as follows. In the sentence 1, by replacing

In all the cases above, the variable x is said to be *bound* (or *governed*) by the quantifiers $\forall x$ or $\exists x$. And it's said that the *scope* of the quantifier $\forall x$ in the sentences 1 and 2 is the whole sentence $(Fx \rightarrow Lab)$ or $(Fx \rightarrow Lxb)$ whereas the scope of $\exists x$ in the sentence 3 is just Fx. Thus, it's okay to use the same variable x in the sentence 4 because the scopes of the two existential quantifiers don't overlap. (The first $\exists x$ just binds Fx whereas the second binds Lxb.)

What if we omit the second $\exists x$ in the sentence 4? In that case, x in Lxb stops being bound (remember, the first $\exists x$ just binds Fx), and said to be *free*. And then, $(\exists xFx \rightarrow Lxb)$ stops being a sentence.

2.1.2.1 Examples of L_2-Sentences

F^1a, L^2ab, $-Fa$, $(Fa \wedge Lab)$, $\exists xFx$ (from Fa), $\exists xFx \wedge Lab$, $\forall Lax$ (from Lab), $\forall xLxx$ (from $\forall xLax$), $\exists xLxx$ (from Laa), $\exists xLxa$ (from Laa), $\exists x(Fx \wedge Lxb)$.

2.1.2.2 Examples of Non-Sentences

F^1ab (F^1 is a one-place predicate so it has to be followed by one names), L^2a (L^2 is a two-place predicate so it has to be followed by two names), F^1x (in terms of the number of names following the predicate letter, there's no problem; however, the variable x is not bound hence this is not a sentence), $(\exists xFx \wedge Lxb)$ (x in Lxb is not bound hence $(\exists xFx \wedge Lxb)$ is not a sentence).

2.2 Translation from/into L_2-Sentences

Before moving on to the semantics (i.e. interpretations) of L_2-sentences, let's see how L_2-sentences are translated into English and how English sentences are translated into L_2-sentences.

Let's start with simple examples: sentences with a quantifier and a predicate.

---

a in Fa with x and putting $\forall x$ in front of the whole sentence, we get the quantified sentence. On the other hand, in the sentence 2, by replacing both a's in Fa and Lab with x and putting $\forall x$ in front of the whole sentence, we get the quantified sentence as well. In the sentence 3, by replacing a in Fa with x and putting $\exists x$ in front of Fx, we get the quantified sentence (and this is not equivalent to $\exists x(Fx \rightarrow Lab)$!). Finally, by replacing a in both Fa and Lab and putting $\exists x$ in front of *each* atomic sentences, we get the quantified sentence.

2.2.1 Translating L_2-sentences with one predicate

As was mentioned in footnote 1, the same quantified sentence can be translated into various English sentences. For example, with a one-place predicate N which means "... is a number", $\forall xNx$ can be translated as

- All are numbers.
- Everything is a number.
- Anything whatsoever is a number.

Similarly, $\exists xNx$ can be translated as

- There is at least one number.
- There are numbers.
- Some are numbers.
- Something is a number.
- Numbers exist.

If we replace Nx with $-Nx$, the English translations would be

$\forall x - Nx$: Everything is not a number.
$\exists x - Nx$: Some are not numbers.

Moreover, the above translations can be paraphrased as "there's no number" and "not everything is a number" respectively. And because "there's no number" and "not everything is a number" are just the negations of "there are numbers" and "everything is a number", these are also expressed as $-\exists xNx$ and $-\forall xNx$ respectively.

To recap:

For some one-place predicate P,

$\forall xPx$: All are P.
$\exists xPx$: Some are P.
$\forall x - Px$; $-\exists xPx$: Everything is not P; there's no P.
$\exists x - Px$; $-\forall xPx$: Some are not P; not everything is P.[5]

[5]From the above, we also notice that $\forall x- \equiv -\exists x$ and $\exists x- \equiv -\forall x$; this is the direct

2.2.2 Translating L_2-sentences with two predicates

So far, we've been dealing with L_2-sentences in which just one predicate appears. However, in many cases, we need more than one predicate to express what we want to express. For example, in order to express "all humans are mortal", we need two predicates one of which is "... is a human" and the other of which is "... is mortal". And then, we somehow "bind" them with the universal quantifier.

To translate the "all ... are ..."-type of sentences into L_2, the following four forms (or templates) come in handy.

For some one-place predicates P and Q,

1. **All P are Q:** $\forall x(Px \to Qx)$.
2. **No P are Q:** $\forall x(Px \to -Qx)$; $-\exists x(Px \wedge Qx)$.
3. **Some P are Q:** $\exists x(Px \wedge Qx)$.
4. **Some P are not Q:** $\exists x(Px \wedge -Qx)$; $-\forall x(Px \to Qx)$.

For example, with one-place predicates A and D which mean "... is an apple" and "... is delicious" respectively, "all apples are delicious", "no apples are delicious", "some apples are delicious", and "some apples are not delicious" are translated as $\forall x(Ax \to Dx)$, $\forall x(Ax \to -Dx)$ (or $-\exists x(Ax \wedge Dx)$), $\exists x(Ax \wedge Dx)$, and $\exists x(Ax \wedge -Dx)$ (or $-\forall x(Ax \to Dx)$) respectively.

Note that in the above the universal quantifier $\forall$ is always accompanied by $\to$, and the existential quantifier $\exists$ with $\wedge$. The moral here is

If your translation has the forms $\forall x(\ldots \wedge \ldots)$ or $\exists x(\ldots \to \ldots)$, there may be something wrong with it.

For example, if you translate "All apples are delicious" as $\forall x(Ax \wedge Dx)$, your translation actually means that everything in this world is a delicious apple, which is quite a wild statement.

2.2.2.1 Many faces of $\exists$

As we've seen a number of times in the above, the same L_2-sentence can be translated into English in many ways. For example, let's take a look at the

results of the other equivalences $\exists x \equiv -\forall x-$ and $\forall x \equiv -\exists x-$. See Section 2.1.1.3.

following sentences.

- There is a delicious apple.
- There is at least one delicious apple.
- Some apples are delicious.
- At least one apple is delicious.

As you instantly notice, these sentences are translated as the same L_2-sentence, $\exists x(Ax \wedge Dx)$, and the sentences are so simple that you may think that you're not gonna have any trouble in translating such English sentences. However, in more complicated cases, you may become unsure whether you should use $\forall$ or $\exists$ in translating some English sentences. The following would give you some guide.

> English expressions "there exist(s)/is/are", "some", "at least one" and "a/an" are translated with the existential quantifier $\exists$.[6]

And as a general translation tip,

> When you're asked to translate an English sentence to an L_2-sentence, first try to translate your English sentence into another English sentence which is close to one of the four templates above.

2.2.3 Translating Noun Phrases

Let's think about translating the following sentence: All red apples are delicious. Here, unlike the previous examples, we need two predicates in order to express a noun phrase "red apple". In such cases, all we need is just combine two predicates with $\wedge$.

> When you need more than one predicate to express a noun phrase, just combine those predicates with $\wedge$.

Thus, by introducing yet another one-place predicate R which means "... is red", "all red apples are delicious" are translated as $\forall x((Rx \wedge Ax) \rightarrow Dx)$.

[6] However, a special attention must be paid to the cases where "a/an" is used for expressing something general. See Section 2.2.5.1.

2.2.3.1 Problems

Translate the following into L_2-sentences.

1. There's no delicious red apple.
2. At least one red apple is delicious.
3. Not all red apples are delicious.

2.2.4 Some Translation Practices

Let's translate the following English sentence into L_2. Here, we're given the following predicates: N ("... is a number"), P ("... is prime"), and O ("... is odd").

Every prime number is odd.

As we've seen in the previous section, the noun phrase "prime number" can be expressed with the predicates P and N. And using one of the four template, it's translated as

$$\forall x((Px \wedge Nx) \rightarrow Ox).$$

Obviously, the above statement is false because there's a prime number which is not odd, namely 2. How can we amend the statement? On way is to add an extra predicate "other than 2" to the noun phrase.

Every prime number other than 2 is odd.

The above can be paraphrased as "Every object which is a number, prime, and not 2, is odd". Thus, by introducing a new predicate T which means "... is 2", the above sentence is translated as

$$\forall x((Px \wedge Nx \wedge -Tx) \rightarrow Ox).$$

However, the above is not the only amendment to the false statement. We can amend the statement by rewriting it as "Every prime number is odd or 2". In this case, we need to use compound phrases in the subject

and complement.

$$\forall x((Px \wedge Nx) \rightarrow (Ox \vee Tx)).$$

Although these L_2-sentences look different, they capture the very same situation. And actually, they are logically equivalent.

2.2.4.1 Problem

Show that $\forall x((Px \wedge Nx \wedge -Tx) \rightarrow Ox)$ and $\forall x((Px \wedge Nx) \rightarrow (Ox \vee Tx))$ are logically equivalent by transforming the one into the other using $(\alpha \rightarrow \beta) \equiv (-\alpha \vee \beta)$, $-(\alpha \wedge \beta) \equiv (-\alpha \vee -\beta)$, and $-(\alpha \vee \beta) \equiv (-\alpha \wedge -\beta)$.

2.2.5 Some Translation Tips

In this section, we're gonna take a look at how we translate some English words/phrases into L_2-sentences.

2.2.5.1 If . . . , then . . . .

Basically, if you come across a sentence which has the if-then structure, you simply translate it as a conditional. For example, the sentence "if Alma is hungry, she eats something" is translated as $(Ha \rightarrow \exists xEax)$ (here, the name a refers to Alma, and the two predicate Hx and Exy mean "x is hungry" and "x eats y" respectively). However, in some cases, special care should be taken. Let's think about the following sentence.

If an apple is red, it is delicious.

In the above, "it" in the consequent of the conditional refers back to "an apple" in the antecedent. Thus, we need to make the connection between "it" and "an apple" explicit. But how? We use one of the quantifiers.

The quantifiers can make such connections explicit. For example, "an apple is red" can be expressed as $\exists x(Ax \wedge Rx)$. This sentence literally means that there exists at least one object which is an apple and red. Then, on first sight, it seems that the sentence is translated as $\exists x((Ax \wedge Rx) \rightarrow Dx)$. However, this doesn't capture the meaning of the sentence properly. the sentence doesn't merely talk about this or that red apple's deliciousness; rather,

it talks about a red apple in general is delicious. Thus, the proper translation is

$$\forall x((Rx \wedge Ax) \rightarrow Dx).$$

2.2.5.2 Only

There's a handy template for translating sentences in which the word "only" appears.

Only P are Q: $\forall x(Qx \rightarrow Px)$.

Thus, with this template, "only red apples are delicious" is translated as

$$\forall x(Dx \rightarrow (Rx \wedge Ax)).$$[7]

2.2.5.3 Unless

The words "unless" is simply interpreted as "if not". Thus, "Q unless P" is interpreted as "if not P, then Q". For example, "An apple is not delicious unless it's red" is interpreted as "if it's not red, an apple is not delicious", and then translated as $\forall x(-Rx \rightarrow -(Ax \wedge Dx))$. And this suggests that if an apple is delicious, it should be red; thus, we have another translation: $\forall x((Ax \wedge Dx) \rightarrow Rx)$.[8]

[7] Why can't we translate this as $\forall x((Rx \wedge Ax) \rightarrow Dx)$? In order to answer this question, first let's think about a quantified conditional.

In general, in a quantified conditional $\forall x(Px \rightarrow Qx)$, the cases (or things) which is said to be P are among the cases (or things) which are said to be Q; but the reverse doesn't generally hold. For example, in the conditional $\forall x(Rx \wedge Ax) \rightarrow Dx$, red apples are among things which are said to be delicious. However, what is delicious is not necessarily a red apple. The conditional $\forall x(Rx \wedge Ax) \rightarrow Dx$ implies that there might be some other things than red apples which are delicious.

On the other hand, the statement "only red apples are delicious" means that what are delicious are red apples, and nothing else is delicious; what are delicious are among red apples (however, there might be some red apples which are not delicious). Thus, the statement is translated as $\forall x(Dx \rightarrow (Rx \wedge Ax))$, not $\forall x((Rx \wedge Ax) \rightarrow Dx)$.

[8] From this, you may notice that $(P \rightarrow Q)$ is equivalent to $(-Q \rightarrow -P)$; the latter (resp. the former) is called the *contrapositive* of the former (resp. the latter).

Furthermore, "unless" can be interpreted as "or". The reason is as follows. From the above remark, "Q unless P" is translated as $(-P \to Q)$. This conditional can be written as $(--P \vee Q)$ (using $(\alpha \to \beta) \equiv (-\alpha \vee \beta)$). Therefore, "$Q$ unless P" $\equiv (-P \to Q) \equiv (P \vee Q) \equiv$ "P or Q".

2.2.5.4 Any

The word "any" is perhaps the most troubling one to translate because it sometimes means "some", sometimes means "every".

Unfortunately, there's no etched-in-stone way to figure out in which sense "any" is used. However, there are some rules of thumb.

If any . . . : If there's at least one . . . , then . . . $\Rightarrow \exists x(\ldots x \ldots) \to \ldots$

Ex. If anyone can do it, then Alma can do it.
$\Rightarrow$ If there's at least one who can do it, then Alma can do it.
$\Rightarrow (\exists x Dx \to Da)$.

Not any . . . : There's no . . . $\Rightarrow -\exists x(\ldots x \ldots)$.

Ex. Not any can do it.
$\Rightarrow$ There's no one who can do it.
$\Rightarrow -\exists x Dx$.

Any . . . : Every . . . $\Rightarrow \forall x(\ldots x \ldots)$.

Ex. Anything Alma eats is . . .
$\Rightarrow$ Everything Alma eats is . . .
$\Rightarrow \forall x(Eax \to \ldots)$.

Unfortunately again, none of the above rules of thumb doesn't seem to be readily applied to a sentence like "if Alma eats anything, then she eats an apple". In this case, you need to interpret your sentence in both ways and decide which interpretation is more natural.

First, let's interpret "any" as "every"; then, we have "If Alma eats everything, then Alma eats an apple", which sounds a little bit too wild. On the other hand, if we interpret "any" as "some", we have "If Alma eats something, then Alma eats an apple", which sounds more natural.

(You may also wonder how you should translate "Alma eats an apple". Here, remember that in many cases "a/an" is interpreted as "at

least one" in L_2. Thus, "Alma eats an apple" is interpreted as "There's at least one apple which Alma eats", and then translated as $\exists x(Dax \wedge Ax)$.)

2.2.6 Problems

1. Translate the following sentences. In translating them, use the following names and predicate: h: Holmes; m: Moriarty; Cxy: x can catch y.

1. Holmes can catch anyone who can catch Moriarty.
2. Holmes can catch anyone whom Moriarty can catch.
3. Holmes can catch anyone who can be caught by Moriarty.
4. If anyone can catch Moriarty, then Holmes can.
5. If everyone can catch Moriarty, then Holmes can.
6. Anyone who can catch Holmes can catch Moriarty.
7. No one can catch Holmes unless he can catch Moriarty.

2. Formalize the following argument. In formalizing it, use the following predicates: Ax: I avoid x; Bx: x is carnivorous; Cx: x is a cat; Dx: I detest x; Ex: x is in this house; Hx: x is a kangaroo; Kx: x kills mice; Lx: x loves to gaze at the moon; Mx: x prowls at night; Nx: x is suitable for pets; Rx: x takes to me. Furthermore, in the argument, we are talking only about animals. (Thus, we can formalize the argument without a predicate which means "... is an animal".)

> The only animals in this house are cats.
> Every animal is suitable for pets if it loves to gaze at the moon.
> When I detest an animal, I avoid it.
> No animals are carnivorous unless they prowl at night.
> No cat fails to kill mice.
> No animals ever take to me except what are in this house.
> Kangaroos are not suitable for pets.
> None but carnivora kill mice.
> I detest animals that do not take to me.
> Animals that prowl at night always love to gaze at the moon.
> ---
> I avoid kangaroos.

(The above argument is taken from a book by Charles Lutwidge

Dodgson, aka Lewis Carroll, who is needless to say the author of *Alice in Wonderland*, and incidentally taught logic at Oxford.)

2.3 Semantics of L_2

In Section 1.2, we learned how to assign truth values to an L_1-sentence; the truth values of an L_1-sentence is determined by those of sentence letters in it. The truth-value assignments to an L_2-sentence is carried out in a similar fashion as in the cases of L_1-sentences; the truth values of an L_2-sentence is determined by its components. However, in the cases of L_2-sentences, we have more components besides sentence letters as "building blocks" or "truth makers" of L_2-sentences; predicates. And recall that a predicate is nothing but a sentence; it has its own truth value which is determined by objects the names in the predicate (remember: in general, an n-place predicate should have n names) refer to. Schematically, this procedure is depicted as follows.

$$\begin{array}{l} \text{Truth values of an } L_2\text{-sentence} \\ \xrightarrow{\text{are determined by}} \text{Truth values of predicates in the sentence} \\ \xrightarrow{\text{are determined by}} \text{References of names in the predicate.} \end{array}$$

Thus, in order to assign truth values to an L_2-sentence, we need to start with assigning objects to names.

2.3.1 Assignment of an Object to a Name

In order to assign an object to a name, first we need to decide what kind of objects we're talking about. This collection of objects we're talking about is called the *domain*. The domain can be any collection but it cannot be empty; it has to have at least one objects in it. Other than that, there's no restriction as to how we set up our domain or the *universe of discourse*. The domain may be a collection of numbers, of people, of numbers and people, or of anything ever imaginable.

Now, let's suppose that our domain is comprised of three numbers $1, 2, 3$; we write this as $\{1, 2, 3\}$. Once we set up the domain, we can assign an object of the domain to a name. For example, if we have a name a, we assign *one and only one* member of the domain; the same name cannot refer to two different objects in the domain although it may be the case that two

different names refer to the same object in the domain (see Section 2.1.1.1 for a bit more details). And a name has to refer to something in the domain; namely, an empty name is not allowed (i.e., a name without reference). The object assigned to a name is called an *extension* or *valuation* of that name and denoted as $Ext(a)$ or $v(a)$; and we say "the name *refers to* the assigned object". Thus, the extensions (or valuations) for the names a, b, c may be

$a : 1; b : 2; c : 3$
(in this set-up, we say "a refers to 1, b to 2, and c to 3 respectively")

or

$a : 1; b : 1; c : 3$
(a and b refer to the same object, namely 1)

or

$a : 1$
(there's no names for 2 and 3; yes, an object in the domain doesn't have to be named).

2.3.2 Assignment of Truth Values to a Predicate

Once we assigned objects to names in a predicate, we're ready to assign truth values to the predicate. And how we assign truth values to a predicate depends on how many names the predicate in question has. Let's start with the simplest case: a one-place predicate.

2.3.2.1 Assignment of Truth Values to a One-Place Predicate

Recall that a one-place predicate is that which takes one name, and this is sometimes indicated by the superscript (for more detail, see Section 2.1.1.2); for example, the superscript 1 in F^1 indicates that this predicate takes one name, and with some name a, F^1a means that an object referred to by a has the property F; or more simply, F^1a means that a is F. Thus, intuitively, if an object referred to by the name a actually has the property F (or simply, if a is actually F), the sentence F^1a is true.

A sub-collection (which is usually called a *subset*) of the domain the members of which have the property F is called the *extension* of F. For example, if our domain is $\{1, 2, 3\}$, its subsets $\emptyset$ (or Λ; both are read as "the empty set"), $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 2\}$, $\{2, 3\}$, $\{1, 3\}$, and $\{1, 2, 3\}$ (namely, the domain itself) are all candidates for the extension of F and denoted as $Ext(F^1)$. (Note that in a set (or subset), the order doesn't matter; $\{1, 2\}$ is the same as $\{2, 1\}$, $\{1, 2, 3\}$ is the same as $\{2, 1, 3\}$, $\{3, 2, 1\}$, etc.)

For example, if the extension of F is $\{1, 3\}$, Fa (here, we omit the superscript 1) is going to be true when a refers to 1 or 3; otherwise (in this case, when a refers to 2), it's going to be false.

Note that the extension of a one-place predicate can be empty; for example, if our domain is the set of people, and if the one-place predicate Rx means "x is a reptile", obviously there's nothing in the extension of R. In this case, we write the extension of R as $\emptyset$ and say "nothing in the domain is a reptile".[9]

2.3.2.2 Assignment of Truth Values to a Two-Place Predicate

A two-place predicate is that which takes two names and this is sometimes indicated by the superscript 2; for example, the superscript 2 in L^2 indicates that this predicate takes two names, and with some names a and b, L^2ab means that an object referred to by a has the relation L to an object referred to by b; or more simply, L^2ab means that a has the relation L to b. Thus, intuitively, if an object referred to by the name a actually has the relation L to an object referred to by b (or simply, if a actually has the relation L to b), the sentence L^2ab is true.

The rough description above is almost the same as that for a one-place predicate. However, the form of the extensions for a two-place predicate is significantly different from that of a one-place predicate; the extension of a two-place predicate is a set comprised of *ordered pairs* of the members of the domain, not just a subset of the members of the domain.

An *ordered pair* is just two members of the domain enclosed with the angle brackets $\langle$ and $\rangle$. For example, $\langle 1, 2\rangle$, $\langle 2, 3\rangle$, and $\langle 3, 1\rangle$ are all ordered pairs. Note that, for some members x, y of the domain, $\langle x, y\rangle$ is *not* the same as $\langle y, x\rangle$ unless x, y are the same.

[9] You cannot write this empty extension as $\{\emptyset\}$; $\{\emptyset\}$ *does* have a member, namely $\emptyset$, and consequently, $\{\emptyset\}$ is *not* empty.

The reason why the extension of a two-place predicate has to be a set of ordered pairs, not just a subset of the domain, comes from the following observation.

Let Lab (we omit the superscript 2 here) mean "a loves b". if a and b referred to 1 and 2 respectively, and if the extension of L were just comprised of a subset $\{1, 2\}$ of the domain, Lba as well as Lab would be true with this extension because $\{1, 2\}$ is the same $\{2, 1\}$. However, the sentence "a loves b" (which is meant by Lab) is not the same "b loves a" (Lba). Therefore, we have to differentiate these two sentences. For this reason, we need to use a set of ordered pairs for the extension of a two-place predicate.

Now, let's take a look at a simple example. If the extension of L is $\{\langle 1, 2\rangle, \langle 2, 1\rangle\}$, Lab is going to be true when a refers to 1 and b to 2, or a to 2 and 1; otherwise (for example, a refers to 1 but b to 3), it's going to be false. In either case, if Lab is true, a and b love with each other.

As in the case of a one-place predicate, the extension of a two-predicate can be empty; for example, if the extension of L is empty, no one loves anyone under such an interpretation.

2.3.3 Examples

Problems concerning the interpretation of L_2-sentences roughly fall into the following two types.

1. Given an interpretation, determine whether some sentences are true or not under that interpretation.
2. Given a sentence, construct an interpretation under which the given sentences are true (or false).

In this section, we're gonna take a look at some examples of these two types of problems.

2.3.3.1 Determining the Truth Values of Sentences under a Given Interpretation

Suppose that we're given the following two interpretations I_1 and I_2.

	I_1	I_2
Domain:	$\{1, 2, 3\}$	$\{1, 2, 3\}$
Extension of F^1:	$\{1\}$	$\emptyset$
Extension of G^1:	$\{2\}$	$\{2\}$
Extension of L^2:	$\{\langle 1, 1\rangle, \langle 1, 2\rangle\}$	$\emptyset$
Extension of a:	1	1
Extension of b:	2	2

Let's determine whether the following sentences are true or not under these interpretations.

1. Fa	5. Laa	9. $\exists x{-}Fx$	13. $\forall x(Fx \to Gx)$
2. $(Fa \wedge Gb)$	6. $(Lba \to Laa)$	10. $\exists x(Fx \wedge Gx)$	14. $\forall x(Fx \leftrightarrow Gx)$
3. $-Fb$	7. $(Laa \to Lba)$	11. $(\exists xFx \wedge \exists xGx)$	
4. $(Fb \to Gb)$	8. $\exists xFx$	12. $\forall xFx$	

1. Fa
   This sentence is true under I_1 because the reference of a, namely 1, is in the extension of F. On the other hand, the sentence is false under I_2 because there's nothing which is F in I_2.

2. $(Fa \wedge Gb)$
   The sentence is a conjunction. Therefore, in order for it to be true, both conjuncts have to be true. Under I_1, the references of a and b are 1 and 2 respectively, and 1 is in the extension of F and 2 in the extension of G. Thus, since both conjuncts are actually true under I_1, the sentence is true under I_1. On the other hand, as we know from the case of Fa, nothing is F in I_2, and consequently, Fa is false under I_2. Thus, the sentence is false under I_2.

3. $-Fb$
   The sentence is a negation. Thus, in order for it to be true, Fb has to be false, which means that the reference of b shouldn't be in the extension of F. In I_1 and I_2, the reference of b, namely 2, is not in the extension of F. Therefore, the sentence is true under both I_1 and I_2.

4. $(Fb \to Gb)$
   The sentence is a conditional. Thus, in order for it to be true, at least one of the following has to be the case: Fb is false or Gb is true. In

I_1 and I_2, Fb is false (the reference of b is not in the extension of F). Therefore, the sentence is true under both interpretations.

5. Laa
   The sentence is a two-place predicate. Thus, in order for it to be true, the ordered pair which is comprised of the reference of a, namely $\langle 1, 1\rangle$ in both interpretations, has to be in the extension of L. That ordered pair is in the extension of L under I_1 but not in that of L under I_2 (because the extension of L is empty in L_2). Therefore, the sentence is true under I_1 whereas it's false under I_2

6. $(Lba \rightarrow Laa)$
   The sentence is a conditional with two two-place predicates. Thus, in order for it to be true, at least one of the following has to be the case: the ordered pair which is comprised of the references of b and a, namely $\langle 2, 1\rangle$m is *not* in the extension of L, or the ordered pair which is comprised of the reference of a, namely $\langle 1, 1\rangle$, is in the extension of L. In both interpretations, $\langle 2, 1\rangle$ is not in the extension of L. Therefore, the sentence is true under both I_1 and I_2.

7. $(Laa \rightarrow Lba)$
   In order for the sentence to be true, at least one of the following has to be the case: the ordered pair $\langle 1, 1\rangle$ is not in the extension of L or the ordered pair $\langle 2, 1\rangle$ is in the extension of L. In I_2, $\langle 1, 1\rangle$ is not in the extension of L. Therefore, the sentence is true under I_2. On the other hand, in I_1, $\langle 1, 1\rangle$ is in the extension of L *and* $\langle 2, 1\rangle$ is not. Therefore, under I_1, the sentence is false.

8. $\exists x Fx$
   The sentence is an existential sentence. Thus, in order for it to be true, there has to be at least one object in the extension of F. In I_1, there is one object in the extension of F, namely 1. Therefore, the sentence is true under I_1. On the other hand, in I_2, there's nothing in the extension of F. Therefore, the sentence is false under I_2.

9. $\exists x{-}Fx$
   The sentence tells us that there has to be at least one object which is not F. Thus, in order for it to be true, there has to be at least one object which is not in the extension of F. In both interpretations, there

are object which are not in the extension of F (in I_1, they are 2 and 3; in I_2, they are 1, 2, and 3). Therefore, the sentence is true under both interpretations.

10. $\exists x(Fx \wedge Gx)$
    In order for the sentence to be true, there has to be at least one object which is in the extensions of F and G. However, in both interpretations, there's no shared object in the extensions of F and G. Therefore, the sentence is false under both interpretations.

11. $(\exists xFx \wedge \exists xGx)$
    In order for the sentence to be true, there has to be at least one object in the extension of F *and* there has to be at least one object in the extension of G. In I_1, there is one object in the extensions of F and G. Therefore, the sentence is true under I_1. On the other hand, in I_2, there is no object in the extension of F. Therefore, the sentence is false under I_2.

12. $\forall xFx$
    In order for the sentence to be true, everything in the domain has to be in the extension of F. In both interpretations, the extension of F fails to contain all members of the domain. Therefore, the sentence is false under both interpretations.

13. $\forall x(Fx \rightarrow Gx)$
    in order for the sentence to be true, each member of the domain, if it's in the extension of F, has to be in the extension of G. In our case, only member of the domain which is in the extension of F is 1. Thus, if the sentence is true, 1 has to be in the extension of G as well. However, since the extension of G is empty, there's no way for 1 to be in the extension of G. Therefore, the sentence is false.

14. $\forall x(Fx \leftrightarrow Gx)$
    In order for the sentence to be true, the extension of F has to be the same as the extension of G. However, they are clearly different; one is comprised of 1, the other is the empty set. Therefore, the sentence is false.

2.3.3.2 Constructing an Interpretation under Which a Given Sentences Are True (or False)

This type of problems requires much more substantial work. In the previous type of problems, all you have to do is to check whether a sentence is true or false under a given interpretation; here, you need to construct your own interpretation under which a given sentence is true (or false).

Let's start with a very simple sentence: Fa. In constructing an interpretation, first you need to set up the domain. Here, for the sake of simplicity, let our domain be $\{1, 2, 3\}$ throughout this section. Once you set up the domain, next you need to specify the extensions of predicates; and as we have seen in the above, if a predicate in question has names, we need to assign objects to them as well. In our case, there's a name in our sentence Fa, namely a; thus, we need to first assign an object of the domain to the name. Let's assign 1 to the name a. Then, Fa is going to be true if 1 is in the extension of F (the instances of such extensions are: $\{1\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}$). On the other hand, if the extension of F doesn't contain 1, Fa is going to be false (the instances of such extensions are: $\emptyset, \{2\}, \{3\}, \{2, 3\}$).

Next, let's think about a slightly more complicated example: $\forall xFx$. This is, needless to say, a universal sentence which says "everything in the domain is F". Thus, every members of the domain need to be in the extension of F; so, the extension should be $\{1, 2, 3\}$. On the other hand, the extension of F doesn't contain all of the domain, $\forall xFx$ is going to be false.

How about $(\forall xFx \rightarrow Lab)$? Although there appears a universal sentence, this is just a conditional. So, the easiest way to make the sentence true would be to make its antecedent false or the consequent true. For the antecedent to be false, the extension of F can be anything except the domain itself. And for the consequent to be true, the extension of L should contain the ordered pair of the objects which are referred to by the names a and b. Let a and b refer to 1 and 2 respectively; thus, the extension of L should be $\{\langle 1, 2\rangle\}$. On the other hand, to make the sentence false, we need to make the antecedent true and the consequent false; thus, the extension of F should be $\{1, 2, 3\}$ and that of L shouldn't contain the objects referred to by a and b.

Note that $(\forall xFx \rightarrow Lab)$ is *not* equivalent to $\forall x(Fx \rightarrow Lab)$. For appreciating the difference between them (or more particularly, for finding an interpretation on which one sentence is true but the other is false), let's construct interpretations for $\forall x(Fx \rightarrow Lab)$. The sentence can be trans-

lated into the following quasi-English sentence: "For each x in the domain, $(Fx \to Lab)$ holds". Thus, it can be transformed as the following conjunction: $((Fa \to Lab) \wedge (Fb \to Lab) \wedge (Fc \to Lab))$ (here, a, b, c refer to $1, 2, 3$ respectively). The easiest way to make all the conjuncts in this conjunction true would be, once again, to make all the antecedents in the conjuncts (namely, Fa, Fb, and Fc) false or to make all the consequents (namely, Lab) true. And we know how to accomplish this; assigning the emptyset ($\emptyset$) to the extension of F or assigning $\{\langle 1, 2\rangle\}$ to the extension of L.

On the other hand, to make $\forall x(Fx \to Lab)$ false, it would suffice to make at least one of the conjuncts in $((Fa \to Lab) \wedge (Fb \to Lab) \wedge (Fc \to Lab))$ false. So, let's make the first conjunct $(Fa \to Lab)$ false. For $(Fa \to Lab)$ to be false, its antecedent Fa has to be true, the consequent Lab false; thus, the extension of F should contain 1 and the extension of L shouldn't contain the ordered pair $\langle 1, 2\rangle$. Thus, the assignments $\{1\}$ for the extension of F and $\emptyset$ for the extension of L make $\forall x(Fx \to Lab)$ false *but* the same assignments make $(\forall x Fx \to Lab)$ true.

There's another approach to finding interpretations which make $\forall x(Fx \to Lab)$ *false*. First, negate the sentence and transform it using the following equivalences.

$$\begin{aligned} (\alpha \to \beta) &\equiv (-\alpha \vee \beta) \\ -(\alpha \wedge \beta) &\equiv (-\alpha \vee -\beta) \\ -(\alpha \vee \beta) &\equiv (-\alpha \wedge -\beta) \\ -\forall &\equiv \exists - \\ -\exists &\equiv \forall - \end{aligned}$$

With these equivalences, we can transform the negated sentence as follows.

$$\begin{aligned} & -\forall x(Fx \to Lab) \\ \equiv\ & \exists x - (Fx \to Lab) \\ \equiv\ & \exists x - (-Fx \vee Lab) \\ \equiv\ & \exists x(- - Fx \wedge -Lab) \\ \equiv\ & \exists x(Fx \wedge -Lab) \end{aligned}$$

(You may feel that this procedure of transformation itself is quite cumbersome; however, with some practices, you'll be able to do this quite easily.)

Since there's no x in $-Lab$, $\exists x(Fx \wedge -Lab)$ is equivalent to $(\exists x Fx \wedge -Lab)$. Thus, $\exists x(Fx \wedge -Lab)$ is going to be true as long as the extension of F contains at least one object from the domain (for example, setting it up as $\{1, 2, 3\}$, namely the domain itself) and the extension of L doesn't contain the ordered pair which is comprised of the objects referred to by a and b (for example, setting it up as $\emptyset$). And then, this makes $(\forall x(Fx \rightarrow Lab))$ false.

To round off this section, let's construct two interpretations one of which makes $\exists(Fx \rightarrow Lab)$ true, the other false.

Just like a universal sentence can be thought of as a conjunction, an existential sentence can be thought of as a disjunction; namely, with the names a, b, c which refer to $1, 2, 3$ respectively, $\exists x(Fx \rightarrow Lab)$ can be written as $((Fa \rightarrow Lab) \vee (Fb \rightarrow Lab) \vee (Fc \rightarrow Lab))$. Thus, for $\exists x(Fx \rightarrow Lab)$ to be true, at least one of the disjuncts $((Fa \rightarrow Lab)$, $(Fa \rightarrow Lab)$, and $(Fa \rightarrow Lab))$ has to be true. To accomplish this, namely, to make one of the disjuncts true, is easy; just make Fa (or Fb or Fc) false, or make Lab true. Thus, setting up the extension of F as empty or the extension of L as $\{\langle 1, 2\rangle\}$ would do.

On the other hand, to find an interpretation which makes the sentence false is not that straightforward. Here, let's negate and transform the sentence first.

$$\begin{aligned} & -\exists x(Fx \rightarrow Lab) \\ \equiv\ & \forall x - (Fx \rightarrow Lab) \\ \equiv\ & \forall x - (-Fx \vee Lab) \\ \equiv\ & \forall x(- - Fx \wedge -Lab) \\ \equiv\ & \forall x(Fx \wedge -Lab) \end{aligned}$$

Since there's no x in $-Lab$, $\forall x(Fx \wedge -Lab)$ is equivalent to $(\forall x Fx \wedge -Lab)$. Thus, $\forall x(Fx \wedge -Lab)$ is going to be true as long as the extension of F contains all the members of the domain (namely, setting it up as $\{1, 2, 3\}$, the domain itself) and the extension of L doesn't contain the ordered pair which is comprised of the objects referred to by a and b (for example, setting it up as $\emptyset$). And then, this makes $(\forall x(Fx \rightarrow Lab))$ false.

Sometimes it's much easier to find an interpretation which makes a sentence false in this way. So, if you're not so sure about what extensions would make a sentence false, it might be a good idea to first negate the sentence in question and try to find an interpretation which makes the negated sentence *true*.

2.4 Truth-Tree Method for L_2

Since the language L_2 is an extension of L_1, the truth-tree method for L_2 is also an extension of the truth-tree method for L_1; we add a few new rules to the ten rules we have studied in 1.9. Those new rules are all concerning the quantifiers. First, let's look at two rules called the *negated quantifier rules*.

Negated Quantifier Rules (NQ)

11. ✓ $-\forall x \ldots$
$\exists x - \ldots$

12. ✓ $-\exists x \ldots$
$\forall x - \ldots$

The meanings (or functions) of these rules should be clear enough; if we have a sentence which starts with $-\forall x$ (resp. $-\exists x$), we can replace $-\forall x$ with $\exists x-$ (resp. $\forall x-$). For example, if you have $-\forall x Fx$ in your tree, you can write down $\exists x - Fx$.

The next two rules are more important and (unfortunately) it would take some time to get used to them. These are called the *instantiation rules for the quantifiers*.

Instantiation Rules for the Quantifiers

13. **Universal Instantiation (UI)**

$\forall x \ldots x \ldots$
$\ldots$ name $\ldots$

In the above, the name with which you replace x has to be one which has already appeared in the path *unless* there's no name in the path. Also note that you must not put the check mark even after you instantiated a universal sentence because you may instantiate the same universal sentence again.

14. **Existential Instantiation (EI)**

✓ $\exists x \ldots x \ldots$
$\ldots$ name $\ldots$

In the above, the name with which you replace x has to be new to the path.

In applying these rules (including ones we've studied in 1.9), you may want to follow the following order of application.

Order of Application of the Rules

1. Rules for negation, conjunction, disjunction, conditional, and bi-conditional. (Rules 2–10; these rules are called the *truth-functional rules*)
2. Rules for negated quantifiers. (Rules 11–12)
3. EI. (Rule 14)
4. UI. (Rule 13)

You may notice that the rule 1, namely the rule for closure, isn't included in the above. This omission means that you should apply the closure rule whenever you can apply it.

The above order is not mandatory but if you follow this order, your tree would be going to be shorter in most cases. (But of course there are exceptions; namely, there are some cases where following the order makes the tree longer and more complicated. We'll see one of such instances in the below.)

2.4.1 Testing the satisfiability of a sentence

Let's test the satisfiability of $(\exists xFx \vee -\exists xFx)$. To test the satisfiability of a sentence, we write a truth tree for the sentence itself. If there is at least one finished open path, the sentence is satisfiable.

1.	✓ $(\exists xFx \vee -\exists xFx)$		
2.	✓ $\exists xFx$	✓ $-\exists xFx$	Rule 5 to Line 1
3.	Fa	$\forall x - Fx$	EI with a new name a from Line 2; NQ to Line 2
4.		$-Fa$	UI with a new name a from Line 3

For the left path of the tree, there's nothing we can do further; the left path will remains open no matter what happens to the right path. For the right path, we can do nothing for closing the path; no matter how many times we instantiate $\forall x - Fx$, there's no hope for closing the path. Thus, both paths are open; the sentence is satisfiable.

How about the satisfiability of $(\forall xFx \wedge \exists x - Fx)$?

1.	✓ $(\forall xFx \wedge \exists x - Fx)$	
2.	$\forall xFx$	Rule 3 to Line 1
3.	✓ $\exists x - Fx$	Rule 3 to Line 1
4.	$-Fa$	EI with a new name a from Line 3
5.	Fa	UI with the name a from Line 2
	$\times$	

The tree is closed; thus, the sentence is not satisfiable (i.e. it's unsatisfiable).

2.4.2 Testing the Validity of a Sentence

Now we know that $(\exists xFx \vee -\exists xFx)$ is satisfiable. Is it valid? To test the validity of a sentence, we need to write a truth tree for the negation of the sentence. If the tree is closed, the sentence is valid.

1.	✓ $-(\exists xFx \vee -\exists xFx)$	
2.	$-\exists xFx$	Rule 6 to Line 1
3.	$--\exists xFx$	Rule 6 to Line 1
	$\times$	

The tree is closed; thus, the sentence is valid.

2.4.3 Testing the Validity of an Argument

Let's recall what motivates the extension of the language L_1. Our motivation of extending L_1 was to capture arguments like the following.

All humans are mortal.
Socrates is a human.

Socrates is mortal.

Now let H and M be predicates which mean "...is a human" and "...is mortal" respectively and let s be a name for Socrates. Then, the above argument can be formalized as follows.

$\forall x(Hx \rightarrow Mx)$
Hs

Ms

Let's see whether this arguments is valid or not by writing a truth tree. To test the validity of an argument, we start with writing down its premise(s) and the negation of its conclusion.

1.	$\forall x(Hx \to Mx)$	
2.	Hs	
3.	$-Ms$	
4.	$\checkmark\ (Hs \to Ms)$	UI with the name s from Line 1
5.	$-Hs \quad Ms$ $\times \quad\ \times$	Rule 7 to Line 4

All the paths are closed; thus, the argument is valid.

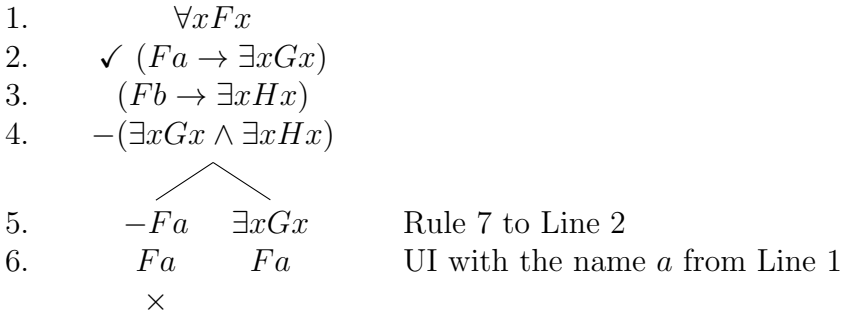
Next let's see whether the following argument is valid or not.

$$\begin{array}{l} \forall xFx \\ (Fa \to \exists xGx) \\ \underline{(Fb \to \exists xHx)} \\ (\exists xGx \wedge \exists xHx) \end{array}$$

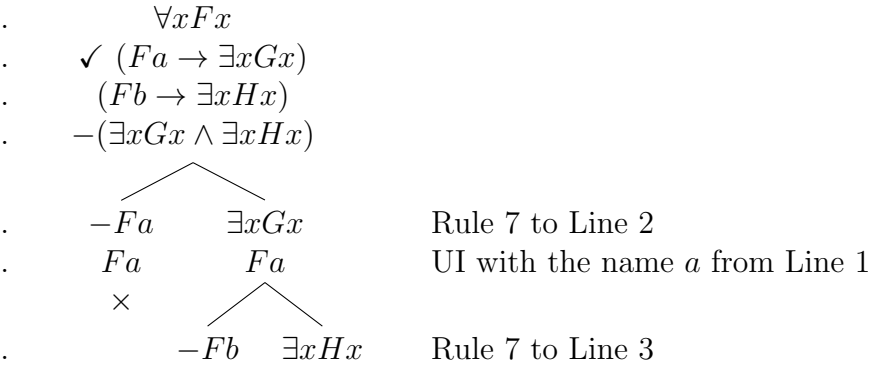
Here's the tree.

1.	$\forall xFx$	
2.	$\checkmark\ (Fa \to \exists xGx)$	
3.	$(Fb \to \exists xHx)$	
4.	$-(\exists xGx \wedge \exists xHx)$	
5.	$-Fa \quad \exists xGx$	Rule 7 to Line 2

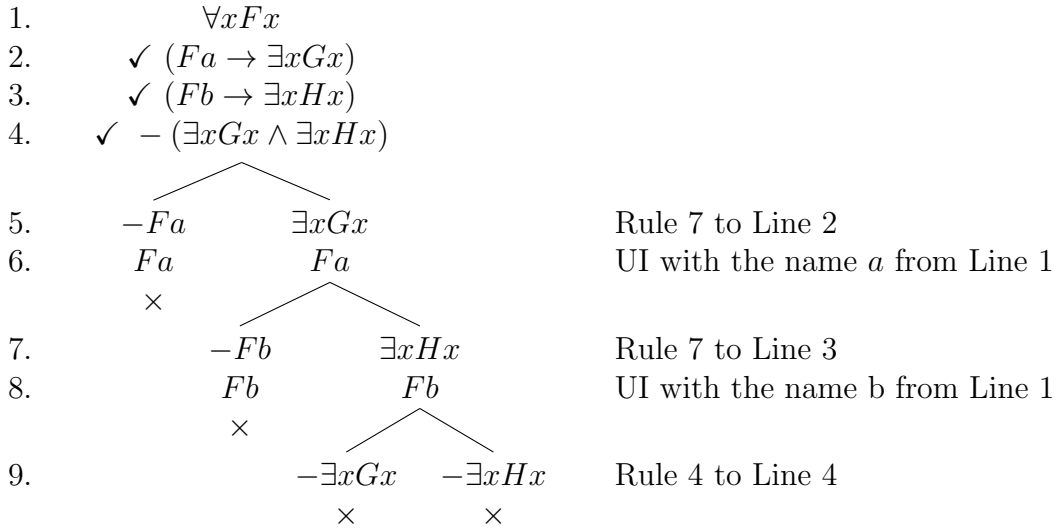
If we follow the recommended order of application, we should decompose either Lines 3 or 4 next. However, we also notice that if we instantiate Line 1 with the name a, the left path is going to be closed. So let's do it.



Next we are back to the recommended order and decompose Line 3.



We can do the same thing for $-Fb$ as we did for $-Fa$ and then decompose Line 4.



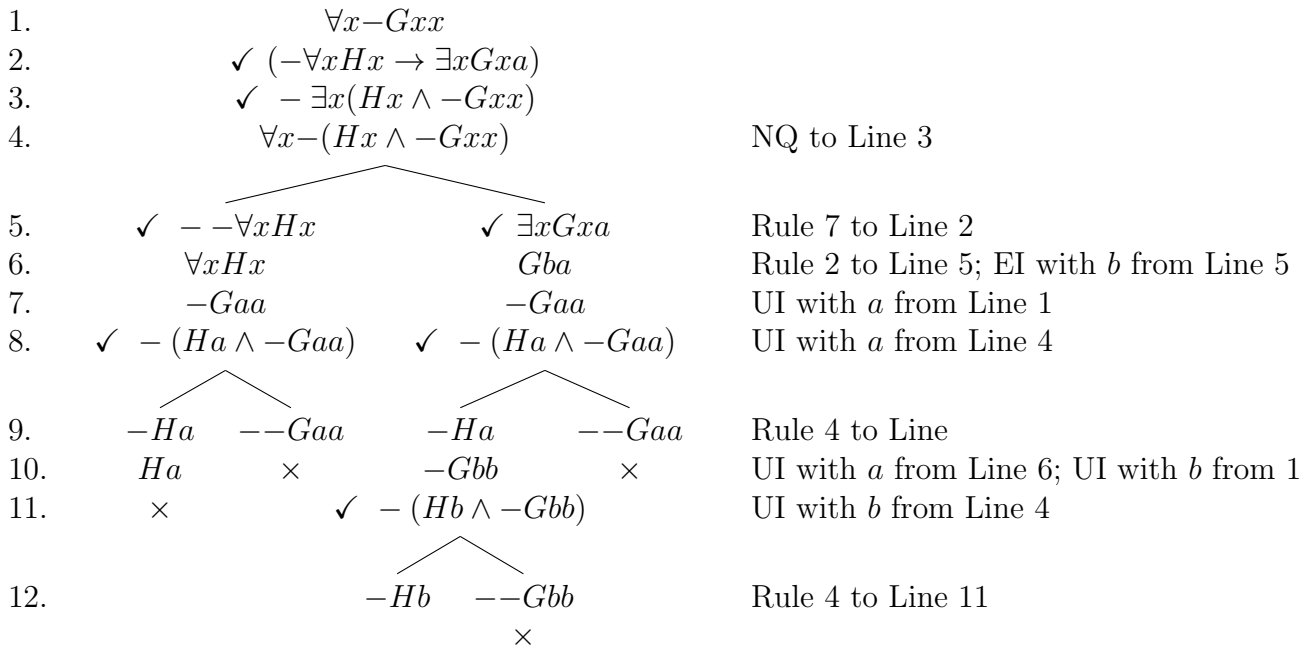
All the paths are closed; thus, the argument is valid. (Note that you don't even have to instantiate $\exists xGx$ and $\exists xHx$; these cause immediate contradiction with the two sentences in Line 9.)

2.4.4 Construct an Interpretation from an Open Path

Let's write a truth tree for the following argument.

$$\frac{\begin{array}{l}\forall x{-}Gxx \\ (-\forall xHx \rightarrow \exists xGxa)\end{array}}{\exists x(Hx \wedge {-}Gxx)}$$

The tree is as follows.



The tree remains open; the argument is invalid.

Now let's construct an interpretation which shows the invalidity of the argument (namely, a counterexample). The procedure is as follows.

1. **Collect atomic sentences which appear as a line by itself in the open path.**
   In the above example, Gba is the only atomic sentence which appears as a line by itself in the open path.

2. **List the names which appear in the atomic sentences collected in the step 1; and assign a numerical object (i.e.** $1, 2, 3$ **etc.) sequentially to each object. A set of those numerical objects will be the domain.**
   In the above, the names which appear in the atomic sentences are a and b; and we assign 1 and 2 to a and b respectively. The domain is, therefore, $\{1, 2\}$.

3. **Set up an extension for each predicate from the atomic sentences collected in the step 1. For predicates which appear**

as a component of some sentences but not as a line by itself, assign $\emptyset$ as their extensions.
In the above, the only atomic sentence which appears as a line by itself in the open path is Gba; and the extension for G is $\{\langle 2,1\rangle\}$. On the other hand, the predicate H doesn't appear as a line by itself; thus, the extension for H is $\emptyset$.

2.4.5 Testing the Equivalence

To test the equivalence between sentences α and β, we test the validity of $(\alpha \leftrightarrow \beta)$; and this amounts to test two implications: $\alpha \models \beta$ and $\beta \models \alpha$ (think why). If both implications hold, α and β are equivalent; if at least one of the implications fails, α and β are not equivalent.

Let's see whether $\forall x(Ax \wedge Bx)$ and $(\forall xAx \wedge \forall xBx)$ are equivalent or not. We need to start our tree with $-(\forall x(Ax \wedge Bx) \leftrightarrow (\forall xAx \wedge \forall xBx))$ of course.

1.	✓ $-(\forall x(Ax \wedge Bx) \leftrightarrow (\forall xAx \wedge \forall xBx))$		
2.	$\forall x(Ax \wedge Bx)$	$-\forall x(Ax \wedge Bx)$	Rule 10 to Line 1
3.	$-(\forall xAx \wedge \forall xBx)$	$(\forall xAx \wedge \forall xBx)$	Rule 10 to Line 1

Let's first finish the left branch.

2.	$\forall x(Ax \wedge Bx)$		Rule 10 to Line 1
3.	✓ $-(\forall xAx \wedge \forall xBx)$		Rule 10 to Line 1
4.	✓ $-\forall xAx$	✓ $-\forall xBx$	Rule 4 to Line 3
5.	✓ $\exists x{-}Ax$	✓ $\exists x{-}Bx$	NQ to Line 4
6.	$-Aa$	$-Ba$	EI with a from Line 5
7.	$(Aa \wedge Ba)$	$(Aa \wedge Ba)$	UI with a from Line 2
8.	Aa	Aa	Rule 3 to Line 7
9.	Ba	Ba	Rule 3 to Line 7
	×	×	

Next, the right branch.

2.	✓ $-\forall x(Ax \wedge Bx)$		Rule 10 to Line 1
3.	✓ $(\forall xAx \wedge \forall xBx)$		Rule 10 to Line 1
4.	$\forall xAx$		Rule 3 to Line 3
5.	$\forall xBx$		Rule 3 to Line 3
6.	✓ $\exists x{-}(Ax \wedge Bx)$		NQ to Line 2
7.	✓ $-(Aa \wedge Ba)$		EI with a from Line 6
8.	$-Aa$	$-Ba$	Rule 4 to Line 8
9.	Aa	Aa	UI with a from Line 4
10.	×	Ba	UI with a from Line 5
		×	

All paths are closed; $\forall x(Ax \wedge Bx)$ and $(\forall xAx \wedge \forall xBx)$ are equivalent.

Next, let's see whether $\exists x(Ax \wedge Bx)$ and $(\exists xAx \wedge \exists xBx)$ are equivalent or not.

1.	$-(\exists x(Ax \wedge Bx) \leftrightarrow (\exists xAx \wedge \exists xBx))$		
2.	$\exists x(Ax \wedge Bx)$	$-\exists x(Ax \wedge Bx)$	Rule 10 to Line 1
3.	$-(\exists xAx \wedge \exists xBx)$	$(\exists xAx \wedge \exists xBx)$	Rule 10 to Line 1

The left branch:

2.	$\exists x(Ax \wedge Bx)$		Rule 10 to Line 1
3.	✓ $-(\exists xAx \wedge \exists xBx)$		Rule 10 to Line 1
4.	✓ $-\exists xAx$	✓ $-\exists xBx$	Rule 4 to Line 3
5.	$\forall x{-}Ax$	$\forall x{-}Bx$	NQ to Line 4
6.	✓ $(Aa \wedge Ba)$	✓ $(Aa \wedge Ba)$	EI with a from Line 2
7.	Aa	Aa	Rule 3 to Line 6
8.	Ba	Ba	Rule 3 to Line 6
	×	×	

The right branch:

2.	$-\exists x(Ax \wedge Bx)$	Rule 10 to Line 1
3.	✓ $(\exists xAx \wedge \exists xBx)$	Rule 10 to Line 1
4.	✓ $\exists xAx$	Rule 3 to Line 3
5.	✓ $\exists xBx$	Rule 3 to Line 3
6.	$\forall x{-}(Ax \wedge Bx)$	NQ to Line 2
7.	Aa	EI with a from Line 4
8.	Bb	EI with b from Line 5
9.	✓ $-(Aa \wedge Ba)$	UI with a from Line 6
10.	✓ $-(Ab \wedge Bb)$	UI with b from Line 6
11.	$-Aa$ × $-Ba$	Rule 4 to Line 9
12.	$-Ab$ $-Bb$ ×	Rule 4 to Line 10

The tree remains open; $\exists x(Ax \wedge Bx)$ and $(\exists xAx \wedge \exists xBx)$ are not equivalent. And from the open path, we can construct an interpretation (a counterexample) in which one of the sentences is true but the other is false as follows (see the previous section for how to construct an interpretation from an open path).

Domain $= \{1, 2\}$
Ext$(a) = 1$
Ext$(b) = 2$
Ext$(A) = \{1\}$
Ext$(B) = \{2\}$

2.4.6 Some Metalogical Considerations

So far we've been assuming that we can trust the results of the tree method but not provided any justification for that assumption; now it's time to provide it.

The justification for the trustworthiness of the tree method can be broken down into the justifications for the following three properties of the tree method for the languages L_1 and L_2.

1. A tree always stops; in other words, there's no tree whose length is infinite. (Decidability)

2. If the initial list of the tree is satisfiable, there's at least one open path in the finished tree. (Soundness)

3. If there's at least one open path in the finished tree, the initial list of the tree is satisfiable. (Completeness)

First, let's verify that a tree always stops.

2.4.6.1 Decidability

Decidability of the tree method is justified by the following properties of the method.

1. Conclusions of the rules are shorter than the premise.

2. Each rule gives a finite number of conclusions.

3. Each line is decomposed (or instantiated) just once.

4. No rule is applied to literals.

In short, by applying the rules, each sentence in the initial list is decomposed into a finite number of literals.

(Note that, strictly speaking, the properties 1 and 3 above don't hold of the language L_2; the conclusion of NQ has the same length as the premise, and a universal sentence can be instantiated more than once. For the first issue, it can be said that, even though the conclusion of NQ isn't shorter than the premise, the conclusion of the conclusion of NQ is always shorter than the original premise; so, this issue doesn't affect the overall claim. For the second issue, first note that there will be only a finite number of names after decomposing all the sentences except universal ones; and those universal sentences are instantiated only with a finite number of names. Therefore, the number of the instantiations of universal sentences is finite as well.)

2.4.6.2 Soundness

This is a direct result of the soundness property of our rules (for the soundness property of the rules, see 1.9.1); if each sentence in the initial list of the tree is true in an interpretation, each conclusion of a rule (in the case of a branching

rule, at least one of its conclusions) has to be true in that interpretation. Consequently, there's at least one path in which all the conclusions are true in an interpretation. In such a path, there can't be a contradiction; thus, such a path remains open.

2.4.6.3 Completeness

This is a direct result of the completeness property of our rules (for the completeness property of the rules, see 1.9.1); if there's at least one open path in the finished tree, there's an interpretation which makes all the conclusions in that path true. And by the completeness property of the rules, the truth of the conclusions guarantees that of each sentence in the initial list.

2.4.6.4 Problem

An interpretation I of a sentence S of L_2 is a model of S, if S is true in I. A model is finite, if its domain is finite. Explain why every satisfiable sentence of L_2 has a finite model.

Chapter 3

Language L_3

The syntactical and semantical aspects (i.e. its vocabulary, the formation rules, and how we interpret its sentences) of L_3 are just the same as those of L_2. Only difference between L_3 and L_2 is that in L_3 there are sentences with *multiple quantifiers with overlapping scope*.

Here, the determiner "with overlapping scope" is important because we've already encountered with sentences with multiple quantifiers *without* overlapping scope. For example, take a look at the following sentence.

$$(\exists x Dax \to \exists x(Dax \wedge Ax))$$

In the above, there are multiple occurrence of the quantifiers. However, the scope of these quantifiers are not overlapping. The first quantifier $\exists x$ governs (or binds) Dax whereas the second governs $(Dax \wedge Ax)$. In each sentence which is governed by the quantifier, there are no other quantifiers. However, in the following sentence, the scope of the quantifiers is overlapping.

$$\forall x \forall y Lxy$$

The first quantifier $\forall x$ governs $\forall y Lxy$ and the second governs Lxy. Here, there's another quantifier in the scope of $\forall x$. In this sense, the scope of $\forall x$ and $\forall y$ is said to be *overlapping*.

First, let's take a look at how we translate (from/into) and interpret L_3-sentences in the following sections.

3.1 Translation from/into L_3-sentences

3.1.1 Translating L_3-Sentences

3.1.1.1 By Way of a Quasi-English Sentence

For a starter, let's take a look at the following sentences with the predicate Lxy which means "x loves y" with the domain = {people} .

1. $\forall x \forall y Lxy$
2. $\forall x \exists y Lxy$
3. $\exists x \forall y Lxy$
4. $\exists x \exists y Lxy$

These four combinations of the quantifiers can be translated into the following quasi (i.e. not-so-natural) English phrases.

$\forall \boldsymbol{x} \forall \boldsymbol{y}$**:** For each x and for each y ....
$\forall \boldsymbol{x} \exists \boldsymbol{y}$**:** For each x there is at least one y such that ....
$\exists \boldsymbol{x} \forall \boldsymbol{y}$**:** There is at least one x such that for each y ....
$\exists \boldsymbol{x} \exists \boldsymbol{y}$**:** There is at least one x and there is at least one y such that ....

With these templates, the above sentences can be translated as

1′. For each person x and for each person y, x loves y.
2′. For each person x there is at least one person y such that x loves y.
3′. There is at least one person x such that for each person y, x loves y.
4′. There is at least one person x and there is at least one person y such that x loves y.

Of course, these sentences can be translated into more natural English sentences as follows.

1″. Everyone loves everyone.
2″. Everyone loves someone.
3″. Someone loves everyone.

4″. Someone loves someone.

If you're dealing with simple sentences like these, it would be easy to translate the L_3-sentences directly into natural English sentences. However, in translating a more complicated L_3-sentence, it would be a good idea to translate it by way of a quasi-English translation; in so doing, you can reduce the chance of making mistakes.

For example, let's translate the following L_3-sentence by way of a quasi-English sentence.

$\forall x \forall y((Ox \wedge Oy) \rightarrow Oxy)$

(Here, the one-place predicate Ox means "x is odd" and the two-place predicate Oxy means "x times y is odd". Even though these predicates are expressed by the same letter O, there's no danger of confusion because you can tell them apart by looking at how many names/variables it takes.)

According to the above templates, this L_3-sentence can be translated as "for each x and for each y, if x is odd and y is odd, then x times y is odd". No matter how unnatural this quasi-English translation seems, you can get a sense of what the L_3-sentence says. Based on this sense, you can translate the quasi-English sentence into a more natural English sentence as "the product of two odd numbers is also odd".

Another example: let's translate $-\exists x \exists y((Px \wedge Py) \wedge Pxy)$ by way of a quasi-English sentence. (Here, the one-place predicate Px means "x is prime" and the two-place predicate Pxy means "x times y is prime".)

> **Quasi-English Translation:** It is not the case that there is at least one x and there is at least one y such that x is prime, y is prime, and x times y is prime.

> **More Natural English Translation:** The product of two prime numbers is never prime.

Yet another example: $\forall x \forall y(Lxy \rightarrow Lyx)$. (Here, Lxy is our now-familiar predicate "x loves y".)

> **Quasi-English Translation:** For each x and for each y, if x loves y, then y loves x.

> **More Natural English Translation:** Love always comes to fruition.

3.1.1.2 Translating an L_3-Sentence Step by Step

Let's translate the following L_3-sentences step by step.

$\forall x(Lxa \to \forall y(Lya \to Lxy))$ (Here, a refers to Alma.)

First, note that in translating the above L_3-sentence, we can utilize one of the four templates which we've studies in Section 2.2.2: All P are $Q \equiv \forall x(Px \to Qx)$. Then, the L_3-sentence can be translated as

All who love Alma has the property $\forall y(Lya \to Lxy)$.

And this property $\forall y(Lya \to Lxy)$ is also the "all P are Q"-type, so it can be translated as

All who love Alma is loved by $x \models x$ loves all who love Alma.

In the above, x is nothing but "all who love Alma". Thus, the final translation is

All who love Alma love all who love Alma.

A more complicated example: $\exists x((Px \wedge Ex) \wedge \forall y((Py \wedge Lyx) \to Oy))$ (Px: x is prime; E: x is even; Lxy: x is larger than y; Oy: x is odd).

First, this L_3-sentence can be translated as

There is at least one even prime number such that $\forall y((Py \wedge Lyx) \to Oy)$.

And $\forall y((Py \wedge Lyx) \to Oy)$ can be translated as

All prime numbers larger than x is odd.

In the above, x is nothing but even prime number. Thus, what the original L_3-sentence tells us is

All prime numbers larger than 2 is odd.

Let's try another type of step-by-step translation. Our target L_3-sentence is this: $\exists x \forall y(Lya \to Lxy)$. This time, we translate two components Lya and Lxy first. These components are, without any difficulty, translated as "y loves Alma" and "x loves y" respectively. Here, x in the latter translation should be translated as "someone" because this x is governed by $\exists$. And since y is governed by $\forall$, the latter translation can be translated as "someone loves everyone". However, that "someone" doesn't love everyone without condition; here, the former translation comes in. This someone loves everyone if that "everyone" loves Alma. Thus, using a proper relative pronoun, we can translate this sentence as "someone loves everyone who loves Alma".

3.1.2 Translating English sentences into L_3-Sentences

3.1.2.1 Trial and Error

Let's translate "someone loves everyone who loves him- or herself" by trial and error.

First, note that we can break this sentence down into two parts: "someone loves everyone" and "everyone loves him- or herself". Also note that "someone" is expressed with $\exists$, "'everyone' with $\forall$. Thus, these sentences can be translated as $\exists x \forall y Lxy$ and $\forall y Lyy$ respectively. (Why use y in $\forall y Lyy$? Well, that's because we used y for expressing "everyone" whom someone (who is expressed by x) loves.)

The next task is to combine what we've got above. There seem the following three possible combinations.

1. $\exists x \forall y(Lxy \wedge Lyy)$.
2. $\exists x \forall y(Lxy \to Lyy)$.
3. $\exists x \forall y(Lyy \to Lxy)$.

Recall that $\exists x \forall y$ can be translated into the following quasi-English phrase: There is at least one x such that for each y .... Let's put the quasi-English translations of Lxy and Lyy to "...". Then, we have

There is at least one x such that for each y, x loves y and y loves y.

On first sight, this (quasi-English) translation seems good. However, this translation claims not only that someone loves all the self-lovers, but also that everyone in this world is a self-lover. This is not what the original sentence says.

How about the second one? Its quasi-English translation is this.

There is at least one x such that for each y, if x loves y, y loves y.

On first sight, this translation also seems good. However, the translation claims that someone loves *only* self-lovers. This is not what the original sentence says either.

Lastly, here comes the third sentence. The following is its quasi-English translation.

There is at least one x such that for each y, if y loves y, x loves y.

According to the above translation, once a person meets the condition that s/he is a self-lover, then someone loves all of such persons. This is exactly what the original sentence says.

Lesson to be learned
In "all/some x love all y who ...", "..." is the condition for y to be loved by x. Thus, the sentence can be paraphrased as "all/some love all y if y meets the condition ..."; and then, this paraphrase is translated as

$$\forall/\exists x(\ldots \to Lxy).$$

Of course, you can apply this scheme for other translations by replacing L with another predicate.

Note that in "all/some x love some y who ...", "..." cannot be taken as a condition for y to be loved by x; "..." is merely a description about y. Thus, the translation is $\forall/\exists x(Lxy \wedge \ldots)$.

3.1.2.2 Utilizing What We Have at Our Disposal

A slightly more complicated example: Everyone who loves everyone who loves Alma loves Alma; let's translate this by utilizing what we have at our disposal.

First, note that the above sentence has the same form as "all P are Q" (here, P is "loving everyone who loves Alma" and Q is "loving Alma"); thus, we can somehow utilize the $\forall x(Px \to Qx)$ (see Section 2.2.2). Moreover, in translating "loves everyone who loves Alma", we can utilize what we studied in the previous section (loving Alma is a condition for "everyone" to be loved). So, the translation is

$$\forall x(\forall y(Lya \to Lxy) \to Lxa).$$

If what we have is "someone who loves everyone who loves Alma loves Alma", we can utilize the "some P are Q" scheme. Thus, the translation is

$$\exists x(\forall y(Lya \to Lxy) \land Lxa).$$

3.1.2.3 Translating Step by Step

Last example: Everyone who loves Alma loves someone who loves Alma. We're gonna translate this step by step.

First, we translate the above as follows. (Here, we apply the "all P are Q" scheme.)

$$\forall x(Lxa \to x \text{ loves someone who loves Alma})$$

Then we can translate it as

$$\forall x(Lxa \to \text{There's someone who loves Alma and } x \text{ loves that person})$$

And finally,

$$\forall x(Lxa \to \exists y(Lya \land Lxy))$$

3.1.2.4 Introducing the Templates

As we've seen in the above, there are several ways to approach translation problems. In what follows, I'll give you some templates for L_3 translation.

First, note that the above sentences have some additional descriptions for those who love or/and for those who are loved. Let's call an additional description for those who love D_x, that for those who are loved D_y. Then, we have the following templates.

1. $\forall x(D_x \to \forall y(D_y \to Lxy))$
2. $\forall x(D_x \to \exists y(D_y \wedge Lxy))$
3. $\exists x(D_x \wedge \forall y(D_y \to Lxy))$
4. $\exists x(D_x \wedge \exists y(D_y \wedge Lxy))$

Rough (quasi) translations of the above are:

1′. All with the additional description D_x love all with the additional description D_y.
2′. All with the additional description D_x love some with the additional description D_y.
3′. Some with the additional description D_x love all with the additional description D_y.
4′. Some with the additional description D_x love some with the additional description D_y.

And the basic structures of these sentences are:

1″. All love all.
2″. All love some.
3″. Some love all.
4″. Some love some.

In using these templates, first figure out which basic structure your sentence has, and use the corresponding template. (If your sentence has the basic structure "Some love all", you need to use the template $\exists x(D_x \wedge \forall y(D_y \to Lxy))$.)

Now, in "all who love Alma love some who don't love Alma", D_x is

Lxa (with the name a referring Alma) and D_y is $-Lya$, and the basic structure of the sentence is "all love some"; thus, the translation of the sentence is:

$$\forall x(Lxa \to \exists y(-Lya \wedge Lxy)).$$

What if our sentence lacks one of additional descriptions? For example, let's think about "all who love Alma love all" and "some love all who don't love Alma". The former lacks an additional description for those who are loved (namely, in our case, for y), the latter for those who love; moreover, the basic structure of the former is "all love all", the latter "some love all". Thus, we get the following:

a. $\forall x(Lxa \to \forall y(D_y \to Lxy))$
b. $\exists x(D_x \wedge \forall y(-Lya \to Lxy))$

Now we have to remove the place-holders for additional descriptions for x and y (namely, D_y in the former, D_x for the latter). But simply removing those wouldn't do; we also have to remove the logical connectives which follow those place-holders D_x and D_y. Thus, we get the following:

a′. $\forall x(Lxa \to \forall y(Lxy))$
b′. $\exists x(\forall y(-Lya \to Lxy))$

They look good but there are unnecessary brackets in $\forall y(Lxy)$ (of the former sentence) and $(\forall y(-Lya \to Lxy))$ (of the latter); so let's add a finishing touch by removing them.

a″. $\forall x(Lxa \to \forall yLxy)$
b″. $\exists x\forall y(-Lya \to Lxy)$

Of course, replacing L with another predicate, we can translate sentences whose main verb is not "love(s)". For example, let's translate "Everyone who loves Alma knows everyone who doesn't love Alma". With the predicate K which means "... know(s) ...", we can translate the sentence as

$$\forall x(Lxa \to \forall y(-Lya \to Kxy))$$

Still, those templates look too complicated? Well, first of all, you

don't have to memorize them; you just have to take a look at this document when you need them. But you may want to memorize them for the exam situations; if that's the case, the following is a small tip.

- First, memorize $Qx(D_x\ C_1\ Qy(Dy\ C_2\ Pxy))$. (Replace P with an appropriate predicate for your sentence.)
- If Qx is $\forall x$, C_1 is $\rightarrow$; if Qx is $\exists x$, C_1 is $\wedge$. (You should be able to memorize this easily because the universal quantifier is usually associated with $\rightarrow$, and the existential usually with $\wedge$.)
- If Qy is $\forall y$, C_2 is $\rightarrow$; if Qy is $\exists y$, C_2 is $\wedge$.

3.1.3 Problems

1. Translate the following L_3-sentences into English sentences using the following keys: Ex: x is even; Exy: x times y is even; Ox: x is odd; Oxy: x times y is odd; Sxy: x plus y is even; Px: x is prime; Lxy: x is larger than y.

1. $\forall x(Ex \rightarrow \forall y Exy)$
2. $\forall x \forall y((Ox \wedge Oy) \rightarrow Oxy)$
3. $\forall x \forall y(Sxy \rightarrow ((Ex \wedge Ey) \vee (Ox \wedge Oy)))$
4. $\forall x((Px \wedge \exists y(Py \wedge Lxy)) \rightarrow Ox)$
5. $-\exists x(Py \wedge \forall y(Px \rightarrow Lyx))$

2. Translate the following sentences.

1. Alma loves everyone who loves her.
2. Everyone who knows Alma loves her.
3. Someone loves everyone who loves him- or herself.
4. Everyone who loves Alma knows everyone who doesn't love Alma.
5. Everyone who loves someone loves someone who loves someone.
6. Everyone who loves everyone who loves Alma knows Alma.

3. Translate the following English sentences into L_3-sentences using the following keys: z: 0; Gxy: $x > y$; Lxy: $y > x$; Exy: $x = y$.

1. For every number there is a greater number.
2. Every number other than 0 is greater than some number.
3. 0 is the one and only number having the property that no number is less than it.

4. Using Lxy in the above question, express $=$ and $\neq$. (Of course, you cannot use Exy here.)

3.2 Interpretation of L_3-Sentences

Basically, you can interpret an L_3-sentence in the way which is described in Section 2.3. And as in the cases for L_2-interpretations, there are two types of problems: deciding a truth value of a sentence under a given interpretation and providing an interpretation which makes a given sentence true or false. Let's see these types of problems one by one.

3.2.1 Deciding a Truth Value of a Sentence under a Given Interpretation

If the sentence you're dealing with is a simple sentence like $\forall x \forall y Lxy$ or $\exists x \forall y Lxy$, there should be no problem in deciding its truth value under a given interpretation; so, let's deal with slightly more complicated examples in what follows.

$\boldsymbol{\forall x \exists y - Lxy}$
Domain $= \{1, 2, 3\}$
$v(L) = \{\langle 1, 1\rangle, \langle 2, 1\rangle, \langle 2, 2\rangle, \langle 3, 3\rangle\}$

Before you start deciding its truth value, you should try to figure out what the sentence means. In this case, the meaning of the sentence is easy to figure out: For each person, there's at least one person s/he doesn't love. According to the above interpretation, the person 1 doesn't love 2 and 3, 2 doesn't love 3, and 3 doesn't love 1 and 2. Thus, in the above interpretation, $\forall x \exists y - Lxy$ is going to be true.

What if we have the following interpretation for $\forall x \exists y - Lxy$?

$v(L) = \{\langle 2, 1\rangle, \langle 2, 2\rangle, \langle 2, 3\rangle, \langle 3, 3\rangle\}$

According to the above interpretation, the person 2 loves everyone in the domain; thus, the sentence is going to be false under this interpretation.

$\exists \boldsymbol{x} \forall \boldsymbol{y} (\boldsymbol{Lxy} \rightarrow \boldsymbol{Lyx})$
Domain $= \{1, 2, 3\}$
$v(L) = \{\langle 1, 1\rangle, \langle 2, 1\rangle, \langle 3, 1\rangle, \langle 3, 2\rangle\}$

The meaning of the above sentence is: There's someone whose love is always reciprocated. The meaning doesn't ring a bell? Then, try the following method.

First, recall that an existential sentence can be thought of as a disjunction. Thus, the sentence can be transformed into

$$(\forall y(L\textcircled{1}y \rightarrow Ly\textcircled{1}) \vee \forall y(L\textcircled{2}y \rightarrow Ly\textcircled{2}) \vee \forall y(L\textcircled{3}y \rightarrow Ly\textcircled{3}))$$

The existence of $\langle 1, 1\rangle$ in the extension of L makes the first disjunct $(\forall y(L\textcircled{1}y \rightarrow Ly\textcircled{1}))$ true; thus, the sentence is going to be true in the above interpretation.

$\exists \boldsymbol{x} \forall \boldsymbol{y} (\boldsymbol{Rxy} \leftrightarrow \boldsymbol{x = y})$
Domain $= \{1, 2\}$
$v(R) = \{\langle 1, 2\rangle\}$

The meaning of the sentence is unclear. (We don't even know what R means!) Still, if you can figure out its truth value without further ado, that's terrific; skip to the next section. But if you can't, first transform it into a disjunction.

$$(\forall y(R\textcircled{1}y \leftrightarrow \textcircled{1} = y) \vee \forall y(R\textcircled{2}y \leftrightarrow \textcircled{2} = y))$$

If you're still not so sure, keep transforming it; turn the universal sentences into conjunctions.

$$\begin{array}{l}(((R\textcircled{1}\textcircled{1} \leftrightarrow \textcircled{1} = \textcircled{1}) \wedge (R\textcircled{1}\textcircled{2} \leftrightarrow \textcircled{1} = \textcircled{2}))\vee \\ \forall y((R\textcircled{2}\textcircled{1} \leftrightarrow \textcircled{2} = \textcircled{1}) \wedge (R\textcircled{2}\textcircled{2} \leftrightarrow \textcircled{2} = \textcircled{2})))\end{array}$$

Then, assign a truth value to each R-sentence; since the only member in the extension of R is $\langle 1, 2\rangle$, all R-sentences except $R\textcircled{1}\textcircled{2}$ are false.

$$(((\underbrace{R\textcircled{1}\textcircled{1}}_{\text{F}} \leftrightarrow \textcircled{1} = \textcircled{1}) \wedge (\underbrace{R\textcircled{1}\textcircled{2}}_{\text{T}} \leftrightarrow \textcircled{1} = \textcircled{2}))\vee$$
$$\forall y((\underbrace{R\textcircled{2}\textcircled{1}}_{\text{F}} \leftrightarrow \textcircled{2} = \textcircled{1}) \wedge (\underbrace{R\textcircled{2}\textcircled{2}}_{\text{F}} \leftrightarrow \textcircled{2} = \textcircled{2})))$$

Next, assign a truth value to each =-sentence.

$$(((\underbrace{R\textcircled{1}\textcircled{1}}_{\text{F}} \leftrightarrow \underbrace{\textcircled{1} = \textcircled{1}}_{\text{T}}) \wedge (\underbrace{R\textcircled{1}\textcircled{2}}_{\text{T}} \leftrightarrow \underbrace{\textcircled{1} = \textcircled{2}}_{\text{F}}))\vee$$
$$\forall y((\underbrace{R\textcircled{2}\textcircled{1}}_{\text{F}} \leftrightarrow \underbrace{\textcircled{2} = \textcircled{1}}_{\text{F}}) \wedge (\underbrace{R\textcircled{2}\textcircled{2}}_{\text{F}} \leftrightarrow \underbrace{\textcircled{2} = \textcircled{2}}_{\text{T}})))$$

Finally, assign a truth value to each biconditional. (If you're uncertain about how to assign a truth value to a biconditional, see 1.2.5.)

$$(\underbrace{(\underbrace{R\textcircled{1}\textcircled{1}}_{\text{F}} \leftrightarrow \underbrace{\textcircled{1} = \textcircled{1}}_{\text{T}})}_{\text{F}} \wedge \underbrace{(\underbrace{R\textcircled{1}\textcircled{2}}_{\text{T}} \leftrightarrow \underbrace{\textcircled{1} = \textcircled{2}}_{\text{F}})}_{\text{F}})\vee$$
$$\forall y(\underbrace{(\underbrace{R\textcircled{2}\textcircled{1}}_{\text{F}} \leftrightarrow \underbrace{\textcircled{2} = \textcircled{1}}_{\text{F}})}_{\text{T}} \wedge \underbrace{(\underbrace{R\textcircled{2}\textcircled{2}}_{\text{F}} \leftrightarrow \underbrace{\textcircled{2} = \textcircled{2}}_{\text{T}})}_{\text{F}}))$$

At this point, the truth value of the above sentence should be clear.

3.2.2 Providing an Interpretation Which Makes a Given Sentence True or False

Let's consider $\forall x \forall y \forall z((Lxy \wedge Lyz) \rightarrow Lxz)$ and provide an interpretation which makes the sentence true.

Suppose we are given $\{1, 2\}$ as the domain; and further suppose we have $\langle 1, 2\rangle$ and $\langle 2, 1\rangle$ in the extension of L; then, when x ranges over 1, y over 2, and z over 1, the antecedent of the conditional $((Lxy \wedge Lyz))$ is going to be true. Thus, in order to make the sentence true, the extension should contain $\langle 1, 1\rangle$ as well for making the consequent (Lxz) true.

Is that all? No; when x ranges over 2, y over 1, and z over 2, the antecedent is going to be true as well. Thus, we also need to include $\langle 2, 2\rangle$ to the extension. So, one of the extensions which make the sentence true is: $\{\langle 1, 2\rangle, \langle 2, 1\rangle, \langle 1, 1\rangle, \langle 2, 2\rangle\}$, all the possible ordered pairs which are comprised of the members of the domain.

How about an interpretation which makes the sentence false? Let's follow the way described in Section 2.3.3.2.

First, let's negate the sentence and transform it.

$$\begin{aligned} -\forall x\forall y\forall z((Lxy \wedge Lyz) \rightarrow Lxz) &\equiv \exists x\exists y\exists z-((Lxy \wedge Lyz) \rightarrow Lxz) \\ &\equiv \exists x\exists y\exists z-(-(Lxy \wedge Lyz) \vee Lxz) \\ &\equiv \exists x\exists y\exists z(--(Lxy \wedge Lyz) \wedge -Lxz) \\ &\equiv \exists x\exists y\exists z((Lxy \wedge Lyz) \wedge -Lxz) \end{aligned}$$

Then, if there's at least one configuration of x, y, z which makes Lxy and Lyz true but Lxz false, we're done. From the above consideration, we know that the ordered pairs $\langle 1, 2\rangle$ and $\langle 2, 1\rangle$ makes both Lxy and Lyz; and if the extension lacks the ordered pair $\langle 1, 1\rangle$, that makes Lxz false. Thus, one of the extensions which make the sentence false is: $\{\langle 1, 2\rangle, \langle 2, 1\rangle\}$

3.2.3 Providing One Interpretation for Two Sentences

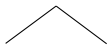
There's yet another type of interpretation questions. In this type of questions, you provide one interpretation for two sentences, and your interpretation should make one sentence true and the other false. For example, let's provide an interpretation for $\exists x\forall yLxy$ and $\forall y\exists xLxy$.

For simple sentences like $\exists x\forall yLxy$ or $\forall y\exists xLxy$, you can easily figure out what these sentences mean, and based on the meanings of sentences, you provide an interpretation which makes one sentence true and the other false. In our example, the meaning of $\exists x\forall yLxy$ is "there's at least one object x which bears the relation L for all objects y" (if we take L as the predicate "... loves ...", the sentence just means that some love all); and the meaning of $\forall y\exists xLxy$ is "for each object y there's at least one object x which bears the relation L to y" (or more concretely, "everyone is loved by someone"). Thus, one of the interpretations which makes one sentence ($\forall y\exists xLxy$) true and the other ($\exists x\forall yLxy$) false is

Domain $= \{1, 2, 3\}$
$v(L) = \{\langle 1, 1\rangle, \langle 2, 2\rangle, \langle 3, 3\rangle\}$.

What if you can't figure out the meaning of sentences easily? For example, the meanings of $\exists x(Lxa \rightarrow Fa)$ and $(\exists xLxa \rightarrow Fa)$ are bit hard

to figure out. Then, write a truth tree for them as follows.

1.	✓ $\exists x(Lxa \to Fa)$	One of the sentences
2.	✓ $-(\exists xLxa \to Fa)$	Negation of the other
3.	✓ $\exists xLxa$	Rule 8 to Line 2
4.	$-Fa$	Rule 8 to Line 2
5.	✓ $(Lba \to Fa)$	EI to Line 1
		
6.	$-Lba$ $\quad$ Fa	Rule 7 to Line 5
7.	Lca $\quad$ $\times$	EI to Line 3

From the above tree, we can construct an interpretation which makes two sentences in the initial list true; and such an interpretation makes $\exists x(Lxa \to Fa)$ true and $(\exists xLxa \to Fa)$ false. This is exactly what we need.

Domain $= \{1, 2, 3\}$
$v(a) = 1$
$v(b) = 2$
$v(c) = 3$
$v(L) = \{\langle 3, 1\rangle\}$
$v(F) = \emptyset$

(Note that if we start with the initial list $\{-\exists x(Lxa \to Fa), (\exists xLxa \to Fa)\}$, the tree is closed. So, if you tried this method and ended up with the closed tree, don't be discouraged; just start over by negating the other sentence.)

3.2.4 Problems

1.

Determine the truth values of the following sentences under the given interpretations.

1. $\exists x \forall y - Lxy$
   Domain $= \{1, 2, 3\}$
   $v(L) = \{\langle 2, 1\rangle, \langle 2, 2\rangle, \langle 3, 2\rangle, \langle 3, 3\rangle\}$

2. $\forall x \exists y(Lxy \to Lyx)$
   Domain $= \{1, 2, 3\}$
   $v(L) = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \langle 2, 1\rangle, \langle 3, 3\rangle\}$

3. $\exists x \forall y(Lxy \to Lyx)$
   Domain $= \{1, 2, 3\}$
   $v(L) = \{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 1\rangle\}$

4. $\exists x \forall y(Rxy \leftrightarrow x = y)$
   Domain $= \{1, 2\}$
   $v(R) = \{\langle 2, 2\rangle\}$

2.

For each of the following sentences specify an interpretation in which it is true and another interpretation in which it is false. You can set up the domain for each case freely.

In constructing an interpretation, try not to use the empty set or the domain itself unless doing so is absolutely necessary.

1. $\forall x \forall y(Lxy \to -Lyx)$
2. $-\exists x \forall y(Lxy \wedge Lyx)$
3. $\forall x(\forall y(Kay \to Kxy) \to Kxa)$ (a is a name)
4. $-\exists x(Lax \wedge \forall y(Kay \to Kxy))$ (a is a name)
5. $\forall x \forall y \forall z((Rxy \wedge Ryz) \to Rxz)$

3.

For each of the following pairs of sentences, construct an interpretation which makes one sentence true but the other false.

1. $\forall x \forall y(Gyx \vee Gxy), \forall x \forall y(Gxx \vee Gxy)$
2. $\exists x(Bx \wedge \forall y Dyx), \forall x(Bx \to \forall y Dyx)$

3.3 Truth Trees for L_3

The rules are just the same as ones described in 2.4. In this section, we're gonna see some examples of truth trees for L_3.

3.3.1 Testing the satisfiability of a sentence

Let's test the satisfiability of $(\exists x \forall x(Dx \to Hxz) \to \forall x(Dx \to \exists z Hxz))$. To test the satisfiability of a sentence, we write a truth tree for the sentence itself. If there is at least one finished open path, the sentence is satisfiable.

1.	✓ $(\exists z \forall x(Dx \to Hxz) \to \forall x(Dx \to \exists z Hxz))$	
2.	✓ $-\exists z \forall x(Dx \to Hxz)$ | $\forall x(Dx \to \exists z Hxz)$	
3.	$\forall z \exists x -(Dx \to Hxz)$	Rules 13 and 14 to Line 2
4.	✓ $\exists x -(Dx \to Hxa)$	Rule 11 to Line 3
5.	✓ $-(Db \to Hba)$	Rules 12 to Line 4
6.	Db	Rule 8 to Line 5
7.	$-Hba$	Rule 8 to Line 5

At this point, we notice that there's nothing we can do for closing the left path. No matter how many times we instantiate $\forall z \exists x -(Dx \to Hxz)$, all we get are the sentences of the forms "$D\ldots$" and "$-H\ldots$"; we'll never find any contradictory sentence with Db or $-Hba$. Thus, the sentence is satisfiable.

And from the above finished open path, we can construct an interpretation which makes the sentence true. There are two names in the left path; this means that our domain should be comprised of two object. Let our domain be $\{1, 2\}$, and let a and b be the names for 1 and 2 respectively. We have Db as a line by itself; thus, the extension of D has to be $\{2\}$. On the other hand, there's no line in which a sentence of the form $H\ldots$ appears as a line by itself; this means that the extension of H is empty. Thus, the following interpretation makes the sentence true.

Domain $= \{1, 2\}$
Extension of a: 1
Extension of b: 2
Extension of D: $\{2\}$
Extension of H: $\emptyset$

3.3.2 Testing the Validity of a Sentence

Now we know that $(\exists x \forall x(Dx \to Hxz) \to \forall x(Dx \to \exists z Hxz))$ is satisfiable. Is it valid? To test the validity of a sentence, we need to write a truth tree for the negation of the sentence. If the tree is closed, the sentence is valid.

1.	✓ $-(\exists z \forall x(Dx \to Hxz) \to \forall x(Dx \to \exists z Hxz))$	
2.	✓ $\exists z \forall x(Dx \to Hxz)$	Rule 8 to Line 1
3.	✓ $-\forall x(Dx \to \exists z Hxz)$	Rule 8 to Line 1
4.	✓ $\exists x -(Dx \to \exists z Hxz)$	Rule 13 to Line 3
5.	✓ $-(Da \to \exists z Haz)$	Rule 12 to Line 4
6.	Da	Rule 8 to Line 5
7.	✓ $-\exists z Haz$	Rule 8 to Line 5
8.	$\forall z - Haz$	Rule 14 to Line 7
9.	$\forall x(Dx \to Hxb)$	Rule 12 to Line 2
10.	✓ $(Da \to Hab)$	Rule 11 to Line 9
11.	$-Da$ $\quad$ Hab	
12.	$\times$ $\quad$ $-Hab$	Rule 11 to Line 8
	$\times$	

All the paths are closed; the sentence is valid.

3.3.3 Testing the Satisfiability, Again

Let's test the satisfiability of $(\forall x \exists y(Fx \to Gy) \leftrightarrow (\exists x Fx \wedge \forall y - Gy))$.

1.	✓ $(\forall x \exists y(Fx \to Gy) \leftrightarrow (\exists x Fx \wedge \forall y{-}Gy))$	
2.	$\forall x \exists y(Fx \to Gy)$ $\quad$ $-\forall x \exists y(Fx \to Gy)$	Rule 9 to Line 1
3.	✓ $(\exists x Fx \wedge \forall y{-}Gy)$ $\quad$ $-(\exists x Fx \wedge \forall y{-}Gy)$	Rule 9 to Line 1
4.	✓ $\exists x Fx$	Rule 3 to Line 3
5.	$\forall y{-}Gy$	Rule 3 to Line 3
6.	Fa	Rule 12 to Line 4
7.	✓ $\exists y(Fa \to Gy)$	Rule 11 to Line 2
8.	✓ $(Fa \to Gb)$	Rule 12 to Line 7
9.	$-Fa$ $\quad$ Gb	Rule 7 to Line 8
10.	$\times$ $\quad$ $-Gb$	Rule 11 to Line 5
	$\times$	

The left path is closed. Now let's move on to the right path.

2.	✓ $-\forall x \exists y(Fx \to Gy)$	Rule 9 to Line 1
3.	✓ $-(\exists x Fx \wedge \forall y{-}Gy)$	Rule 9 to Line 1
4.	✓ $\exists x \forall y{-}(Fx \to Gy)$	Rules 13 and 14 to Line 2
5.	$\forall y{-}(Fa \to Gy)$	Rule 12 to Line 4
6.	✓ $-\exists x Fx$ $\quad$ ✓ $-\forall y{-}Gy$	Rule 8 to Line 3
7.	$\forall x{-}Fx$	Rule 14 to Line 6
8.	✓ $-(Fa \to Ga)$	Rule 11 to Line 5
9.	Fa	Rule 8 to Line 8
10.	$-Ga$	Rule 8 to Line 8
11.	$-Fa$	Rule 11 to Line 7
12.	$\times$ $\quad$ ✓ $\exists y Gy$	Rules 13 and 2 to Line 6
13.	Gb	Rule 12 to Line 12
14.	✓ $-(Fa \to Gb)$	Rule 11 to Line 5)
15.	Fa	Rule 8 to Line 14
16.	$-Gb$	Rule 8 to Line 14
17.	$\times$	

All the paths are closed; therefore, the sentence is unsatisfiable.

Do you want to know when the sentence is going to be false? If so, write a truth tree for the negation of the sentence.

1.	✓ $-(\forall x\exists y(Fx \to Gy) \leftrightarrow (\exists xFx \wedge \forall y{-}Gy))$	
2.	$\forall x\exists y(Fx \to Gy)$ $\quad -\forall x\exists y(Fx \to Gy)$	Rule 10 to Line 1
3.	✓ $-(\exists xFx \wedge \forall y{-}Gy)$ $\quad (\exists xFx \wedge \forall y{-}Gy)$	Rule 10 to Line 1
4.	✓ $-\exists xFx$ $\quad -\forall y{-}Gy$	Rule 4 to Line 3
5.	$\forall x{-}Fx$	Rule 13 to Line 4
6.	✓ $\exists y(Fa \to Gy)$	Rule 11 to Line 2
7.	✓ $(Fa \to Gb)$	Rule 12 to Line 6
8.	$-Fa$ $\quad Gb$	Rule 7 to Line 7
9.	$-Fa$	Rule 11 to Line 5

The left-most path is (kinda) finished and open. (Why "kinda" finished? The reason is as follows. Every time we instantiate the universal sentence $\forall x\exists y(Fx \to Gy)$, we're gonna have a new name; and $\forall x{-}Fx$ doesn't do any good for closing the path with a new name.) At this point, there are two names in the path: a and b. So, our domain is comprised of two members $\{1, 2\}$. (And of course, a refers to 1, b to 2.) Moreover, no predicate appears as a line by itself. Therefore, we can set up the extensions of both predicates as empty. The following interpretation makes the sentence false.

Domain $= \{1, 2\}$
Extension of a: 1
Extension of b: 2
Extension of F: $\emptyset$
Extension of G: $\emptyset$

(The above tree also tells us that $\forall x\exists y(Fx \to Gy)$ and $(\exists xFx \wedge \forall y{-}Gy)$ are not logically equivalent.)

3.3.4 Testing the Validity of an Argument

Now let's test the validity of the following argument.

$$\begin{array}{l}\forall x(\exists y Lxy \rightarrow \forall z Lzx)\\ \exists x \exists y Lxy\\ \hline \forall x \forall y Lxy\end{array}$$

To test the validity of an argument, we write a tree for the premises themselves and the negation of the conclusion.

1.	$\forall x(\exists y Lxy \rightarrow \forall z Lzx)$	
2.	$\checkmark\ \exists x \exists y Lxy$	
3.	$\checkmark\ -\forall x \forall y Lxy$	
4.	$\checkmark\ \exists x -\forall y Lxy$	Rule 13 to Line 3
5.	$\checkmark\ -\forall y Lay$	Rule 12 to Line 4
6.	$\checkmark\ \exists y - Lay$	Rule 13 to Line 5
7.	$-Lab$	Rule 12 to Line 6
8.	$\checkmark\ \exists y Lcy$	Rule 12 to Line 2
9.	Lcd	Rule 12 to Line 8
10.	$\checkmark\ (\exists y Lby \rightarrow \forall z Lzb)$	Rule 11 to Line 1
11.	$\checkmark\ -\exists y Lby$ | $\forall z Lzb$	Rule 7 to Line 10
12.	$\forall y - Lby$ | Lab	Rule 14 to Line 11; Rule 11 to Line 11
13.	$\checkmark\ (\exists y Lcy \rightarrow \forall z Lzc)$ | $\times$	Rule 11 to Line 1
14.	$\checkmark\ -\exists y Lcy$ | $\forall z Lzc$	Rule 7 to Line 13
15.	$\forall y - Lcy$ | $-Lbc$	Rule 14 to Line 15; Rule 11 to 12
16.	$-Lcd$ | Lbc	Rule 11 to Line 16; Rule 11 to Line 14
	$\times$ | $\times$	

The tree is closed; the argument is valid.

3.3.5 Testing the Equivalence between Sentences

Let's figure out whether $\forall x(Fx \rightarrow \exists y Gya)$ and $(\exists x Fx \rightarrow \exists y Gya)$ are logically equivalent.

1.	✓ $-(\forall x(Fx \to \exists yGya) \leftrightarrow (\exists xFx \to \exists yGya))$		
2.	$\forall x(Fx \to \exists yGya)$	$-\forall x(Fx \to \exists yGya)$	Rule 10 to Line 1
3.	✓ $-(\exists xFx \to \exists yGya)$	$(\exists xFx \to \exists yGya)$	Rule 10 to Line 1
4.	✓ $\exists xFx$		Rule 8 to Line 3
5.	✓ $-\exists yGya$		Rule 8 to Line 3
6.	$\forall y{-}Gya$		Rule 14 to Line 5
7.	Fb		Rule 12 to Line 4
8.	✓ $(Fb \to \exists yGya)$		Rule 11 to Line 2
9.	$-Fb$	✓ $\exists yGya$	Rule 7 to Line 8
10.	×	Gca	Rule 12 to Line 9
11.		$-Gca$	Rule 11 to Line 6
		×	

The left side is closed. Let's test the right side.

2.	✓ $-\forall x(Fx \to \exists yGya)$		Rule 10 to Line 1
3.	✓ $(\exists xFx \to \exists yGya)$		Rule 10 to Line 1
4.	✓ $\exists x{-}(Fx \to \exists yGya)$		Rule 13 to Line 2
5.	✓ $-(Fa \to \exists yGya)$		Rule 12 to Line 4
6.	Fa		Rule 8 to Line 5
7.	✓ $-\exists yGya$		Rule 8 to Line 5
8.	$\forall y{-}Gya$		Rule 14 to Line 7
9.	✓ $-\exists xFx$	✓ $\exists yGya$	Rule 7 to Line 3
10.	$\forall x{-}Fx$	Gba	Rule 14 to Line 9; Rule 12 to Line 9
11.	$-Fa$	$-Gba$	Rule 11 to Line 10; Rule 11 to Line 8
	×	×	

The right path is also closed. Therefore, $\forall x(Fx \to \exists yGya)$ and $(\exists xFx \to \exists yGya)$ are logically equivalent.

Chapter 4

Further Extension of L_3

We're gonna add a finishing touch to our subject by introducing two more concepts: *identity* and *function*. This addition makes our language L_3 what is usually called *first-order logic*.

4.1 Identity

Identity is just a predicate; as such, it takes names and gives us back a sentence. And its meaning is what you exactly expect: Something is identical to something. For this sole purpose, we introduce a new symbol $=$. Different from other predicates we've encountered so far, we adopt conventional infix notation; namely, $a = b$ means that a is identical to b.

As a starter, let's take a look at the following simple argument.

$$\frac{\begin{array}{l} a = b \\ b = c \end{array}}{a = c}$$

The argument claims that if a is identical to b and b is identical to c, then a is identical to c; in short, the identity relation is transitive. The truth of it seems pretty obvious but we logicians never trust any claim unless there's a proof of it. So let's prove it. The following are the truth-tree rules for identity.

15. $a \neq a$	16. $a = b$	17. $a = b$
$\times$	$\ldots a \ldots$	$\ldots b \ldots$
	$\ldots b \ldots$	$\ldots a \ldots$

Note that we don't check off the identity sentence $a = b$ in Rules 16 and 17; we can apply Rules 16 and 17 as many times as we want.

In the rule 15, $a \neq a$ is just another notation for $-a = a$. Then, the rule tells us that we can close the path where a sentence of the form $a \neq a$ appears. The rules 16 and 17 tells us that, if we have an identity sentence $a = b$ in the path, we can replace a with b (resp. b with a) in a sentence. For example, if we have $a = b$ and Lab in the same path, we can get Laa by replacing b with a in Lab. And from Laa, we can get (or recover) Lab (yes, you don't have to replace both occurrences of a with b).

Now we're ready for writing a truth tree for the above argument. Since we want to test the validity of the argument, we need to start with the premises and the negation of the conclusion of course.

1.	$a = b$	
2.	$b = c$	
3.	$a \neq c$	
4.	$a = c$	Rule 17 to Lines 1 and 2
	$\times$	

Thus, as we expected, the argument is valid.

Can the above argument be generalized for any members of the domain? This generalized claim is expressed in one sentence: $\forall x \forall y \forall z((x = y \wedge y = z) \rightarrow x = z)$. If this sentence is proven to be valid, it would follow that the identity relation is actually transitive. In order to test the validity of this sentence, we start our tree with the negation of the sentence.

1.	✓ $-\forall x\forall y\forall z((x = y \wedge y = z) \rightarrow x = z)$	
2.	✓ $\exists x\exists y\exists z-((x = y \wedge y = z) \rightarrow x = z)$	3 NQs to Line 1
3.	✓ $-((a = b \wedge b = c) \rightarrow a = c)$	3 EIs to Line 2
4.	✓ $(a = b \wedge b = c)$	Rule 8 to Line 3
5.	$a \neq c$	Rule 8 to Line 3
6.	$a = b$	Rule 3 to Line 4
7.	$b = c$	Rule 3 to Line 4
8.	$a = c$	Rule 17 to Lines 6 and 7
	×	

The tree is closed; thus, transitivity does actually hold for the identity relation between any three members.

Now let's figure out whether the negated version of the above claim holds. The sentence the validity of which we're gonna test is this: $\forall x\forall y\forall z((x \neq y \wedge y \neq z) \rightarrow x \neq z)$.

1.	✓ $-\forall x\forall y\forall z((x \neq y \wedge y \neq z) \rightarrow x \neq z)$	
2.	✓ $\exists x\exists y\exists z-((x \neq y \wedge y \neq z) \rightarrow x \neq z)$	3 NQs to Line 1
3.	✓ $-((a \neq b \wedge b \neq c) \rightarrow a \neq c)$	3 EIs to Line 2
4.	$(a \neq b \wedge b \neq c)$	Rule 8 to Line 3
5.	✓ $-a \neq c$	Rule 8 to Line 3
6.	$a = c$	Rule 2 to Line 5
7.	$a \neq b$	Rule 3 to Line 4
8.	$b \neq c$	Rule 3 to Line 4
9.	$c \neq b$	Rule 16 to Lines 6 and 7
10.	$b \neq a$	Rule 17 to Lines 6 and 8

At this point, there's nothing we can do any further; we did all we could. Thus, the sentence is not valid. Let's construct an interpretation on which the sentence is going to be false.

First note that there are three names a, b, and c, so let's set up our domain as $\{1, 2, 3\}$ and the extensions of a, b, c as $1, 2, 3$ respectively. Now the only atomic sentence which appears as a line by itself is $a = c$. This sentence tells us that the extensions of a and c should refer to the same object in the domain; we have to change the extension of either a or c to capture this fact. So let's change the extension of c to 1. What about the extension of $=$? Well, since that is obvious enough, we usually omit the extension of

=. (If you're a perfectionist, of course you can write down the extension; in our case, it should be $\{\langle 1,1\rangle, \langle 2,2\rangle, \langle 3,3\rangle\}$.) Thus, one of the interpretations which makes the sentence false is:

Domain : $\{1, 2, 3\}$
Extension of a : 1
Extension of b : 2
Extension of c : 1.

In the above interpretation, although 1 is not identical to 2 and 2 is not identical to 1, 1 is identical to 1.

Let's do two more examples in which predicates appear. First, consider the following argument.

$\forall x \forall y(Fxy \vee Fyx)$
$a = b$
$\overline{\forall x(Fxa \vee Fbx)}$

For the validity checking, our tree of course starts with the premises themselves and the negation of the conclusion.

1.	$\forall x \forall y(Fxy \vee Fyx)$	
2.	$a = b$	
3.	✓ $-\forall x(Fxa \vee Fbx)$	
4.	✓ $\exists x{-}(Fxa \vee Fbx)$	NQ to Line 3
5.	✓ $-(Fca \vee Fbc)$	EI to Line 4
6.	$-Fca$	Rule 6 to Line 5
7.	$-Fbc$	Rule 6 to Line 5
8.	$\forall y(Fby \vee Fyb)$	UI to Line 1
9.	✓ $(Fbc \vee Fcb)$	UI to Line 9
10.	Fbc $\quad$ Fcb	Rule 5 to Line 9
11.	× $\quad$ Fca	Rule 17 to Lines 2 and 10
	×	

The tree is closed; the argument is valid.

Next, the following argument.

$\exists x F x a$
$\forall x(x = a \rightarrow x = b)$
$\overline{\exists x F x x}$

1.		✓ $\exists x F x a$	
2.		$\forall x(x = a \rightarrow x = b)$	
3.		✓ $-\exists x F x x$	
4.		$\forall x - F x x$	NQ to Line 3
5.		$F c a$	EI to Line 1
6.		✓ $(a = a \rightarrow a = b)$	UI to Line 2
7.		✓ $(b = a \rightarrow b = b)$	UI to Line 2
8.		✓ $(c = a \rightarrow c = b)$	UI to Line 2
9.	$a \neq a$ ×	$a = b$	Rule 7 to Line 6
10.		$b \neq a$	$b = b$ Rule 7 to Line 7
11.			$c \neq a$ $c = b$ Rule 7 to Line 8
12.			$-Fcc$ $-Fcc$ UI to Line 4
13.			$-Fbb$ $c = a$ Rule 17 to Lines 9 and 11
14.			$-Faa$ $-Fca$ Rule 16 to Lines 12 and 13
15.		$b \neq b$ ×	× Rule 16 to Lines 9 and 10

The tree is finished (we instantiated all universal sentences with all names); still, the second path from the left remains open (in order to bring about a contradiction, we need $-Fca$ but it's impossible because a line of the path clearly states that c is not equal to a (Line 11)).

We have two names in the path but two of them (a and b) are identical; thus, we can set up the domain and the extensions of the names as

Domain $= \{1, 2\}$
$v(a) = 1$
$v(b) = 1$

$$v(c) = 2$$

For the extension of F, there's only one line in which an F-sentence appears by itself (Line 5); thus, the extension of F is

$$v(F) = \{2, 1\}$$

4.1.1 Problems

Determine whether each of the following sentences is valid, unsatisfiable, or neither of them by writing a truth tree. If the sentence is valid or unsatisfiable, construct an interpretation from your open path which makes it true or false. If it's neither of them, provide two interpretation one of which makes it true and the other false.

1. $(\forall x\ x = a \rightarrow (\exists x Fx \rightarrow \forall x Fx))$
2. $\forall x \forall y\ x \neq y$
3. $\exists x \exists y\ x \neq y$
4. $\forall x \forall y((Fx \leftrightarrow Fy) \rightarrow x = y)$
5. $((\exists x Gax \rightarrow -\exists x Gxa) \rightarrow \forall x(Gxa \rightarrow x \neq a))$

4.2 Numerical Expressions

One of the benefits of introducing the identity predicate is that we can express numerical expressions like "at least", "at most", or "exactly" with $=$. Let's see how in the below.

4.2.1 At Least

At least one

As we know (and actually you *should* know), we can express the "at least one" type of expressions without the help of the identity predicate. For example, "there's at least one P" can be expressed as follows.

$$\exists x Px.$$

We can also express the same sentence with the help of $=$.

$$\exists x \forall y (x = y \rightarrow Py).$$

It might be a bit hard to understand why the above is equivalent to $\exists x Px$. Of course, you can convince yourself by writing a truth tree for their equivalence. But here, let's try to understand why conceptually.

First, let's translate the above L_3-sentence into a quasi-English sentence as follows. (If you're not so sure about how to translate an L_3-sentence into a quasi-English sentence, see Section 3.1.1.1)

There's at least one x such that for each y, if x is identical to y, y is P.

Here, note that a universal sentence can be thought of as a conjunction; if a refers to the object referred to by x, and if $a_1, \ldots, a_n$ refer to all the objects of thedomain, the above sentence can be thought of as $((a = a_1 \rightarrow Pa_1) \wedge \ldots \wedge (a = a_n \rightarrow Pa_n))$. Therefore, *at least* one of $a = a_1, \ldots, a = a_n$ has to be true in order for the sentence to be true. And actually, there's always *at least* one object to which the object a refers to is identical: the object a refers to.

At least two

Then, how about the "at least two" expressions? The first thought might be that it's enough to concatenate two existential sentences, namely, $(\exists x Px \wedge \exists y Py)$ for "there are at least two Ps". However, $(\exists x Px \wedge \exists y Py)$ is going to be true even when the extension of P contains just one member. We need something more.

Although it's not quite right, the basic idea above is on the right track; we need two existential sentences to express the "at least two" sentences. What we need further is to make sure that the references of these existential sentences don't refer to the same object. Thus, the right way to express "there are at least two Ps" is

$$\exists x \exists y (Px \wedge Py \wedge x \neq y).$$

At least three

We can express the "at least three" sentences in a very similar manner as

above; we need three existential sentences and make sure that the references of these existential sentences don't refer to the same thing. Thus, "there are at least three Ps" can be expressed as

$$\exists x \exists y \exists z (Px \wedge Py \wedge Pz \wedge x \neq y \wedge y \neq z \wedge z \neq x).$$

4.2.2 At Most

At most one
The phrase "there is at most one ..." can be paraphrased as "it's not the case that there are at least two ...". And we know that how to express "at least two" (and of course we also know how to express "it's not the case that ..."). Thus, "there is at most one P" can be expressed as

$$-\exists x \exists y (Px \wedge Py \wedge x \neq y).$$

The above sentence can be transformed into

$$\forall x \forall y ((Px \wedge Py) \rightarrow x = y).$$ (Verify this.)

The literal translation of the above sentence is "for any x and y, if x and y are both P, then x is identical to y". (In this form, it might be clearer than the above negated existential sentence that there doesn't have to be any P in order for "there is at most one P" to be true.)

The above sentence can be abbreviated further as

$$\exists x \forall y (Py \rightarrow y = x).$$

At most two
In a similar fashion as above, the phrase "there are at most two ..." can be paraphrased as "it's not the case that there are at least three ...". Thus, "there are at most two Ps" can be expressed as

$$-\exists x \exists y \exists z (Px \wedge Py \wedge Pz \wedge x \neq y \wedge y \neq z \wedge z \neq x).$$

The above sentence can be transformed into the following universal sentence.

$\forall x \forall y \forall z((Px \wedge Py \wedge Pz) \rightarrow (x = y \vee y = z \vee z = x))$. (Verify this.)

Again, in order for the sentence to be true, there doesn't have to be any P.

At most three
Let's express a "there are at most three ..." sentence directly in the universal sentential form.

From the above examples, we know that in order to express the "there are at most n ..." sentences we need $n + 1$ universal quantifiers and have to make sure that at least one pair of variables (instantiations of variables, to be exact) refers to the same object. Thus, "there are at most three Ps" can be expressed as

$\forall w \forall x \forall y \forall z((Pw \wedge Px \wedge Py \wedge Pz) \rightarrow (w = x \vee w = y \vee w = z \vee x = y \vee x = z \vee y = z))$.

4.2.3 Exactly

The "exactly n" type of sentences can be expressed with the combination of the "at least n" and "at most n" types of expressions. Once you understand how to express the "at least" and "at most" sentences, it should be easy to express the "exactly" sentences.

Exactly one
The "exactly one" sentences are expressed with the combination of the "at least one" and "at most one" sentences. Thus, "there's exactly one P" can be expressed as

$(\exists x Px \wedge -\exists x \exists y(Px \wedge Py \wedge x \neq y))$.

We know that the second conjunct can be transformed into $\forall x \forall y((Px \wedge Py) \rightarrow x = y)$; thus, the sentence can be rewritten as

$(\exists x Px \wedge \forall x \forall y((Px \wedge Py) \rightarrow x = y))$.

Furthermore, we also know that the second conjunct can be abbreviated as $\exists x \forall y(Py \rightarrow y = x)$; thus, the sentence can be abbreviated as

$(\exists x Px \wedge \exists x \forall y(Py \rightarrow y = x))$. (This can be abbreviated as $\exists x(Px \wedge \forall y(Py \rightarrow y = x))$.)

Recall that $\exists x Px$ is equivalent to $\exists x \forall y(y = x \rightarrow Py)$. Thus, we can rewritten the above as

$$(\exists x \forall y(y = x \rightarrow Py) \wedge \exists x \forall y(Py \rightarrow y = x)).$$

In the above, we can factor out $\exists x \forall y$ in front of the sentence. Thus, we get

$$\exists x \forall y((y = x \rightarrow Py) \wedge (Py \rightarrow y = x)).$$

Note that $(y = x \rightarrow Py) \wedge (Py \rightarrow y = x)$ is nothing but $(Py \leftrightarrow y = x)$. Thus, "there's exactly one P" can be expressed as

$$\exists x \forall y(Py \leftrightarrow y = x).$$

Exactly two
In a very similar manner as in the above, we can formalize the expression "exactly two" as follows.

$$\begin{aligned} & \exists x \exists y(Px \wedge Py \wedge x \neq y \wedge \forall z(Pz \rightarrow (z = x \vee z = y))) \\ \equiv \; & \exists x \exists y(x \neq y \wedge \forall z(Pz \leftrightarrow (z = x \vee z = y))) \end{aligned}$$

4.2.3.1 Problem

By writing truth trees, show that the following sentences are all equivalent.

1. $\exists x \forall y(Fy \leftrightarrow y = x)$
2. $\exists x(Fx \wedge \forall y(Fy \rightarrow y = x))$
3. $(\exists x Fx \wedge \forall y \forall z((Fy \wedge Fz) \rightarrow y = z))$

4.2.4 Definite Description

When it's used with a singular noun, the definite article "the" refers to exactly one object. Thus, a sentence where the word "the" appears can be expressed with the "exactly one" expressions explained above. For example, "the even prime'is a number" can be expressed as

$$\exists x(\forall y((Ey \wedge Py) \leftrightarrow y = x) \wedge Nx).$$

Expressing "The successor of any number is greater than that number" in L_3 is a bit trickier. First, let's translate it as the following quasi L_3-sentence.

$$\forall x(Nx \rightarrow \text{the successor of } x \text{ is greater than } x).$$

And then, all we have to do is to translate "the successor of x is greater than x".

$$\forall x(Nx \rightarrow \exists y \forall z((Szx \leftrightarrow z = y) \wedge Gyx)).$$

We can summarize how to express a definite description as the following rule.

Russell's Rule

$$\ldots \text{the P} \ldots \Rightarrow \exists x(\forall y(Py \leftrightarrow y = x) \wedge \ldots x \ldots)$$

4.2.5 Summary

To recap:

There is at least one P: $\exists x Px$; $\exists x \forall y(y = x \rightarrow Py)$.

There are at least two Ps: $\exists x \exists y(Px \wedge Py \wedge x \neq y)$.

There are at least three Ps: $\exists x \exists y \exists z(Px \wedge Py \wedge Pz \wedge x \neq y \wedge y \neq z \wedge z \neq x)$.

There is at most one P: $\forall x \forall y((Px \wedge Py) \rightarrow x = y)$.

There are at most two Ps: $\forall x \forall y \forall z((Px \wedge Py \wedge Pz) \rightarrow (x = y \vee y = z \vee z = x))$.

There are at most three Ps: $\forall w \forall x \forall y \forall x((Pw \wedge Px \wedge Py \wedge Pz) \rightarrow (w = x \vee w = y \vee w = z \vee x = y \vee x = z \vee y = z))$.

There is exactly one P: $\exists x(Px \wedge \forall y(Py \rightarrow y = x))$; or $\exists x \forall y(Py \leftrightarrow y = x)$

There are exactly two Ps: $\exists x \exists y(Px \wedge Py \wedge x \neq y \wedge \forall z(Pz \rightarrow (z = x \vee z = y)))$; or $\exists x \exists y(x \neq y \wedge \forall z(Pz \leftrightarrow (z = x \vee z = y)))$

4.2.6 Examples

Basically, for translating sentences with numerical expressions, all you have to do is to take care of translating P in the above summary. However, in some cases, you need some strategies. Let's study some of them by translating some sentences.

Alma loves at least two persons.
Translation: $\exists x \exists y(Lax \wedge Lay \wedge x \neq y)$

There should be no problem in translating this; just put Lax and Lay in the places of Px and Py.

At most one person loves everyone.
Translation: $\forall x \forall y(\forall z(Lxz \wedge Lyz) \rightarrow x = y)$

First, you need to replace P with some expression for "loving everyone". We know that "...loves everyone" is translated as $\forall y L \ldots y$ and those who love are x and y in our sentence; so, in the places of Px and Py, we put $\forall z Lxz$ and $\forall z Lyz$ respectively (you need to use a variable other than x or y for $\forall$ here), and then factor out $\forall z$ in front of Lxz and Lyz (of course, in this factoring-out, we need to introduce the brackets).

There are at most two persons who know someone who doesn't love Alma.
Translation: $\forall x \forall y \forall z(\exists w(-Lwa \wedge Kxw \wedge Kyw \wedge Kyz) \rightarrow (x = y \vee y = z \vee z = x))$

This is an "at most" sentence; so we know that we need three variables for denoting persons in question. Let these variables be x, y, z. And for these x, y, z, it is said that they know someone who doesn't love Alma. Thus, utilizing one of the templates, the situation can be expressed as $\exists w(-Lwa \wedge Lxw \wedge Lyw \wedge Lzw)$; and this is your P.

Everyone knows exactly one person who loves him/her.
Translation: $\forall x \exists y((Kxy \wedge Lyx) \wedge \forall z((Kxz \wedge Kzx) \rightarrow z = y))$; or $\forall x \exists y \forall z((Kxz \wedge Lyz) \leftrightarrow z = y)$

You cannot put $\forall x$ right before $(Kxy \wedge Lyx)$; if you did so, the meaning of the translation would be "there's exactly one person who love everyone and everyone knows that person". The original sentence doesn't mean this; rather, it means that each person knows his or her only lover. Thus, in order to express this "each" aspect, we need to put $\forall x$ before $\exists y$.

Exactly two persons know someone who knows everyone who loves Alma.

Translation: $\exists x \exists y(\exists z(\forall v(Lva \rightarrow Kzv) \wedge Kxz \wedge Kyz) \wedge x \neq y \wedge \forall w(\exists z(\forall v(Lva \rightarrow Kzv) \wedge Kwz) \rightarrow (w = x \vee w = y)))$; or $\exists x \exists y(x \neq y \wedge \forall z(\exists v(\forall w(Lwa \rightarrow Kvw) \wedge Kzv) \leftrightarrow (z = x \vee z = y)))$

Essentially, if you can translate the "someone who knows everyone who loves Alma" part, the rest is just a tedious taking-care of the variables. If we forget about the variable-conflict for now, the "someone who" part is translated as $\exists x \forall y(Lya \rightarrow Kxy)$; and exactly two persons know this guy x. Thus, the above translation.

You think these are too tough? Even if so, don't be discouraged. What you'll come across in the real life (I mean, in the exams or assignments) are much easier than these.

4.2.7 Problems

Translate the following.

1. At least two persons love Alma.
2. Alma loves at most one person.
3. There are at most two persons who don't love Alma.
4. Exactly one person knows exactly two persons who love Alma.

4.3 Functions

4.3.1 Basics

In our formal logic, a function receives a name (or names) as its input(s) and returns *one and only one* name as its output. We call its input(s) an *argument* (or *arguments*) and an output the *value*.

There's also another condition which a function has to meet: For each object of the domain, there has to be the value. Thus, the following extension of a function f is *not* legitimate.

Domain : $\{1, 2\}$
Extension of f: $\{\langle 1, 1\rangle, \langle 1, 2\rangle\}$

The reason why the above is not legitimate is:

1. There are two values for 1 (violating the condition 1).
2. There's no value for 2 (violating the condition 2).

An example of proper extensions for f would be $\{\langle 1, 1\rangle, \langle 2, 2\rangle\}$.

> 1. For any input(s), a function has to return exactly one value.
> 2. For each object of the domain, a function has to return a value.
>
> In one sentence: For each object of the domain, a function has to return exactly one value.

We can think of an n-place function; and in order to explicitly express we're taking about an n-place function, we sometimes add a superscript like $f^n(t_1, \ldots, t_n)$ and write the extension for such a function as $\{\langle\langle t_1, \ldots, t_n\rangle, value\rangle\}$. For example, if $g^2(x, y)$ is a function expressing the sum of x and y, its extension would be $\{\langle\langle 1, 1\rangle, 2\rangle, \langle\langle 1, 2\rangle, 3\rangle, \ldots\}$.

4.3.2 Translating with Functions

With functions, we can translate "exactly one" phrases much more easily. For example, consider the following sentences.

1. Alma has exactly one father.
2. The father of Alma is older than Alma.
3. The father of any person is older than that person.

Translations of these sentences with predicates are as follows. (Here, we use the predicates Px (x is a person), Fxy (x is a father of y), Oxy (x is older than y), and the name a for Alma.)

$1'$. $\exists x \forall y(Fya \leftrightarrow y = x)$
$2'$. $\exists x(\forall y(Fya \leftrightarrow y = x) \wedge Oxa)$
$3'$. $\forall x(Px \rightarrow \exists y(\forall z(Fzx \leftrightarrow z = y) \wedge Oyx))$

With a function, we can translate these sentences more succinctly. (Here, we use the function $f(x)$ for "the father of x".)

$1''$. $\exists x x = f(a)$
$2''$. $Of(a)a$
$3''$. $\forall x(Px \to Of(x)x)$

4.3.3 Truth-Tree Method for Functions

By introducing the concept of a function, we need to modify the rules for the quantifiers a bit, and for that modification, we need to introduce the new term *terms*.

Terms

1. Names are terms. Names are also called *constants*.
2. Variables are terms.
3. If $t_1, \ldots, t_n$ are terms, so is $f(t_1, \ldots, t_n)$.

If a term doesn't contain any variable, it's called *closed*; if it does, it's called *open*. For example, a, b, $f(a)$, $g(a, b)$ are all closed terms whereas x, y, $f(x)$, $g(x, b)$, and $g(x, y)$ are all open.

Now, the modified rules for the quantifiers are

Modified EI
If you have $\exists x \ldots x \ldots$, you can replace x in it with a new constant.

Modified UI
If you have $\forall x \ldots x \ldots$, you can replace x in it with any combination of names and function symbols which are already in the path where the universal sentence you're going to instantiate appears.

The modification to EI is made in order to make clear that we cannot replace a variable in an existential sentence with a functional form like $f(a)$. Thus, the rule isn't essentially changed.

On the other hand, the modification to UI gives us more freedom in instantiation. For example, let's suppose we have $f(a)$ and $g(b, c)$ in the

path where the universal sentence we're gonna instantiate appears; then, we can instantiate the universal sentence not only with $f(a)$ or $g(b,c)$ but also with any combination of a, b, c, f, and g like $f(c)$, $g(a,b)$, or $f(g(c,f(b)))$. (However, you seldom want to instantiate a universal sentence with such a free combination; in most cases, simple forms like $f(a)$ or $g(b,c)$ would do.)

As an example, let's write a truth tree for testing the validity of $\forall x \exists y(y = f(x) \wedge \forall z(z = f(x) \rightarrow z = y))$.

1.	✓ $-\forall x \exists y(y = f(x) \wedge \forall z(z = f(x) \rightarrow z = y))$	
2.	✓ $\exists x \forall y{-}(y = f(x) \wedge \forall z(z = f(x) \rightarrow z = y))$	2 NQs to Line 1
3.	$\forall y{-}(y = f(a) \wedge \forall z(z = f(a) \rightarrow z = y))$	EI to Line 2
4.	$-(a = f(a) \wedge \forall z(z = f(a) \rightarrow z = a))$	UI to Line 3

In instantiating $\forall y{-}(y = f(a) \wedge \forall z(z = f(a) \rightarrow z = y))$ for the first time, we use the simple name a.

1.	✓ $-\forall x \exists y(y = f(x) \wedge \forall z(z = f(x) \rightarrow z = y))$		
2.	✓ $\exists x \forall y{-}(y = f(x) \wedge \forall z(z = f(x) \rightarrow z = y))$		2 NQs to Line 1
3.	$\forall y{-}(y = f(a) \wedge \forall z(z = f(a) \rightarrow z = y))$		EI to Line 2
4.	✓ $-(a = f(a) \wedge \forall z(z = f(a) \rightarrow z = a))$		UI to Line 3
5.	$a \neq f(a)$	✓ $-\forall z(z = f(a) \rightarrow z = a)$	Rule 4 to Line 4
6.		✓ $\exists z{-}(z = f(a) \rightarrow z = a)$	NQ to Line 5
7.		✓ $-(b = f(a) \rightarrow b = a)$	EI to Line 5
8.		$b = f(a)$	Rule 8 to Line 7
9.		$b \neq a$	Rule 8 to Line 7

At this point, we did all we could do except instantiating Line 3 with other terms. So far, we have two names a and b, and one function f; thus, we can instantiate $\forall y{-}(y = f(a) \wedge \forall z(z = f(a) \rightarrow z = y))$ with $f(a)$ or $f(b)$. Let's instantiate it with $f(a)$. (You write down the following derivation under both paths.)

10.	✓ $-(f(a) = f(a) \wedge \forall z(z = f(a) \to z = f(a)))$	UI to Line 3
11.	$f(a) \neq f(a)$ ✓ $-\forall z(z = f(a) \to z = f(a))$	Rule 4 to Line 10
12.	× ✓ $\exists z{-}(z = f(a) \to z = f(a))$	NQ to Line 11
13.	✓ $-(b = f(a) \to b = f(a))$	EI to Line 12
14.	$b = f(a)$	Rule 8 to Line 13
15.	$b \neq f(a)$	Rule 8 to Line 13
	×	

All the paths are closed; thus, the sentence is valid. (By the way, this sentence is nothing but the condition a function has to meet.)

By the way, according to some textbooks (for example, see Bergmann et al., *The Logic Book*, McGraw-Hill), it's totally okay to instantiate the line 3 with $f(a)$ in the line 4. Thus, the truth tree can be much shorter as follows.

1.	✓ $-\forall x \exists y(y = f(x) \wedge \forall z(z = f(x) \to z = y))$	
2.	✓ $\exists x \forall y{-}(y = f(x) \wedge \forall z(z = f(x) \to z = y))$	2 NQs to Line 1
3.	$\forall y{-}(y = f(a) \wedge \forall z(z = f(a) \to z = y))$	EI to Line 2
4.	✓ $-(f(a) = f(a) \wedge \forall z(z = f(a) \to z = f(a)))$	UI to Line 3
5.	$f(a) \neq f(a)$ ✓ $-\forall z(z = f(a) \to z = f(a))$	Rule 4 to Line 4
6.	× ✓ $\exists x{-}(z = f(a) \to z = f(a))$	NQ to Line 5
7.	✓ $-(b = f(a) \to b = f(a))$	EI to Line 6
8.	$b = f(a)$	Rule 8 to Line 7
9.	$b \neq f(a)$	Rule 8 to Line 7
	×	

The last example: Suppose we're given the following pair of sentences: $-\forall x\ x = f(x)$ and $a \neq f(a)$. Now let's find one interpretation which makes one of the sentences true but the other false. In finding such an interpretation, you can write a truth tree whose initial list is comprised of one of the sentences and the negation of the other. (See **??**.)

1.	✓	$-\forall x\ x = f(x)$	
2.	✓	$-\ \ a \neq f(a)$	
3.		$a = f(a)$	Rule 2 to Line 2
4.	✓	$\exists x\ x \neq f(x)$	NQ to Line 1
5.		$b \neq f(b)$	EI to Line 4

All compound sentences are decomposed; the tree is finished and has an open path. Let's construct an interpretation from the open path. (See 2.4.4.)

First, collect all the names which appear in your tree and assign a number to each name sequentially as follows.

Ext(a): 1
Ext(b): 2

These numbers which you just assigned to the names will be the objects of our domain.

Now, we have to set up the extension for the function f. First, recall that your function has to be *total*; meaning, to each name there has to be a value of f. Thus, the extension has to look like this.

Ext(f): $\{\langle 1, -\rangle, \langle 2, -\rangle\}$

And according to the sentences in the tree, the value for 1 is 1 itself, and the value for 2 is not 2. Since our domain has only two objects, there is only one option for the value for 2: 1. Thus, the extension of f is:

Ext(f): $\{\langle 1, 1\rangle, \langle 2, 1\rangle\}$

Taken the above together, the interpretation which makes one sentence true and the other false is:

Domain: $\{1, 2\}$
Ext(a): 1
Ext(b): 2
Ext(f): $\{\langle 1, 1\rangle, \langle 2, 1\rangle\}$.

Solutions

Solutions for Problems 1.1.4

1. $\rightarrow$
2. $\wedge$
3. $\vee$
4. $\rightarrow$
5. $-$

Solutions for Problems 1.2.7

1.

A	B	$-A$	S_1 $(--A \wedge B)$	S_2 $(-A \leftrightarrow B)$	$(S_1 \rightarrow S_2)$
T	T	F	T	F	F
T	F	F	F	T	T
F	T	T	F	T	T
F	F	T	F	F	T

2.

D	E	H	$-D$	$-H$	S_1 $(D \wedge E)$	S_2 $(-H \vee S_1)$	$(-D \wedge S_2)$
T	T	T	F	F	T	T	F
T	T	F	F	T	T	T	F
T	F	T	F	F	F	F	F
T	F	F	F	T	F	T	F
F	T	T	T	F	F	F	F
F	T	F	T	T	F	T	T
F	E	T	T	F	F	F	F
F	E	F	T	T	F	T	T

3.

A	B	D	$-A$	$-B$	$-D$	S_1 $(-A \wedge B)$	S_2 $(-D \vee -B)$	S_3 $(D \leftrightarrow S_1)$	S_4 $-S_3$	$(S_4 \vee S_2)$
T	T	T	F	F	F	F	F	F	T	T
T	T	F	F	F	T	F	T	T	F	T
T	F	T	F	T	F	F	T	F	T	T
T	F	F	F	T	T	F	T	T	F	T
F	T	T	T	F	F	T	F	T	F	F
F	T	F	T	F	T	T	T	F	T	T
F	F	T	T	T	F	F	T	F	T	T
F	F	F	T	T	T	F	T	T	F	T

4.

E	F	J	$-E$	$-F$	$-J$	S_1 $(E \vee F)$	S_2 $(-E \wedge -F)$	S_3 $(S_1 \wedge S_2)$	S_4 $(J \wedge S_3)$	$(S_4 \rightarrow -J)$
T	T	T	F	F	F	T	F	F	F	T
T	T	F	F	F	T	T	F	F	F	T
T	F	T	F	T	F	T	F	F	F	T
T	F	F	F	T	T	T	F	F	F	T
F	T	T	T	F	F	T	F	F	F	T
F	T	F	T	F	T	T	F	F	F	T
F	F	T	T	T	F	F	T	F	F	T
F	F	F	T	T	T	F	T	F	F	T

5.

A	B	C	$-A$	S_1 $(B \wedge C)$	S_2 $(A \leftrightarrow S_1)$	S_3 $-S_2$	S_4 $(B \leftrightarrow S_3)$	S_5 $-S_4$	S_6 $(-A \leftrightarrow S_5)$	$-S_6$
T	T	T	F	T	T	F	F	T	F	T
T	T	F	F	F	F	T	T	F	T	F
T	F	T	F	F	F	T	F	T	F	T
T	F	F	F	F	F	T	F	T	F	T
F	T	T	T	T	F	T	T	F	F	T
F	T	F	T	F	T	F	F	T	T	F
F	F	T	T	F	T	F	T	F	F	T
F	F	F	T	F	T	F	T	F	F	T

Solutions for Problems 1.3.6

1. Valid/Satisfiable.
2. Valid/Satisfiable.

3. Invalid/Unsatisfiable.
4. Valid/Satisfiable.
5. Valid/Satisfiable.

Solutions for Problems 1.4.1.1

1.

1. In order for this implication to fail, α on the right-hand side has to be false. However, if α on the right-hand side is false, so is α on the left-hand side. There's no way to make α on the left-hand side true but α on the right-hand side false. Therefore, the implication holds.

2. Suppose that $\Gamma \models \alpha$ holds but $\Gamma, \beta \models \alpha$ fails. In order to make this happen, α has to be false. Then, Γ has to be false as well; otherwise, $\Gamma \models \alpha$ is going to fail. Now, if Γ is false, so is $\{\Gamma, \beta\}$ regardless of the truth value of β. Then, $\Gamma, \beta \models \alpha$ is going to be valid. However, this is contradictory to our assumption. Therefore, there's no way to make $\Gamma \models \alpha$ hold but $\Gamma, \beta \models \alpha$ fail. This proves the statement.

3. Suppose that $\Gamma, \Delta \models \beta$ fails. This means that Γ, Δ are true but β is false. Now, we're supposing the truth of $\Gamma \models \alpha$ and $\alpha, \Delta \models \beta$. In order for the latter to be the case, either α or Δ has to be true. However, we're assuming the truth of Δ. Therefore, α has to be false. However, if α is false, $\Gamma \models \alpha$ is going to fail. This contradicts the assumption that $\Gamma \models \alpha$ holds. Therefore, if $\Gamma \models \alpha$ and $\alpha, \Delta \models \beta$ hold, $\Gamma, \Delta \models \beta$ must hold as well.

4. Suppose that $\Gamma \models (\alpha \to \beta)$ fails. Then, Γ and α have to be true, and β has to be false. If so, $\Gamma, \alpha \models \beta$ fails as well. This contradicts the assumption. On the other hand, from the supposition that $\Gamma, \alpha \models \beta$ fails, we can draw the same conclusion. Therefore, the statement holds.

5. Suppose that $\alpha \models \gamma$ fails. If so, α has to be true and γ has to be false. Now, we're assuming the truth of $\alpha \models \beta$ and $\beta \models \gamma$. If γ is false, β has to be true. However, if so, $\alpha \models \beta$ is going to fail. This contradicts the assumption. Therefore, the statement holds.

2.

1. Implication holds.
2. Implication fails. (Counterexample: A=T, B=F, C=T)
3. Implication holds.
4. Implication holds.

Solutions for Problems 1.4.2.1

1. These are all easy. Try to derive a contradiction from the assumption that the implication doesn't hold.

2.

1. Correct.
2. Incorrect.
3. Incorrect.
4. Correct.
5. Incorrect.

Solutions for Problems 1.5.1

1.

1. Suppose that the argument is not valid; thus, $(\alpha \vee \beta)$ and $-\alpha$ are true, but β is false. Since $(\alpha \vee \beta)$ is true and β is false, α has to be true. However, if so, $-\alpha$ is going to be false. Contradiction. The argument is valid.

2. Suppose that the argument is not valid; thus, $(\alpha \rightarrow \beta)$ and $-\beta$ are true, but $-\alpha$ is false. Since $\alpha \rightarrow \beta$ is true and $-\alpha$ is false, β has to be true. However, if so, β is going to be false. Contradiction. The argument is valid.

3. Suppose that the argument is not valid; thus, $(\alpha \rightarrow \gamma), (\beta \rightarrow \gamma), (\alpha \vee \beta)$ are all true, but γ is false. Since $(\alpha \rightarrow \gamma), (\beta \rightarrow \beta)$ are both true and γ is false, both α and β have to be false. However, if so, $(\alpha \vee \beta)$ is going

to be false. Contradiction. The argument is valid.

2.

1. Invalid. (Counterexample: A=T, B=T, C=T/F)
2. Valid.
3. Invalid. (Counterexample: B=T, C=F, D=F)
4. Valid.
5. Invalid. (Counterexample: A=F, B=F, C=F)

Solutions for Problems 1.7.2.1

1. $((A \wedge -B \wedge C) \vee (A \wedge -B \wedge -C) \vee (-A \wedge B \wedge C) \vee (-A \wedge -B \wedge C))$
2. $((A \wedge B \wedge C) \vee (A \wedge -B \wedge C) \vee (A \wedge -B \wedge -C) \vee (-A \wedge B \wedge C))$
3. $((A \wedge B \wedge C) \vee (A \wedge -B \wedge C) \vee (A \wedge -B \wedge -C))$
4. $((A \wedge B \wedge -C) \vee (A \wedge -B \wedge C) \vee (A \wedge -B \wedge -C) \vee (-A \wedge B \wedge C) \vee (-A \wedge B \wedge -C) \vee (-A \wedge -B \wedge C) \vee (-A \wedge -B \wedge -C))$
5. $((A \wedge B \wedge -C) \vee (A \wedge -B \wedge -C) \vee (-A \wedge B \wedge -C) \vee (-A \wedge -B \wedge C))$

Solutions for Problems 1.8.1

1.

1. $(\alpha \vee \beta) : ((\alpha \mid \alpha) | (\beta \mid \beta))$
2. $(\alpha \rightarrow \beta) : (\alpha \mid (\beta \mid \beta))$
3. $(\alpha \downarrow \beta) : (((\alpha \mid \alpha) \mid (\beta \mid \beta)) \mid ((\alpha \mid \alpha) \mid (\beta \mid \beta)))$

2.

1. $-\alpha : (\alpha \downarrow \alpha)$
2. $(\alpha \wedge \beta) : ((\alpha \downarrow \alpha) \downarrow (\beta \downarrow \beta))$
3. $(\alpha \vee \beta) : ((\alpha \downarrow \beta) \downarrow (\alpha \downarrow \beta))$
4. $(\alpha \rightarrow \beta) : (((\alpha \downarrow \alpha) \downarrow \beta) \downarrow ((\alpha \downarrow \alpha) \downarrow \beta))$
5. $((\alpha \mid \beta) : (((\alpha \downarrow \alpha) \downarrow (\beta \downarrow \beta)) \downarrow ((\alpha \downarrow \alpha) \downarrow (\beta \downarrow \beta)))$

3.

1. $(\alpha \vee \beta) : -(-\alpha \wedge -\beta)$

2. $(\alpha \to \beta) : -(\alpha \wedge -\beta)$
3. $(\alpha \mid \beta) : -(\alpha \wedge \beta)$
4. $(\alpha \downarrow \beta) : (-\alpha \wedge -\beta)$

4.

1. $(\alpha \wedge \beta) : -(-\alpha \vee -\beta)$
2. $(\alpha \to \beta) : (-\alpha \vee \beta)$
3. $(\alpha \mid \beta) : (-\alpha \vee -\beta)$
4. $(\alpha \downarrow \beta) : -(\alpha \vee \beta)$

5.

1. $(\alpha \wedge \beta) : -(\alpha \to -\beta)$
2. $(\alpha \vee \beta) : (-\alpha \to \beta)$
3. $(\alpha \mid \beta) : (\alpha \to -\beta)$
4. $(\alpha \downarrow \beta) : -(-\alpha \to \beta)$

Solutions for Problems 1.9.3.1

1.

$$\checkmark\ (J \to (K \to J))$$

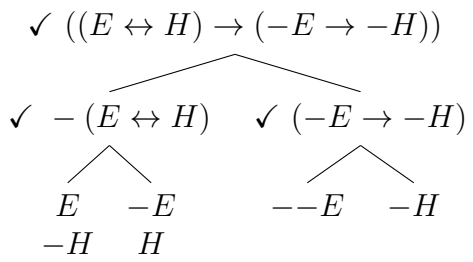
$$-J \qquad (K \to J)$$

This tree is clearly open (you don't have to decompose the right-side path because the left-side path is open no matter what happens to the right-side path. See Rule of Thumb 3); thus, the sentence is satisfiable. However, we still need to decide whether it is valid or not.

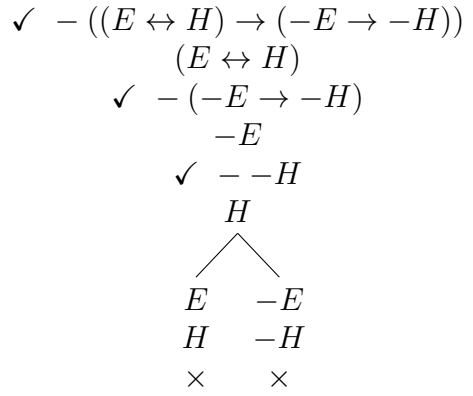
$$\checkmark\ -(J \to (K \to J))$$
$$J$$
$$\checkmark\ -(K \to J)$$
$$K$$
$$-J$$
$$\times$$

This one is closed; thus, the sentence is valid.

2.

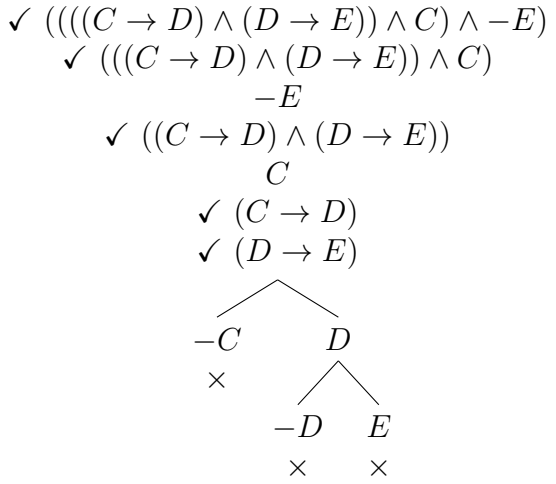


The tree is open; thus, the sentence is satisfiable.
Now we gonna test its validity.



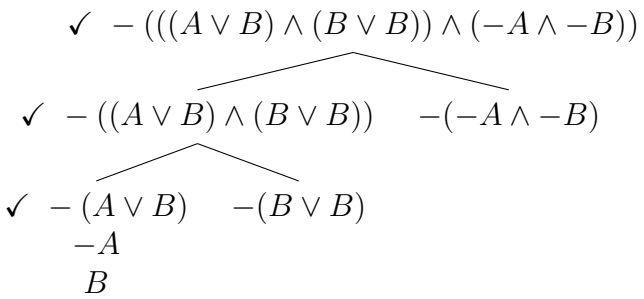
The tree is closed; the sentence is valid.

3.



The tree is closed; the sentence is unsatisfiable (and also invalid).

4.



The left-most path remains open no matter what happens to other paths; thus, the sentence is satisfiable.

Now for the test for its validity.

$$
\begin{array}{c}
--(((A \vee B) \wedge (B \vee B)) \wedge (-A \wedge -B)) \\
\checkmark\ (((A \vee B) \wedge (B \vee B)) \wedge (-A \wedge -B)) \\
\checkmark\ ((A \vee B) \wedge (B \vee B)) \\
\checkmark\ (-A \wedge -B) \\
\checkmark\ (A \vee B) \\
(B \vee B) \\
-A \\
-B \\
\diagup\ \diagdown \\
A \quad B \\
\times \quad \times
\end{array}
$$

The tree is closed; the sentence is valid.

5.

$$
\begin{array}{c}
\checkmark\ (-B \to ((B \vee D) \to D)) \\
\diagup\ \diagdown \\
-B \quad ((B \vee D) \to D)
\end{array}
$$

The tree is open (there's nothing to do with the left-side path); thus, the sentence is satisfiable.

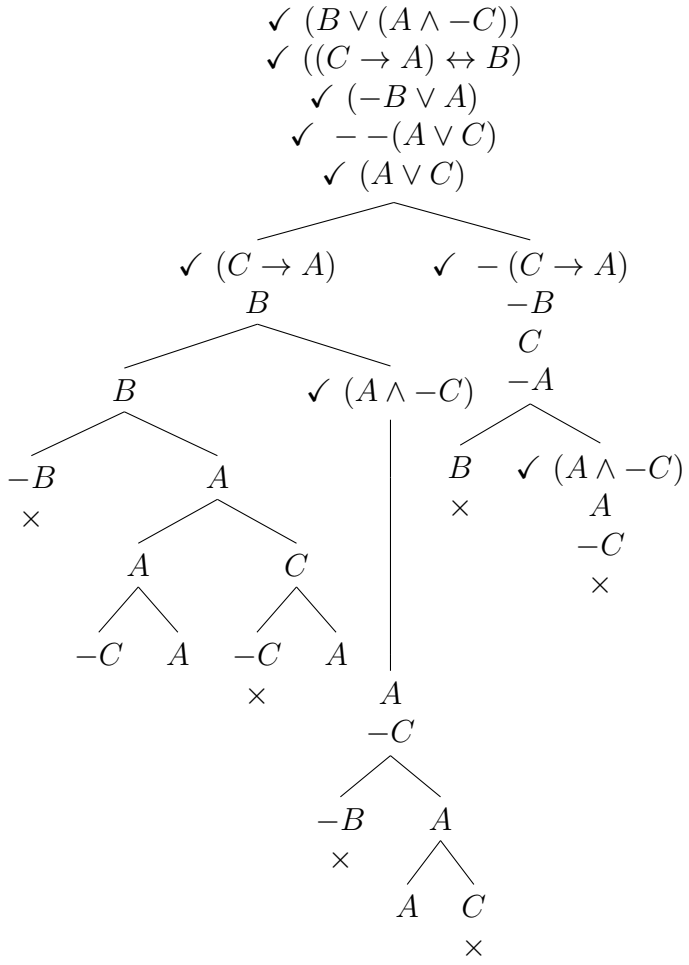
We're gonna test its validity.

$$
\begin{array}{c}
\checkmark\ -(-B \to ((B \vee D) \to D)) \\
-B \\
\checkmark\ -((B \vee D) \to D) \\
\checkmark\ (B \vee D) \\
-D \\
\diagup\ \diagdown \\
B \quad D \\
\times \quad \times
\end{array}
$$

The tree is closed; the sentence is valid.

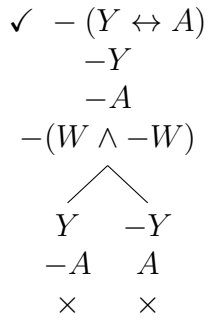
Solutions for Problems 1.9.4.1

1.



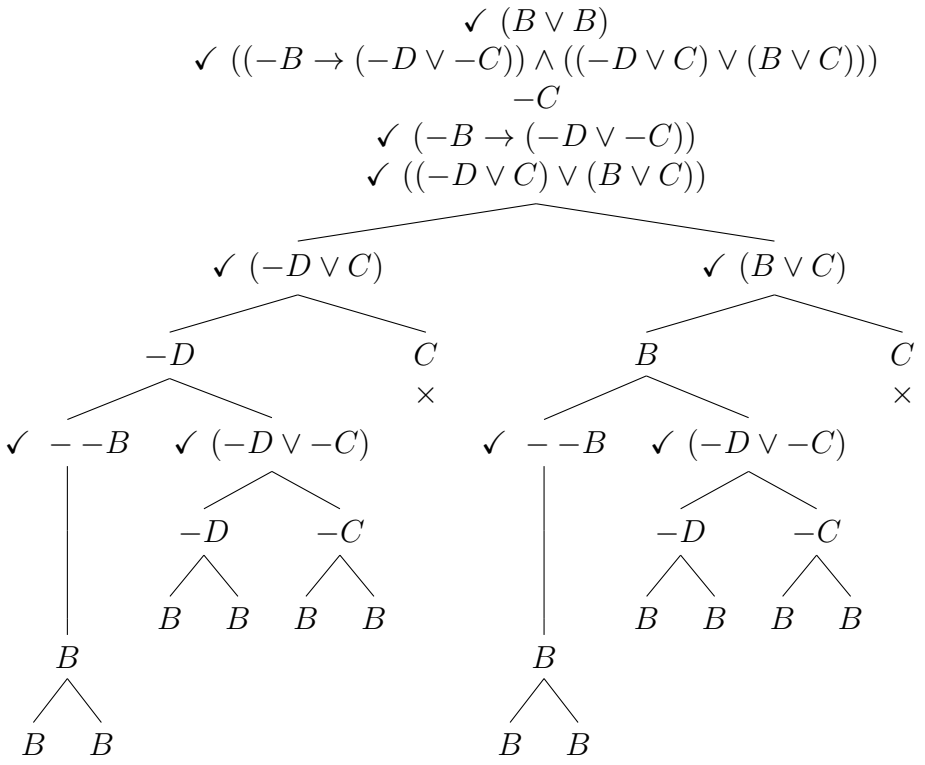
The tree is open; the argument is invalid. A counterexamples are given when A and B are both true, and C is either true or false.

2.



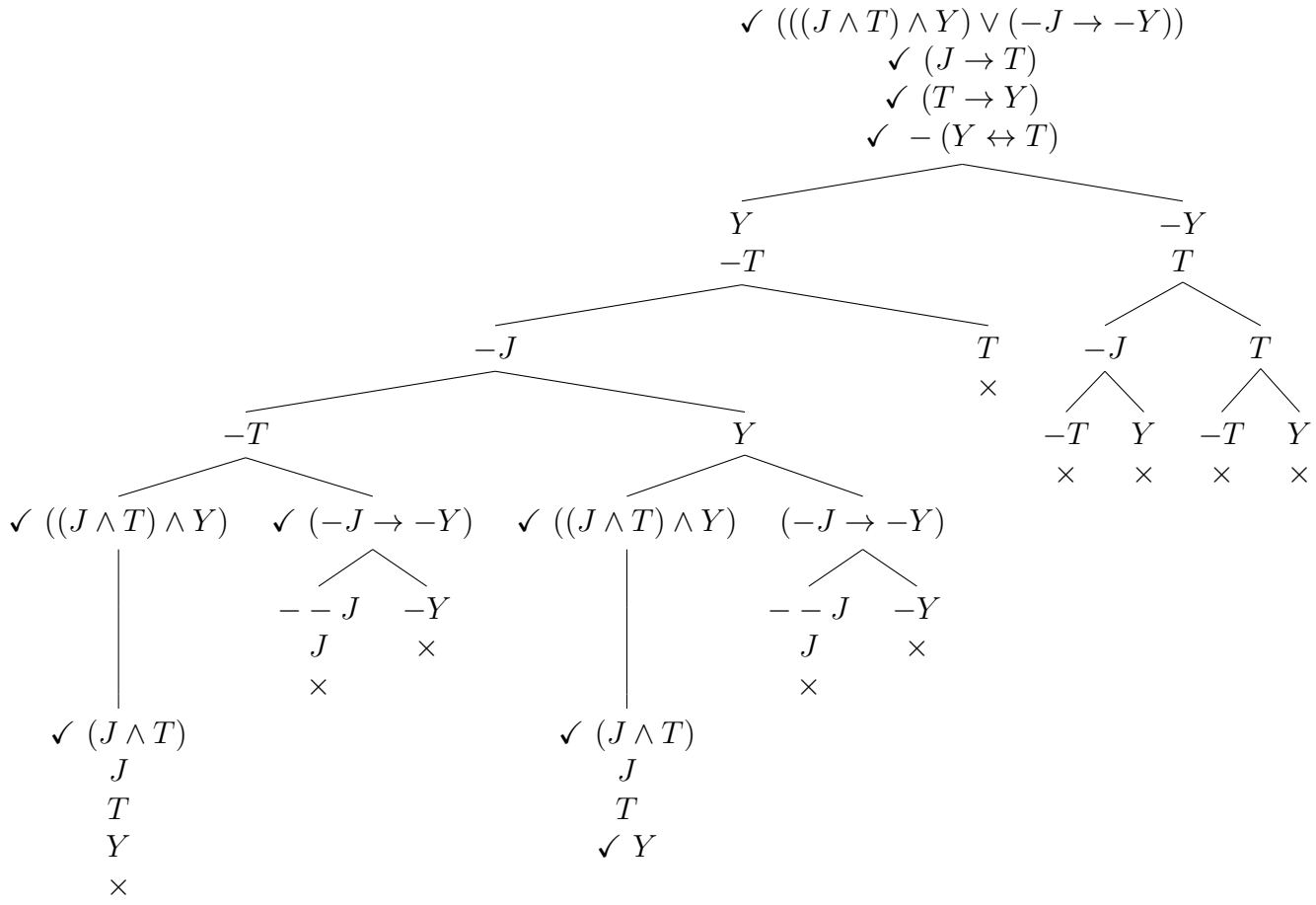
The tree is closed; the argument is valid.

3.



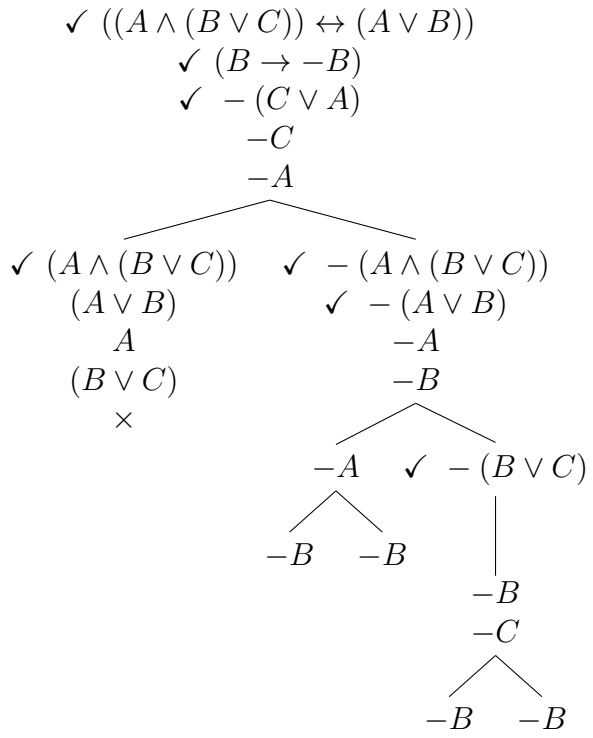
The tree is open; the argument is invalid. A counterexample is given when B is true and C and D are both false.

4.



The tree is closed; the argument is valid.

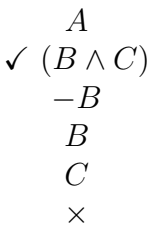
5.



The tree is open; the argument is invalid. A counterexample is given when A, B, C are all false.

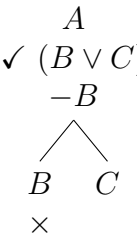
Solutions for Problems 1.9.5.1

1.



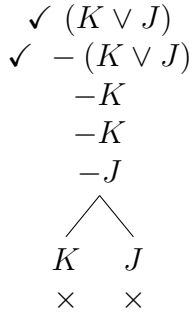
The tree is closed; the implication holds.

2.



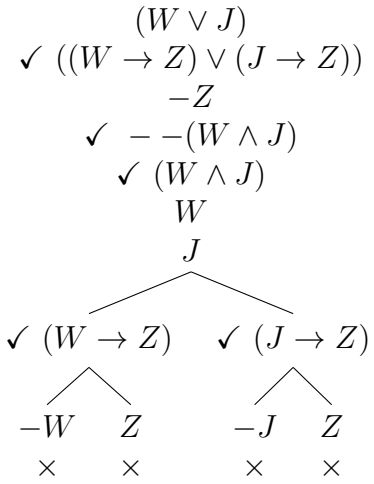
The tree is open; the implication fails. The counterexample is given when A, C are true and B is false.

3.



The tree is closed; the implication holds.

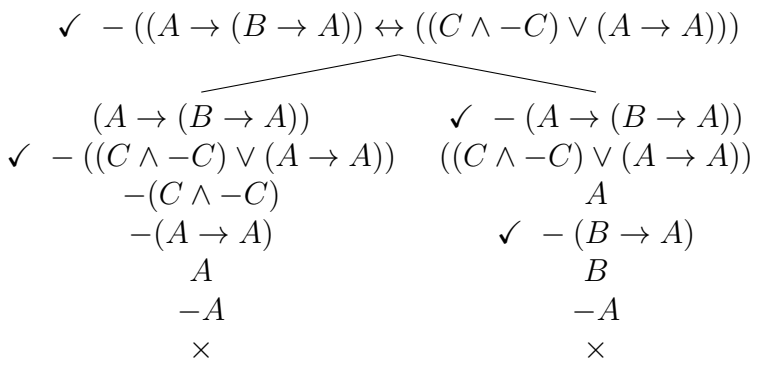
4.



The tree is closed; the implication holds.

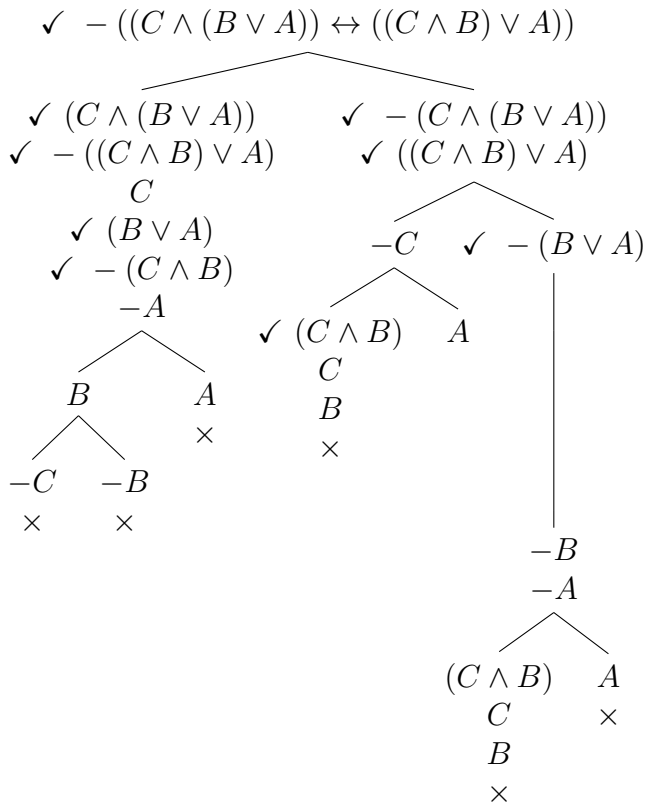
Solutions for Problems 1.9.6.1

1.



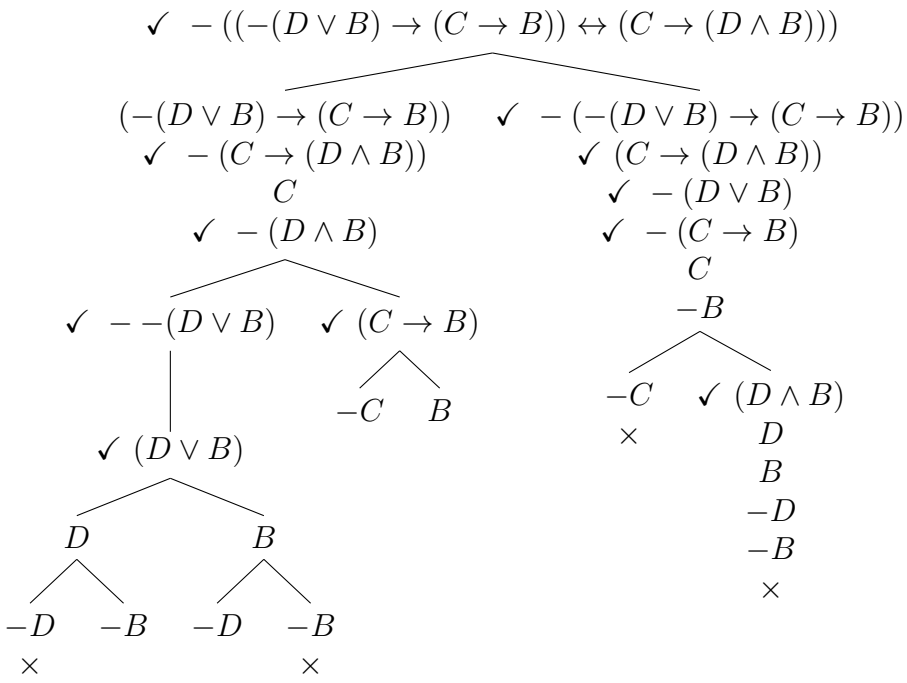
The tree is closed; the equivalence holds.

2.



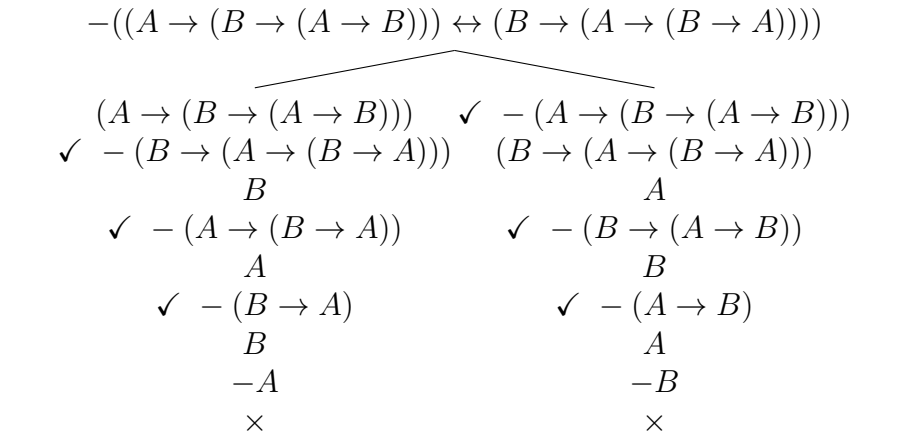
The tree is open; the equivalence fails.

3.



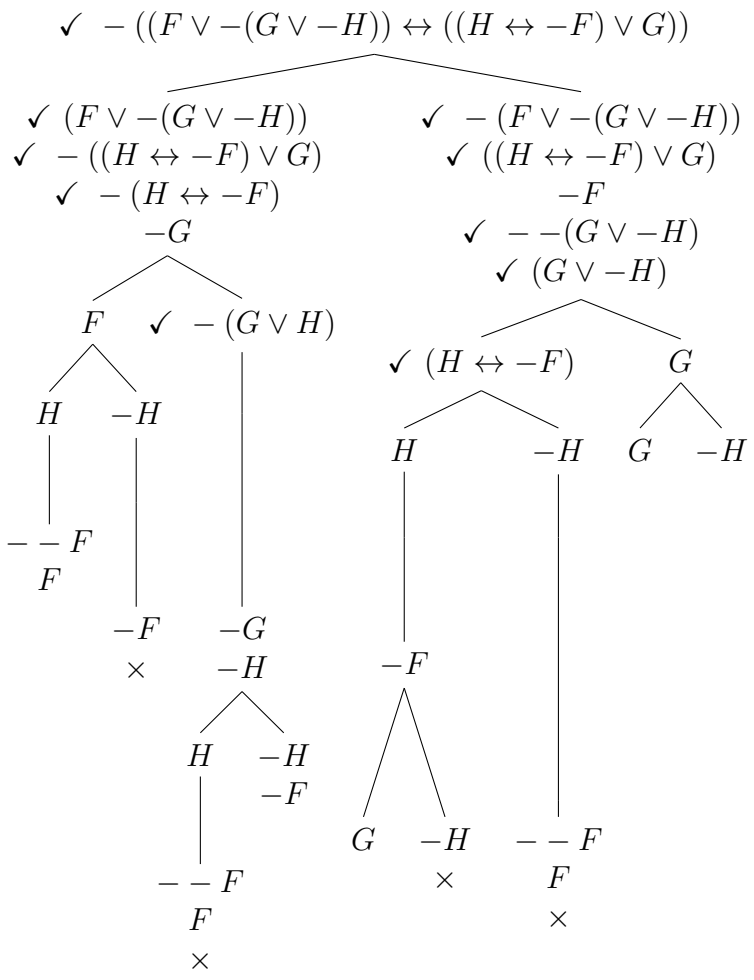
The tree is open; the equivalence fails.

4.



The tree is closed; the equivalence holds.

5.



The tree is open; the equivalence fails.

Solutions for Problems 2.2.3.1

1. $-\exists x((Rx \wedge Ax) \wedge Dx)$
2. $\exists x((Rx \wedge Ax) \wedge Dx)$
3. $-\forall x((Rx \wedge Ax) \rightarrow Dx)$

Solution for Problem 2.2.4.1

In order to show that two sentences are logically equivalent, you need to show that these sentences are mutually implies; namely, given α and β, you need to show $\alpha \models \beta$ and $\beta \models \alpha$.

First, $\forall x((Px \wedge Nx \wedge -Tx) \rightarrow Ox) \models \forall x((Px \wedge Nx) \rightarrow (Ox \vee Tx))$.

$$
\begin{aligned}
& \forall x((Px \wedge Nx \wedge -Tx) \rightarrow Ox) && (4.1)\\
\equiv\ & \forall x(-(Px \wedge Nx \wedge -Tx) \vee Ox) && (4.2)\\
\equiv\ & \forall x((-Px \vee -Nx \vee --Tx) \vee Ox) && (4.3)\\
\equiv\ & \forall x((-Px \vee -Nx \vee Tx) \vee Ox) && (4.4)\\
\equiv\ & \forall x((-Px \vee -Nx) \vee (Ox \vee Tx)) && (4.5)\\
\equiv\ & \forall x(-(Px \wedge Nx) \vee (Ox \vee Tx)) && (4.6)\\
\equiv\ & \forall x((Px \wedge Nx) \rightarrow (Ox \vee Tx)) && (4.7)
\end{aligned}
$$

Annotations:

(1) The starting sentence.
(2) Transformed a conditional to a disjunction using $(\alpha \rightarrow \beta) \equiv (-\alpha \vee \beta)$
(3) Applied one of the De Morgan's laws $(-(\alpha \wedge \beta) \equiv (-\alpha \vee -\beta))$ to $-(Px \wedge Nx \wedge -Tx)$. (Here, you need to apply it twice. Details: $-(Px \wedge Nx \wedge -Tx) \models -(Px \wedge (Nx \wedge -Tx)) \models (-Px \vee -(Nx \wedge -Tx)) \models (-Px \vee (-Nx \vee --Tx)) \models (-Px \vee -Nx \vee --Tx)$.)
(4) Canceled the double negations in $--Tx$.
(5) Changed the order and put the extra pair of the brackets.
(6) Applied the De Morgan to $(-Px \vee -Nx)$.
(7) Changed a disjunction to a conditional.

Second, $\forall x((Px \wedge Nx) \rightarrow (Ox \vee Tx)) \models \forall x((Px \wedge Nx \wedge -Tx) \rightarrow Ox)$.

$$
\begin{aligned}
& \forall x((Px \wedge Nx) \rightarrow (Ox \vee Tx)) && (4.1)\\
\equiv\ & \forall x(-(Px \wedge Nx) \vee (Ox \vee Tx)) && (4.2)\\
\equiv\ & \forall x((-Px \vee -Nx) \vee (Ox \vee Tx)) && (4.3)\\
\equiv\ & \forall x((-Px \vee -Nx \vee Tx) \vee Ox) && (4.4)\\
\equiv\ & \forall x((-Px \vee -Nx \vee --Tx) \vee Ox) && (4.5)\\
\equiv\ & \forall x(-(Px \wedge Nx \wedge -Tx) \vee Ox) && (4.6)\\
\equiv\ & \forall x((Px \wedge Nx \wedge -Tx) \rightarrow Ox) && (4.7)
\end{aligned}
$$

Annotations:
(1) The starting sentence.
(2) Changed a disjunction to a conditional.
(3) Applied the De Morgan.
(4) Moved Tx inside the first pair of the brackets.
(5) Added the double negations to Tx.
(6) Applied the De Morgan.
(7) Changed a disjunction to a conditional.

Solutions for Problems 2.2.6

1.

1. $\forall x(Cxm \to Chx)$
2. $\forall x(Cmx \to Chx)$
3. $\forall x(Cmx \to Chx)$
4. $(\exists xCxm \to Chm)$
5. $(\forall xCxm \to Chm)$
6. $\forall x(Cxh \to Cxm)$
7. $\forall x(Cxh \to Cxm)$

2.

$$\begin{array}{l}
\forall x(Ex \to Cx) \\
\forall x(Lx \to Nx) \\
\forall x(Dx \to Ax) \\
\forall x(-Mx \to -Bx) \\
\forall x(Cx \to Kx) \\
\forall x(-Ex \to -Rx) \\
\forall x(Hx \to -Nx) \\
\forall x(-Bx \to -Kx) \\
\forall x(-Rx \to Dx) \\
\forall x(Mx \to Lx) \\
\hline
\forall x(Hx \to Ax)
\end{array}$$

Solutions for Problems 2.4.6.4

Hint: The explanation can be broken down to the following three points.

1. A tree stops after a finite number of the applications of the rules (decidability); thus, the number of the sentences in the tree has to be finite.

2. If the initial list is satisfiable, there has to be at least one open path in the finished tree (soundness).

3. An interpretation (more precisely, a model) can be constructed from an open path.

First, briefly explain the above points, and then, explain how the claim follows from them.

Solutions for Problems 3.1.3

1.

1. The multiplication of an even number and any number is even.
2. The multiplication of two odd numbers is odd.
3. If the sum of two numbers is even, those two numbers are either both even or both odd.
4. Every prime number other than 2 is odd.
5. There's no largest prime number.

2.

1. $\forall x(Lxa \rightarrow Lax)$
2. $\forall x(Kxa \rightarrow Lxa)$
3. $\exists x \forall y(Lyy \rightarrow Kxy)$
4. $\forall x(Lxa \rightarrow \forall y(-Lya \rightarrow Kxy))$
5. $\forall x(\exists y Lxy \rightarrow \exists z(\exists w Lzw \wedge Lxz))$
6. $\forall x(\forall y(Lya \rightarrow Lxy) \rightarrow Kxa)$

3.

1. $\forall x \exists y Lxy$
2. $\forall x(-Exz \rightarrow \exists y Gxy)$
3. $(-\exists y Lyz \wedge \forall x(-\exists y Lyx \rightarrow Exz))$
   Or $\forall x(-\exists y Lyx \leftrightarrow Exz)$

4.

1. $=$: $(-Lxy \wedge -Lyx)$
2. $\neq$: $(Lxy \vee Lyx)$

Solutions for Problems 3.2.4

1.

1. True. (The meaning of the sentence is: There's someone who doesn't love anyone. In the extension of L, there's no ordered pair of the form $\langle 1, - \rangle$. Thus, 1 is such a person.)

2. True. (The meaning of the sentence is: For everyone, there's someone by whom his/her love is reciprocated. For 1 and 3, their love is reciprocated by themselves; and for 2, his/her love is reciprocated by 1. Thus, everyone's love is reciprocated by someone.)

3. False. (The meaning of the sentence is: There's someone whose love is always reciprocated. However, in the interpretation, no one's love is reciprocated.)

4. True. (It's a bit hard to figure out the meaning of the sentence. So, first transform the sentence into the disjunction of the conjunctions (see pp. 97f), and see if the extension makes the sentence true.)

2.

1. If we take L as a predicate for "... loves ...", the sentence can be translated as "no one's love is reciprocated". Thus, as long as the extension of L doesn't contain any pair of the forms $\langle a, b \rangle$ and $\langle b, a \rangle$, the sentence is going to be true. For example,

   Domain $= \{1, 2, 3\}$
   $v(L) = \{\langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 3, 1 \rangle\}$

   would do.

   For the false part, since the sentence means that no one's love is reciprocated, if there's at least one reciprocated love, the sentence is going to be false. For example

Domain $= \{1, 2, 3\}$
$v(L) = \{\langle 1, 2\rangle, \langle 2, 1\rangle\}$

would do.

2. Again, if we take L as a predicate for "... loves ...", the sentence can be translated as "it's not the case that someone loves everyone and his/her love is always reciprocated"; in other words, the sentence claims that everyone has at least one reciprocated love" (to make clear this reading, transform the sentence into $\forall x \exists y(Lxy \rightarrow Lyx)$). Thus, the interpretation for the question 1 would do.

   For the false part, since the sentence is the negation of "someone loves everyone and his/her love is always reciprocated", the following interpretation makes the original sentence false:

   Domain $= \{1, 2, 3\}$
   $v(L) = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \langle 1, 3\rangle, \langle 2, 1\rangle, \langle 3, 1\rangle\}$.

3. If we take K as a predicate for "... knows ...", the sentence can be translated as "Everyone who knows everyone a knows knows a". Thus, for example, the following interpretation makes the sentence true.

   Domain $= \{1, 2, 3\}$
   $v(a) = 1$
   $v(K) = \{\langle 1, 2\rangle, \langle 3, 1\rangle, \langle 3, 2\rangle\}$

   For the false part, if at least one person knows everyone a knows but that person doesn't know a, the sentence is going to be false. For example, the following interpretation makes the sentence false

   Domain $= \{1, 2, 3\}$
   $v(a) = 1$
   $v(K) = \{\langle 1, 2\rangle, \langle 3, 2\rangle\}$.

4. If we take L and K as in the above, the sentence can be translated as "it's not the case that someone a loves knows everyone a knows"; in other words, the sentence claims that everyone a loves doesn't know someone a knows (to make clear this reading, transform the sentence into $\forall x(Lax \rightarrow \exists y(Kay \wedge -Kxy))$). Thus, for example, the following

interpretation makes the sentence true.

Domain $= \{1, 2, 3\}$
$v(a) = 1$
$v(L) = \{\langle 1, 2\rangle\}$
$v(K) = \{\langle 1, 3\rangle\}$

For the false part, if at least one person a loves knows everyone a knows, the sentence is going to be false. For example, the following interpretation makes the sentence false.

Domain $= \{1, 2, 3\}$
$v(a) = 1$
$v(L) = \{\langle 1, 2\rangle\}$
$v(K) = \{\langle 1, 3\rangle, \langle 2, 3\rangle\}$

5. The sentence simply states that the transitivity law holds for the members of the domain. Thus, for example, the following interpretation makes the sentence true.

   Domain $= \{1, 2, 3\}$
   $v(R) = \{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 1, 3\rangle\}$

   For the false part, the following interpretation makes the sentence false.

   Domain $= \{1, 2, 3\}$
   $v(R) = \{\langle 1, 2\rangle, \langle 2, 3\rangle\}$.

3.

1.

1.	✓ $-\forall x\forall y(Gyx \vee Gxy)$	
2.	$\forall x\forall y(Gxx \vee Gxy)$	
3.	✓ $\exists x\exists y{-}(Gyx \vee Gxy)$	NQ×2 to Line 1
4.	✓ $-(Gba \vee Gab)$	EI×2 to Line 3 with a to x and b to y
5.	$-Gba$	Rule 6 to Line 4
6.	$-Gab$	Rule 6 to Line 4
7.	✓ $(Gaa \vee Gab)$	UI×2 to Line 2 with a to x and b to y
8.	Gaa $\quad$ Gab	Rule 5 to Line 7
9.	✓ $(Gbb \vee Gba)$ $\quad$ ×	UI×2 to Line 2 with b to x and a to y
10.	Gbb $\quad$ Gba	Rule 5 to Line 9
	×	

The leftmost path remains open; thus, from this open path, you can construct the following interpretation.

Domain $= \{1, 2\}$
$Ext(a) = 1$
$Ext(b) = 2$
$Ext(G) = \{\langle 1, 1\rangle, \langle 2, 2\rangle\}$

Note that if you start your tree with $\forall x\forall y(Gyx \vee Gxy)$ and $-\forall x\forall y(Gxx \vee Gxy)$, your tree is gonna be closed. (So you may have to write two truth trees if you're unlucky. If that happens, don't be discouraged; start writing another tree right away.)

2.

1.	✓ $\exists x(Bx \wedge \forall yDyx)$	
2.	✓ $-\forall x(Bx \rightarrow \forall yDyx)$	
3.	✓ $\exists x{-}(Bx \rightarrow \forall yDyx)$	NQ to Line 2
4.	✓ $-(Ba \rightarrow \forall yDya)$	EI to Line 3
5.	Ba	Rule 8 to Line 4
6.	✓ $-\forall yDyx$	Rule 8 to Line 4
7.	✓ $\exists y{-}Dya$	NQ to Line 6
8.	$-Dba$	EI to Line 7
9.	✓ $(Bc \wedge \forall yDyc)$	EI to Line 1
10.	Bc	Rule 3 to Line 9
11.	$\forall yDyc$	Rule 3 to Line 9
12.	Dac	UI to Line 11
13.	Dbc	UI to Line 11
14.	Dcc	UI to Line 11

(In the above, you need to instantiate $\forall yDyc$ in Line 11 with all the names in order to show that you did all you can do.)

From this open tree, you can construct an interpretation as follows.

Domain $= \{1, 2, 3\}$
$Ext(a) = 1$
$Ext(b) = 2$
$Ext(c) = 3$
$Ext(B) = \{1, 3\}$
$Ext(D) = \{\langle 1, 3\rangle, \langle 2, 3\rangle, \langle 3, 3\rangle\}$

Of course, this interpretation makes $\exists x(Bx \wedge \forall yDyx)$ true and $\forall x(Bx \rightarrow \forall yDyx)$ false; thus, the interpretation gives you a correct answer. However, in this case, there's much simpler way to construct an interpretation which makes one of the sentences true and the other false. Notice that the former sentence is an existential sentence with a conjunction and the latter is a universal sentence with a conditional. For the former to be true, there has to be at least one object in the extension of B whereas, for the latter to be true, the empty extension of B would suffice. Therefore, the following interpretation would do as well.

Domain $= \{1, 2, 3\}$
$Ext(B) = \emptyset$
$Ext(D) = \emptyset$

Solutions for Problems 4.1.1

1.

First let's determine whether the sentence is satisfiable or not.

1. ✓ $(\forall x\ x = a \to (\exists x Fx \to \forall x Fx))$
2. ✓ $-\forall x\ x = a$ $\quad (\exists Fx \to \forall x Fx)$
3. ✓ $\exists x x \neq a$
4. $b \neq a$

There's nothing we can do further in order to close the left path. Thus, the tree has at least one finished open path, no matter what's gonna happen to the right path. Therefore, the sentence is satisfiable.

According to the left path, there are two names: a and b. Therefore, we can set up the domain as $\{1, 2\}$ with the extensions of a and b as 1 and 2 respectively. This setting makes $a \neq b$ true obviously.

How about the extension of F? There's no line where the predicate F appears. Thus, we can set up the extension as empty. This makes $\exists x Fx$ false and consequently makes $(\exists x Fx \to \forall x Fx)$ true.

Next let's determine whether the sentence is valid or not.

1. ✓ $-(\forall x\ x = a \to (\exists x Fx \to \forall x Fx))$
2. $\forall x\ x = a$
3. ✓ $-(\exists x Fx \to \forall x Fx)$
4. ✓ $\exists x Fx$
5. ✓ $-\forall x Fx$
6. ✓ $\exists x - Fx$
7. Fb
8. $-Fc$
9. $b = a$
10. $c = a$
11. Fa
12. $-Fa$

The tree is closed; the sentence is valid.

2.

The satisfiability part.

$$\begin{array}{ll} 1. & \forall x \forall y \; x \neq y \\ 2. & a \neq a \\ & \times \end{array}$$

The tree is close; the sentence is unsatisfiable. Under what interpretation is it going to be false? The next tree tells us that.

$$\begin{array}{ll} 1. & \checkmark \; -\forall x \forall y \; x \neq y \\ 2. & \checkmark \; \exists x \exists y \; x \\ 3. & a = b \end{array}$$

There's nothing we can do further; the tree is open. And according to the tree, there are two names; but they refer to the same object in the domain. Thus, one of the interpretations which makes the sentence false is:

Domain: $\{1\}$
Extension of a: 1
Extension of b: 1

3.

$$\begin{array}{ll} 1. & \checkmark \; \exists x \exists y \; x \neq y \\ 2. & a \neq b \end{array}$$

The tree is open; there has to be an interpretation makes the sentence true. And according to the tree, there are two names which refer to different objects of the domain. Thus, one of such interpretations is:

Domain: $\{1, 2\}$
Extension of a: 1
Extension of b: 2

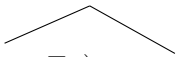
Is the sentence valid? Let's figure it out.

1. ✓ $-\exists x \exists y\ x \neq y$
2. $\forall x \forall y\ x = y$
3. $a = a$

The tree is open; there has to be at least one interpretation which makes the sentence false. And according to the tree, there's only one name: a. Thus, if the domain is comprised of just one object, the sentence is going to be false.

4.

1. $\forall x \forall y((Fx \leftrightarrow Fy) \rightarrow x = y)$
2. ✓ $((Fa \leftrightarrow Fa) \rightarrow a = a)$

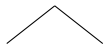


3. $-(Fa \leftrightarrow Fa)$ $\quad a = a$

The right path remains open, no matter what's gonna happen to the left path; thus, the sentence is satisfiable. One of the interpretations which makes the sentence true is one whose domain is comprised of just one member and which has the empty extension for F.

Now the validity part.

1. ✓ $-\forall x \forall y((Fx \leftrightarrow Fy) \rightarrow x = y)$
2. ✓ $\exists x \exists y -((Fx \leftrightarrow Fy) \rightarrow x = y)$
3. ✓ $-((Fa \leftrightarrow Fb) \rightarrow a = b)$
4. ✓ $(Fa \leftrightarrow Fb)$
5. $a \neq b$



6. Fa $\quad -Fa$
7. Fb $\quad -Fb$

The tree is open; the sentence can be false. And from the left path, we can set up one of such interpretations as follows.

Domain: $\{1, 2\}$
Extension of a: 1
Extension of b: 2

Extension of F: $\{1, 2\}$

5.

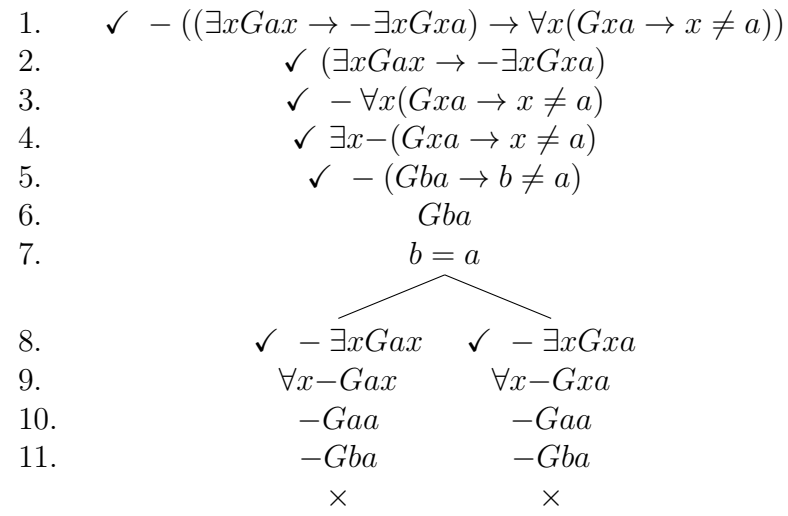
1. ✓ $((\exists x Gax \to -\exists x Gxa) \to \forall x(Gxa \to x \neq a))$

2. ✓ $-(\exists x Gax \to -\exists x Gxa)$ $\quad \forall x(Gxa \to x \neq a)$
3. ✓ $\exists x Gax$
4. ✓ $--\exists x Gxa$
5. ✓ $\exists x Gxa$
6. Gab
7. Gca

The left path remains open, no matter what's gonna happen to the other paths; thus, the sentence is satisfiable. Since there are three names (a, b, c) in the path and there are two G-sentences in the line by themselves, one of the interpretations which makes the sentence true is:

Domain: $\{1, 2, 3\}$
Extension of a: 1
Extension of b: 2
Extension of c: 3
Extension of G: $\{\langle 1, 2\rangle, \langle 3, 1\rangle\}$

Let's move on to the validity part.

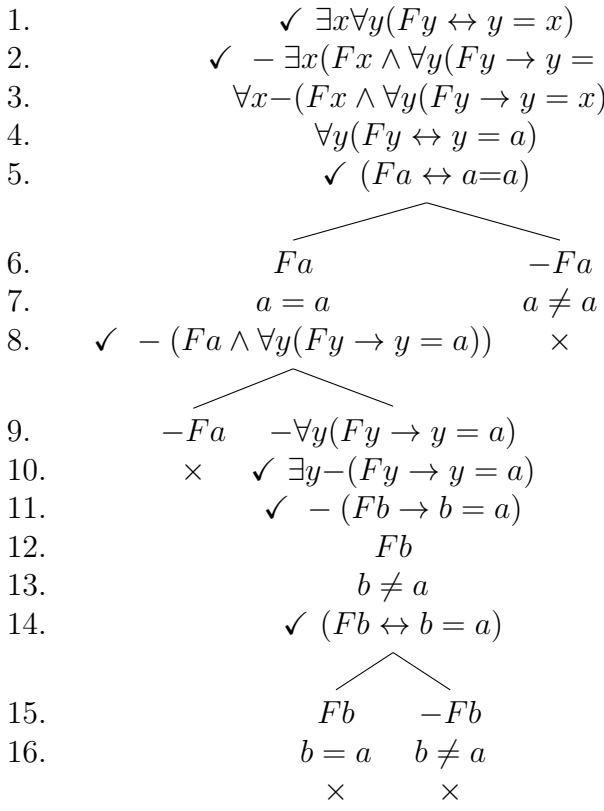


The tree is closed; the sentence is valid.

Solutions for Problems 4.2.3.1

In order to show that the sentences are equivalent with each other, it's enough to show that 1 implies 2, 2 implies 3, and 3 implies 1.

1 implies 2:



The tree is closed; 1 implies 2.

2 implies 3:

1.	$\checkmark\ \exists x(Fx \wedge \forall y(Fy \to y = x))$		
2.	$\checkmark\ -(\exists x Fx \wedge \forall y \forall z((Fy \wedge Fz) \to y = z))$		
3.	$\checkmark\ -\exists x Fx$	$\checkmark\ -\forall y \forall z((Fy \wedge Fz) \to y = z)$	Rule 4 to Line 2
4.	$\forall x{-}Fx$	$\checkmark\ \exists y \exists z{-}((Fy \wedge Fz) \to y = z)$	NQ to Line 3
5.	$\checkmark\ (Fa \wedge \forall y(Fy \to y = a))$	$\checkmark\ (Fa \wedge \forall y(Fy \to y = a))$	EI to Line 1

Let's first finish the left path.

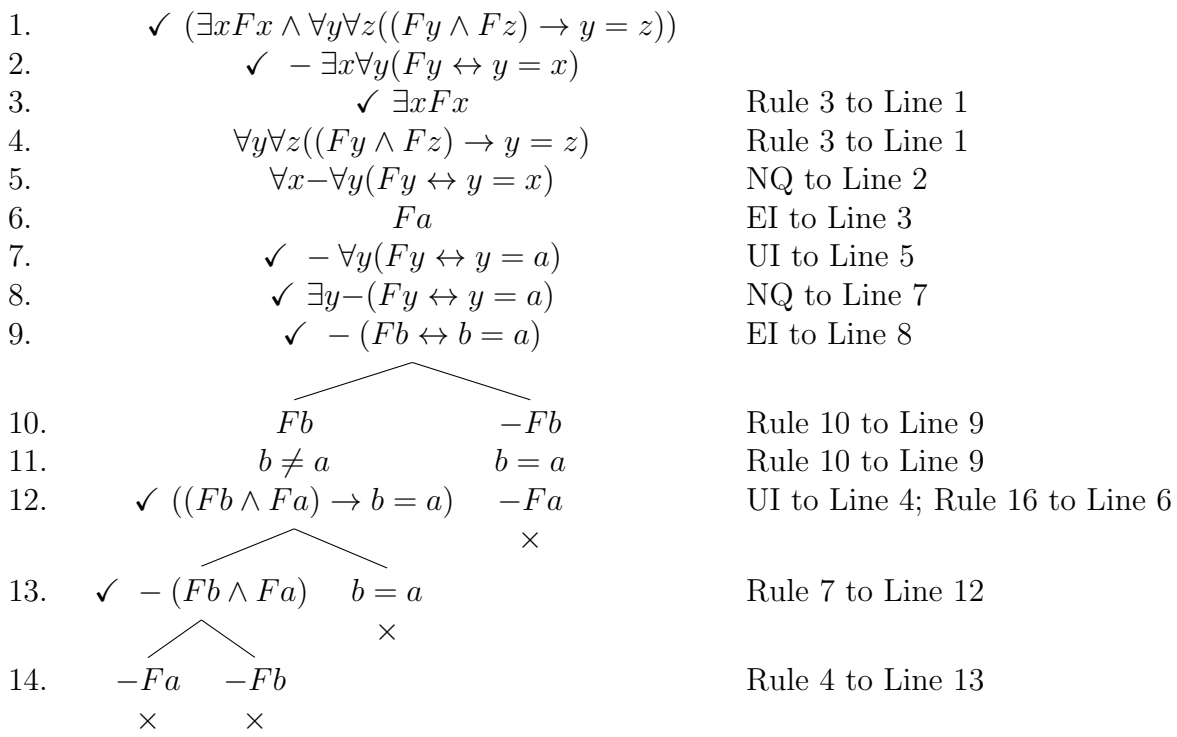
6.	Fa	Rule 3 to Line 5
7.	$\forall y(Fy \to y = a)$	Rule 3 to Line 5
8.	$-Fa$	UI to Line 4
	$\times$	

Next, the right path.

6.	Fa		Rule 3 to Line 5
7.	$\forall y(Fy \to y = a)$		Rule 3 to Line 5
8.	$\checkmark\ -((Fb \wedge Fc) \to b = c)$		EI to Line 4
9.	$\checkmark\ (Fb \wedge Fc)$		Rule 8 to Line 8
10.	$b \neq c$		Rule 8 to Line 8
11.	Fb		Rule 3 to Line 9
12.	Fc		Rule 3 to Line 9
13.	$(Fb \to b = a)$		UI to Line 7
14.	$-Fb$	$b = a$	Rule 7 to Line 13
15.	$\times$	$\checkmark\ (Fc \to c = a)$	UI to Line 7
16.		$-Fc$ $\quad$ $c = a$	Rule 7 to Line 15
17.		$\times$ $\quad$ $b = c$	Rule 17 to Line 14
		$\times$	

The tree is closed; 2 implies 3.

Lastly, 3 implies 1:



The tree is closed; 3 implies 1.

Now we know that (a) 1 implies 2, (b) 2 implies 3, and (c) 3 implies 1. Since implication is transitive, (d) 2 implies 1 (from (b) and (c)); therefore, 1 and 2 are equivalent (from (a) and (d)). Moreover, (e) 3 implies 2 (from (c) and (a)), and (f) 1 implies 3. Therefore, 1 and 3 are equivalent (from (c) and (f)), and 2 and 3 are equivalent (from (b) and (e)).

Solutions for Problems 4.2.7

1. $\exists x \exists y(Lxa \wedge Lya \wedge x \neq y)$
2. $\forall x \forall y((Lax \wedge Lya) \rightarrow x = y)$
3. $\forall x \forall y \forall z((-Lxa \wedge -Lya \wedge -Lza) \rightarrow (x = y \vee y = z \vee z = x))$
4. $\exists x \forall y(\exists v \exists w(v \neq w \wedge Kyv \wedge Kyw \wedge \forall z(Lza \leftrightarrow (z = v \vee z = w))) \leftrightarrow y = x)$