The Logic Course

Maria Serban, Andrei Nasta

January 9, 2014

Contents

Chapter 1

Introduction

1.1 Practical issues

The Benefits of Logic

- Logic captures a standard of rationality: meaningful questions have a definite answer
- It characterizes many common features of sciences.
- In philosophy: clarity, rigour and the lack of ambiguity; it makes informal notions rigorous. Academic philosophy uses logic more and more.
- In linguistics: understanding of related notions, like quantifiers, conjunctions etc. It is an invaluable tool for the semantics of natural language (by similarity or dissimilarity).
- In artificial intelligence, logic provides a way of encoding knowledge, but also ways to establish the limits of such knowledge.
- First-order logic is an artificial language that is to be found in many programming languages.
- First-order logic demystifies formal work.

NB we also talked about the syllabus, our logic website, but we skip those issues here.

1.2 Basic notions

A *proposition* is something expressed by a declarative sentence and which can be true or false. An *argument* is a list of propositions called premises, followed by a word such as ‘therefore’ or ‘then’ and another proposition called the conclusion.

- *Example*
  If everyting is determined, then people are not free. (Premise 1)
  People are free. (Premise 2)
  Then, not everything is determined. (Conclusion)

Is this argument good? There are many ways an argument can be good. In deductive logic, we are interested in the following properties.

Definition 1. An argument is *valid* iff whenever the premises are true the conclusion is also true. Otherwise put, a valid argument cannot have true premises and a false conclusion.[1]

Definition 2. An argument is *sound* just in case it is valid and its premises are true.

[1] Thus, there are three ways a valid argument can be: (i) it can have false premises and a true conclusion, (ii) it can have false premises and a false conclusion, or (iii) it can have true premises and a true conclusion. The last case, (iii), characterizes the sound *and* valid arguments. Compare with the definition of soundness. Note that (i-iii) are only necessary conditions for validity, so you can find invalid arguments that respect (i-iii). These conditions were given to distinguish validity from soundness.

Instance		Argument Form	
It is raining or it is snowing.	(Premise 1)	r or s	(Premise 1)
It is not raining.	(Premise 2)	not r	(Premise 2)
So, it is snowing.	(Conclusion)	$\therefore s$	(Conclusion)

Principles. An instance of a valid argument form is always valid. However, instances of invalid forms *may* be valid.

Invalid Argument Form		Valid Instance	
p	(Premise 1)	r or s	(Premise 1, where $p = r$ or s)
$\neg r$	(Premise 2)	$\neg r$	(Premise 2)
$\therefore s$	(Conclusion)	$\therefore s$	(Conclusion)

Let's review some of the famous argument forms, some of which are invalid.

Modus Ponens If p then q. And in fact p. Therefore, q.
Modus Tollens If p then q. But not q. Therefore, not p.
Affirming the consequent If p then q. q. Therefore, p.
Hypothetical Syllogism If p then q. If q then r. Therefore, if p then r.

1.3 Connectives & Truth-Tables

Atomic and Complex Formulae

- Lower case letters stand for atomic propositions or formulae; capitals are used for complex formulae (atomic propositions can be seen as a species of complex formulae, i.e. formulae having minimal complexity.)
- *Inductive definition of well formed formulae* (*wff*s)

  1. Any atomic formula is a formula.
  2. If A is a formula so is $\neg A$.
  3. If A and B are formulas so is $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$.
  4. Nothing else is a formula.

Look at the following formulae and indicate the main connectors, check whether the rule of parentheses is respected, and tell which formulae are not well formed (wff).

1. $((p \wedge q) \rightarrow r)$
2. $\neg\neg((p \vee \neg q) \leftrightarrow \neg r)$
3. $(((p \rightarrow q) \wedge (q \rightarrow r)) \vee (p \rightarrow r))$
4. $(p \wedge q)\vee)r\neg)$

For instance, consider (1). It is a well formed formula, since it is formed according to the rules given in the inductive definition of wffs above. The main connective of formula (1) is the implication, $\rightarrow$. Since (1) has an equal number of left parentheses and right parentheses, the rule of parentheses is respected.

1.4 Exercises

Note These are the same exercises as on the exercise sheet.

1. Construct truth-tables for complex propositions, and check for tautology, contradiction, contingency.

   a. $((p \wedge q) \rightarrow r) \rightarrow ((p \vee q) \rightarrow r)$
   b. $(p \leftrightarrow q) \wedge ((r \rightarrow \neg p) \wedge (q \rightarrow r))$

c. $(p \leftrightarrow (q \vee r)) \rightarrow (\neg r \rightarrow \neg p)$
d. $(p \leftrightarrow q) \wedge (p \wedge \neg q)$

2. Construct truth-tables for arguments.

   a. $(p \rightarrow q) \therefore (r \rightarrow p) \rightarrow (r \rightarrow q)$
   b. $p \rightarrow (q \rightarrow r) \therefore (p \rightarrow q) \rightarrow r$
   c. $(p \wedge q) \rightarrow r \therefore p \rightarrow (\neg q \vee r)$
   d. $p \rightarrow q, r \rightarrow s \therefore (q \rightarrow r) \rightarrow (p \rightarrow s)$

3. Use the refutation method for checking the validity of the above arguments, and compare it to the truth-tables method.

Chapter 2

Truth-Trees for PL

We introduce a new proof system for propositional logic (PL). Truth-trees are a *mechanical* method, formalizing a natural way of reasoning. Trees give you simple way to prove (metalogical) soundness and completeness.

2.1 Truth-Functionally Valid Inference

Recall that a (truth-functionally) valid inference cannot both have true premises and a false conclusion. (A formula is truth-functional iff its truth-value depends solely on the truth-values of their compounds and their arrangement.)

Instance	Argument form
It is raining or it is snowing.	r or s
It is not raining.	$\neg r$
So, it is snowing.	$\therefore s$

We have shown two methods for checking validity (truth-tables, and the counterexample method). These methods are cumbersome, in different ways, with complex argument forms. We introduce a new method for proving validity: truth-trees.

And now we are going to represent the truth-table of conjunction by tree diagrams. Read upwards, the left hand diagram says that when both p and q are true, so is their conjunction, $p \wedge q$. Similarly, the middle diagram says that if p is false, whatever the truth-value of q, the conjunction $p \wedge q$ is false, and that if q is false, whatever the truth-value of p, then the conjunction $p \wedge q$ is false. So interpreted, the left-hand diagram represents the single row of the truth-table where the conjunction is true. While the middle diagram represents the three rows of the truth-table where the conjunction is false.

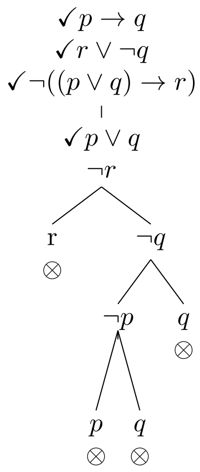
We can eliminate the truth-value tags in the following way. Provided that when any sentence s is false, its negation is true, the middle diagram can be rewritten by replacing formulae of the form $s0$ with formulae of the form $\neg s1$. Thus we obtain the right hand diagram. Subsequently, we shall completely drop the 1s from the trees, thus obtaining the unsigned trees.

The unsigned trees give you the rules for resolving any formula with the truth-tree method. We presented the rules for conjunction. The rules for the other operators are listed below.

2.2 How Trees Work

To test an argument for validity put the premises of the argument and the negation of the conclusion in a list. If these propositions cannot all be true, the argument is valid. If they can be true together, the argument is not valid. Let's see how this works, on the following argument.

$$p \to q, r \lor \neg q \therefore ((p \lor q) \to r)$$

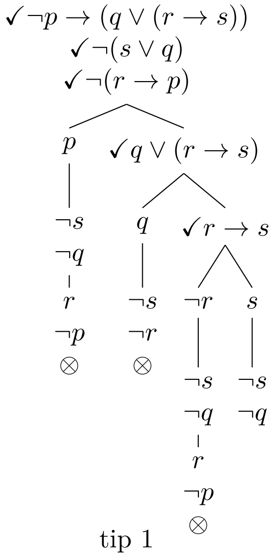


- Some definitions
  - *Closure.* A branch is closed when it contains a formula and its negation. A branch that is not closed is said to be open.
  - *Partially developed trees.* A partially developed tree is a tree wherein some of the formulas are resolved according to the rules.
  - *Completed trees.* A completed tree is a partially developed tree in which, in every open branch, every formula has been resolved.

Note If a completed tree has all its branches closed, then the argument it represents is valid. Otherwise, it is invalid.

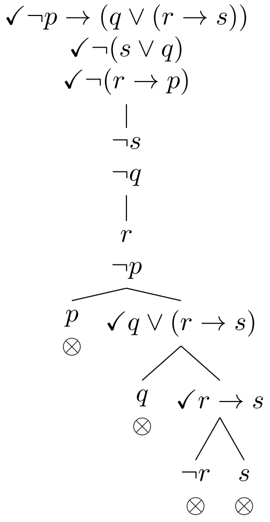
2.3 More trees and some tips

Early branching



tip 1

Late branching

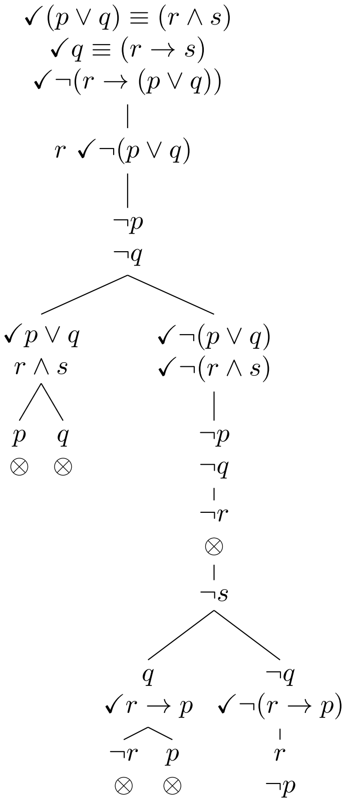


tip 1

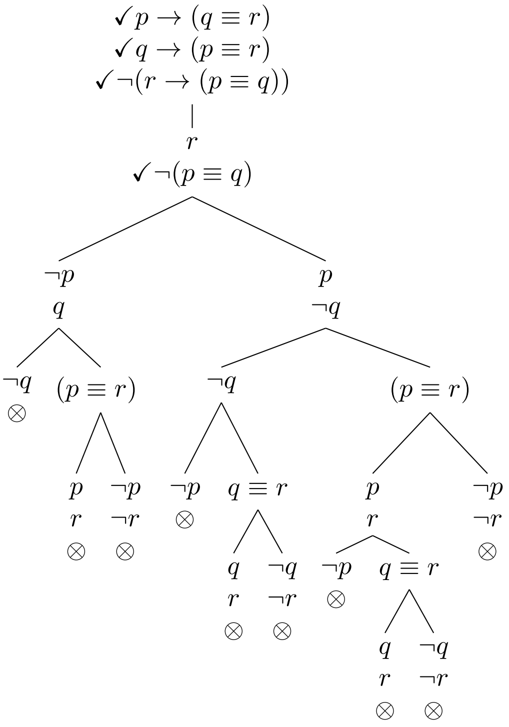
- Tips

1. Use linear rules before branching rules
2. If you think the tree will close, apply rules that lead to closure.

However, these tips are meant to give you an efficient proof strategy, i.e. to shorten the tree-proofs. The validity of the arguments is not affected by the proof strategy.



see tip 2



obscenely large tree

Note The completed tree for an argument gives you a picture of the reasoning used to show that the argument is valid or invalid. A completed tree is a proof.

Note Some trees can be rather large. However, those trees are not more complex than the corresponding truth tables.

2.4 Exercises

1. Construct finished trees for the following set of sentences and say whether the trees are open or closed (Howson ex2 p. 52).

a. $p \to q,\ q \to r,\ \neg r,\ p$
b. $p \to q,\ q \to r,\ \neg r,\ \neg p$
c. $p \lor q,\ \neg q \lor r,\ \neg p,\ \neg r$
d. $p \lor q,\ \neg q \lor r,\ \neg p,\ r$

2. Test the validity of these inferences keeping trees as small as possible (Jeffrey problem 2, p. 27).

a. $\neg p \lor q,\ \neg q \lor r,\ \neg r \lor s \therefore \neg p \lor s$
b. $\neg p \land (q \lor r) \therefore (\neg p \land q) \lor r$
c. $(\neg p \land q) \lor r \therefore \neg p \land (q \lor r)$

3. Identify all counterexamples to each of these inferences (Jeffrey ex1, p.27).

a. $\neg(p \lor q) \therefore \neg p \lor \neg q$
b. $\neg p \lor \neg q \therefore \neg(p \lor q)$
c. $\neg(p \land q) \therefore \neg p \land \neg q$
d. $\neg p \land \neg q \therefore \neg(p \land q)$

4. Symbolize the following argument and test for validity using trees (Restall).

> If a moral theory is studied empirically, then examples of conduct will be considered, and if examples of conduct are considered, principles for selecting examples will be used. But if principles for selecting examples are used, the moral theory is not being studied empirically. Therefore, moral theory is not studied empirically.

5. Construct the trees for the following inferences:

1. $p \to q, q \to (r \lor s), \neg s \therefore p \to r$
2. $(p \land q) \to r, q \to s, r \to \neg s \therefore \neg p$
3. $p \to (r \lor w), q \to s \therefore (p \lor q) \to (r \to (s \lor w))$
4. $p \to (q \lor r), s \to (q \lor p), r \to s \therefore p \leftrightarrow r$
5. $p \lor (q \land r), r \lor (s \land w), (p \lor r) \to (\neg q \lor \neg s) \therefore p \land s$
6. $p \to (q \to (r \to s)), p \land r, r \to q \therefore \neg q \leftrightarrow (s \land \neg s)$
7. $(p \leftrightarrow q) \land (q \leftrightarrow r) \therefore (p \lor \neg p) \land ((q \lor \neg q) \land (r \lor \neg r))$
8. $(p \leftrightarrow q) \lor (q \leftrightarrow r) \therefore p \leftrightarrow (q \lor r)$
9. $(\neg p \land \neg q) \lor r, (p \to s) \land (q \to w), w \to (t \lor v) \therefore \neg t \to (v \lor r)$

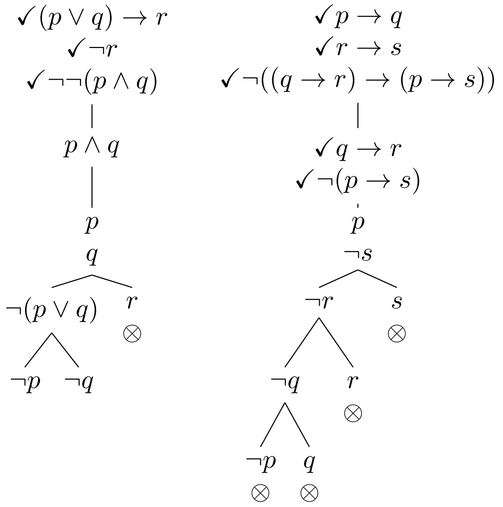
6. Try to solve the exercises from the previous exercise sheet this time using tree method.

Chapter 3

More Trees for PL

3.1 Refutation Trees

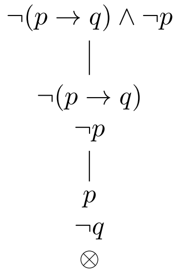
Refutation (or reduction) trees prove the validity of an inference by refuting the hypothesis that the premises together with the negation of the conclusion are consistent (that is, can all true together).

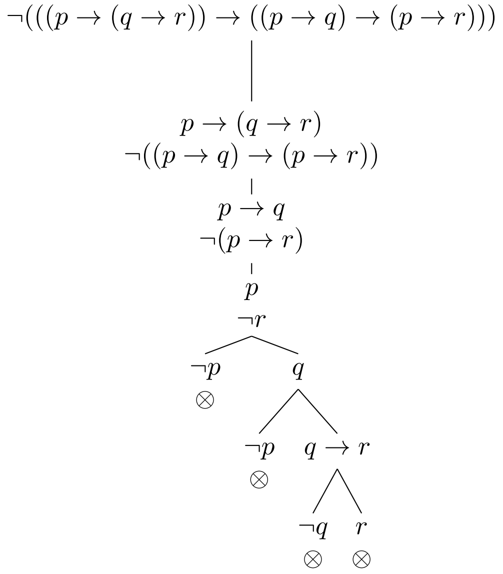


3.2 Trees for Tautologies & Contradictions

We can use truth-trees to test for tautologousness and for contradictoriness. No truth-value distribution over propositions can make a contradiction true, which means that if we construct a truth-tree from it, then the tree will eventually close. Conversely, if the tree closes, then there is no truth-value assignments that make the formula true. The formula is then a contradiction.

We can also construct a tree test for tautologousness. Since the negation of a tautology is a contradiction, by constructing a tree for the negation of the formula to be tested, the tree will close iff the initial formula is a tautology.



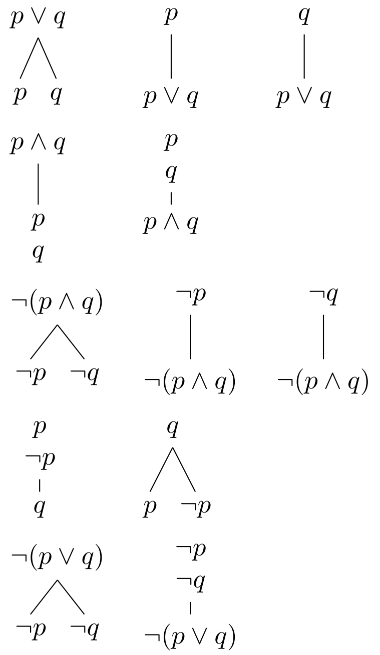


3.3 Deduction Trees

Refutation trees provide indirect proof for the conclusion of the argument. A direct proof would start with the premises and end with the conclusion. In order to get direct proofs we use deduction trees.

Deduction trees start with the premises and end with the conclusion in each open path, *if* the inference is valid and the deduction rules have been followed.

We use the familiar tree rules, called *analytic* rules, as well as their inverses, which are called synthetic rules.



NOTE Deduction trees provide deductions of conclusions from premises iff the inference from those premises to the conclusion is valid.

Exercises

1. Try to solve the exercises from the previous exercise sheet this time using tree method.
2. Construct the trees for the following inferences:

1. $\neg(p \vee q) \therefore \neg p$
2. $(p \wedge q) \therefore (p \vee \neg q)$
3. $(p \vee \neg q), \neg p \therefore \neg\neg q$
4. $(p \wedge s), \neg(s \wedge \neg r) \therefore (r \vee \neg p)$
5. $p, \neg(p \wedge \neg q), (\neg q \vee r) \therefore r$
6. $(p \vee q), \neg(p \wedge \neg r), (r \vee \neg q) \therefore r$
7. $(\neg p \vee \neg(q \vee r)), (q \vee (p \wedge r)) \therefore (\neg p \vee q)$
8. $(p \vee q), \neg(q \wedge \neg\neg r) \therefore \neg(r \vee p)$
9. $(p \wedge (q \vee r)) \therefore (p \wedge q) \vee (p \wedge r)$
10. $(p \vee q), (\neg p \vee r), \neg(q \wedge s) \therefore \neg(\neg r \wedge s)$

3. Construct trees for the following premises. In the case of invalid inferences, list the truth assignments which provide the counterexamples to the validity of the inference.

1. $(p \vee q) \wedge (p \vee r) \therefore (q \wedge r) \vee p$
2. $((p \wedge \neg(q \wedge r)) \vee s), (q \vee s) \therefore (r \wedge s)$
3. $p, \neg(q \wedge (t \vee r)), \neg(\neg q \wedge p), (t \wedge s) \vee (r \wedge s) \therefore s$
4. $\therefore (\neg(\neg(p \wedge q) \vee \neg(p \wedge r)) \vee \neg(p \vee (q \wedge r))$
5. $(p \wedge q) \vee (r \wedge p), \neg(\neg s \wedge p), \neg(q \wedge r) \therefore (q \wedge s)$

Chapter 4

Soundness and Completeness Theorems for PL

4.1 The Language of PL

The aim is to prove rigorously some of the fundamental properties of propositional logic: soundness and completeness. This will establish a relationship between a syntactic property of a set of sentences, namely its generating a closed or open tree, and a semantic property, satisfiability.

Let Σ be a set of propositions, $A, B, C \ldots$ propositional variables. We can represent the language L of propositional logic as $L[A, B, C \ldots \neg, \wedge, \vee, \rightarrow]$.

Next we are going to simplify the tree rules.

An α sentence of L is of the form either $X \wedge Y$, $\neg(X \vee Y)$, or $\neg(X \rightarrow Y)$, where X and Y are sentences of L. Let's further call the first term of each linear branching of figure 1, α_1 and α_2 the second member of the linear branching. Analogously for β sentences: β_1 is the left member and $beta_2$ the right one.

Now, the two groups of trees above can be represented as follows.

These trees are the rules α and β. α_1 and α_2 are descendants of α under the rules (α). Similarly for β_1 and β_2. The α and β rules, together with the double negation rule, are all the rules we need for constructing truth-trees.

We are now in a position to formulate the first theorem.

Theorem 1 If α is any α sentence and β is any β sentence, then $\alpha \Leftrightarrow \alpha_1 \wedge \alpha_2$ and $\beta \Leftrightarrow \beta_1 \vee \beta_2$.

NOTE Any sentence in L is either a literal (atomic proposition), or a double negation sentence, or an α sentence or a β sentence.

Satisfiability A set of sentence is satisfiable iff there is at least an assignment of truth-values to their atomic components which makes all the sentences in the set true as well.

The truth-tree method can test a set of sentences for satisfiability or unsatisfiability. If a finished tree generated by a given set of sentences has an open branch then the set of sentences is satisfiable (completeness). If a finished tree generated from a set of sentences has no open branches, i.e., all its branches are closed, then the set of sentences is unsatisfiable (soundness).

These two facts about the tree method are going to be proved in what follows.

4.2 Soundness

Lemma 1 Suppose that B is a branch on a tree. If all the setences on B are true under some distribution of truth-value to their sentence letters, then B is open.

Proof If B were closed it would contain a sentence and its negation; and both cannot be true together.

Theorem 2 (Soundness) Let Σ be a finite set of senteces of L. If Σ is satisfiable by a truth-value distribution τ over its sentence letters, then any finished tree generated by the rules of tree construction has an open branch, all sentences on which take the value T under τ.

Proof Suppose that a finished tree has been constructed from Σ. Suppose also that we have some distribution τ of truth-values to sentence letters in Σ which makes all the sentences in Σ true. We shall find an open branch in the tree.

1. Let B be the branch-segment consisting just of Σ, i.e. the top node. B is open (by Lemma 1).

2. Now either Σ contains one complex formula or it does not. 2a) If it does not, B is a finished open branch all of whose sentences are true. 2b) If B does contain a complex formula, then it has an extension B' in the tree generated by the application of one the rules (α) or (β) or double negation to some complex sentence on B s.t. all the sentences on B' are true under τ. (i) suppose rule (α) was applied; then the sentences on B' are those in Σ plus α_1 and α_2. Given that all the sentences on B are already true and that α_1 and α_2 are also true (by Theorem 1), it follows that all the sentences on B' will be true as well. (ii) if (β) was applied, then one of β_1 or β_2 must be true (according to Theorem 1). suppose that β_1 is true ; in this case, let B' be the extension of B whose nodes are Σ and β_1. Again, all the sentences on B' are true. (iii) Finally, if the rule of double negation was applied to a sentence $\neg\neg X$, then let B' be the extension of B which includes X. Since by assumption $\neg\neg X$ is true, so is X and so all the sentences on B' are true as well. In each case, since B' contains only true sentences, B' is open by Lemma 1. If B' is not finished then an extension B'' is obtained by applying one of the rules above. Continuing in the same way, we obtain a sequence of extensions of open branches $B, B', B'' \dots$ in the tree. The sequence terminates at some finite stage, and when it does so we have a finished open branch, as required.

4.3 Completeness

Lemma 2 Suppose B is an open branch in a finished tree and let X be any sentence on B. Then, if X is an α sentence, both α_1 and α_2 will appear on B below X. Similarly for β and double negation sentences.

Definition The *weight* of a connector is 1 if the connective is negation ($\neg$), and 2 for all other connectives ($\wedge, \vee, \rightarrow$). The *degree* of a sentence X is the sum of the weights of each connective occurring in X. e.g., $B \vee \neg B$ has degree 3 (=2+1);

Lemma 3 For any α sentence X in L, the degrees of α_1 and α_2 are each less than the degree of X, and if X is a β sentence, the degrees of β_1 and β_2 are each less than the degree of X.

Lemma 4 Let Δ be any set of L sentences and k any integer. Suppose that (i) all the sentences of degree $\leq k$ in $|Delta$ have some property P and (ii) where X is any sentence of degree $> k$ in Δ, if all sentences in Δ of lower degree have P, so does X. Then all the sentences in Δ have P.

Lemma 5 The only sentences of degree o or 1 in L are the literals of L.

Theorem 3 (Completeness) Let Σ be a set of sentences of L. Every finished open branch in a tree generated from Σ determines a truth-value distribution over the sentence letters in Σ which satisfies all the sentences in Σ.

Proof Suppose that a finished open tree has been generated from Σ, and let B be an open branch on the tree. Further suppose that τ is any distribution of truth-values to the sentence letters in Σ s.t. every literal on B takes the value true. (Such a distribution exists because B is open by assumption.) Now we identify the set Δ in Lemma 4 with the set of all sentences on B and we define a property P of sentences (in the same lemma) as follows: a sentence X in $|Delta$ has P just in case X is assigned the value true by τ.

Proving Lemma 4 by induction in this context has 2 steps: (1) Since there is at least one literal on B (since B is opened and finished), the lowest degree of sentences in Δ is 0 or 1. But all the sentences on B of degrees 0 or 1 are literals (cf. Lemma 5) and are true under τ by assumption. (Thus step 1 is established). (2) Consider any sentence X on B of degree greater than 1. Suppose that every sentence on B of degree lower than that of X is true under τ (the inductive hypothesis). From this assumption we show that X is assigned true by τ.

We shall establish the induction step for each of the following cases: (i) X is an α sentence: if X is an α sentence, then α_1 and α_2 are also in B (by Lemma 1) and both are of degree less than X (by Lemma 3). So, by the inductive hypothesis, they are both true under τ. Hence, by Theorem 1, so is X. (ii) X is a β sentence: if X is a β sentence, then one of β_1 or β_2 is on B (by Lemma 1) and both are of degree less than X (by Lemma 3). By the inductive hypothesis, β_1 is true under τ, and so is X. (iii) X is a doubly negated sentence: if X is multiply negated, it has the form $\neg\neg Y$ and at some point, since B is finished, the rule of double negation was applied to X. Hence, Y is on B. But Y has smaller degree than X, and so Y is true under τ (by the inductive hypothesis). Hence $\neg\neg Y$ is true and so is X.

In each of the three possible cases, therefore X is true under τ. The induction step 2 is now complete. So, by Lemma 4, every sentence on B is true under τ. In particular, all the members of Σ are true under τ, since they are all on B.

Chapter 5

Introduction to FOL

In this section we look at the basic notions of first-order logic.

5.1 Predicates & Constants

Consider the following argument.

All male philosophers have beards.
Socrates is a male philosopher.
Therefore, Socrates has a beard.

All tortoises are toothless.
Zeno is a tortoise.
Therefore, Zeno is toothless.

These arguments, which have actually the same logical form, are obviously valid. However, if we formalize them according to propositional logic (PL), $p, q \therefore r$, the arguments appear to be invalid. This is because PL cannot capture some relevant structure of these previous inferences. By contrast, first-order logic (FOL) is capable to capture the relevant structure, and by rendering the arguments using FOL, they appear to be valid, as expected. Here is a gloss of the reasoning expressed by the arguments above according to FOL.

All Fs are Gs.
a is F.
Therefore, a is a G.

This form is valid, and we will be able to prove it soon, using the tree method. The form employs *predicates*, F, G, and this is a general feature of the sentences formed in FOL. (FOL is also called *predicate logic* due to its use of predicate symbols.) The argument also employs constants, i.e. terms like a which stand for individuals (e.g. for Socrates, Zeno etc.), as names do.[1]

Constants name things, predicates describe them. There are *monadic* predicates, e.g. "... has a beard", *dyadic* predicates (e.g. '... is greater than ...') *triadic* predicates (e.g. 'is between ... and ...') and so on. The predicates that have two or more places (the dyadic, triadic predicates etc.) are also called *relations.* (The individual constants that fill the places

[1] We said that the argument uses constants, and this is true in a technical sense as well. We *use* the name 'Socrates' when we talk about Socrates, e.g. in saying that Socrates has a beard. We *mention* the name 'Socrates' when we talk about a word – a string of signs or a sequence of sounds – as in, e.g., "Socrates" has eight letters. The technical distinction we have just presented is that between *use* and *mention.*

of a predicate are also called the *arguments* of the predicate.) We are now able to form sentences combining names and predicates:
India is big. Bi
John loves Mary. Ljm
John and Mary love each other. $Ljm \wedge Lmj$
John and Mary love themselves. $Ljj \wedge Lmm$
Mary's love for John is not reciprocated. $Lmj \wedge \neg Ljm$
If Mary loves John, John doesn't love Mary. $Lmj \to \neg Ljm$

5.2 Quantifiers & Variables

The expressive power of first-order logic is primarily due to the so called quantifiers $\forall$, $\exists$. Consider the following sentences.

1. Some male philosophers have beards.
2. Most logic students are intelligent.
3. All logic students are in the classroom.
4. At least two philosophers have no beard.

Roughly speaking, quantified sentences talk about *quantities* of objects that satisfy a condition or have a property: *all*, *some*, *most* and so on. In order to explain how quantifiers work, we need to introduce a new kind of symbol, namely the *variable.* Variables behave similarly to individual constants because they appear in the list of arguments of a predicate. However, they are also different because they are not used to name individual objects, rather they are ranging over a particular domain of individuals. They function as place-holders that signal the relationship between quantifiers and the arguments of predicates.

Let's formalize some examples with quantifiers and variables, using the logical connectives and predicates already familiar.

Existential Quantifier ($\exists$)

This symbol is used to express claims which in English are introduced by terms such as 'something', 'at least one thing', 'a', 'an'. The quantifier is always used in connection with a variable, and thus is said to be a *variable binding operator.* $\exists x$ is read *for some object x.* Let's take some examples.

5. (a) Some male philosophers have beards.
   (b) $\exists x((Mx \wedge Px) \wedge Bx)$
6. (a) It is not the case that some male philosopher has a beard.
   (b) $\neg(\exists x)(Mx \wedge Px \wedge Bx)$
7. (a) Some philosopher knows a lawyer.
   (b) $\exists x(Px \wedge x$ knows a layer)
   (c) $\exists x(Px \wedge \exists y(Ly \wedge Kxy))$

Universal Quantifier

The universal quantifier is introduced in English by 'each thing', 'every thing', 'all things', 'every thing' etc.. let $\forall x$ mean *all x.* As with the existential quantifier, the universal quantifier is always used in connection with a variable (i.e. it is called a *variable binding operator*).

8. (a) All the students are in the classroom.
   (b) $\forall x(Sx \to Cx)$

9. (a) Every apple is not in the basket.
   (b) $(\forall x)(Ax \to \neg Bx)$
10. (a) Not all businessmen are rich.
   (b) $\neg(\forall x)(Bx \to Rx)$

Note Both kinds of quantifiers have an associated domain of discourse. It's only relative to this domain that the quantified sentences are true or false.

Note The two quantifiers are inter-translatable. The so called De Morgan Laws for quantifiers allow you to 'push a negation past the quantifier' by switching the quantifier. So if we know that *not everything has some property*, then we know that *something does not have that property*, and the other way around. Likewise, if *it's not the case that something has some property*, then *everything must fail to have it*, and vice versa. In symbols, the two equivalences are rendered as follows:

$$\neg(\forall x)Fx \equiv (\exists x)\neg Fx$$
$$\neg(\exists x)Fx \equiv (\forall x)\neg Fx$$

This shows that any existentially quantified formula $(\exists x)$ can be transformed in a universally quantified sentence $(\forall x)$.

5.3 Language & Translations

To recap, the syntax of FOL contains the following symbols and sequences of symbols:

- *constants* $a, b, c, \ldots$
- *predicates* $F, G, H \ldots$, which have one or more places
- *variables* $x, y, z \ldots$
- *quantifiers* $\forall, \exists$
- *connectives* $\neg, \wedge, \vee, \to, \ldots$
- *formulae* constructed from the above symbols, according to the following well-formedness rules:[2]
  1. Φ is of the form Fa, for any predicate F and any constant a.
  2. Φ has the form $QxFx$, for any quantifier Q, variable x, and predicate F.
  3. Φ has any of the forms Φ_1, $\neg\Phi_1$, $\Phi_1 * \Phi_2$, where $*$ can be any binary connective.
  4. These rules form well-formed formulae (*wff*s) of FOL, and nothing else does.

Note An important notion is that of the *scope* of a quantifier. First let $(\ldots x \ldots)$ be any string of symbols containing (at least) an occurrence of the variable x. Then, the scope of a quantifier Q ($\exists$ or $\forall$) in $Qx(\ldots x \ldots)$ is the string $(\ldots x \ldots)$. The quantifier Q scopes over that string (and any of the symbols in it). An interesting case is the phenomenon of scope ambiguity, which appear when two quantifiers occur in the same sentence. For instance:

1. a. A mistake was made by every student.
   b. $(\exists x)(Mx \wedge (\forall y)(Sy \to Dyx))$
   c. $(\forall y)(Sy \to (\exists x)(Mx \wedge Dyx))$

Note There is a distinction between formulae such as Fa and $\exists Fx$, on the one hand, and formulae such as Fx, Gxy. These formulae mark a double distinction between closed formulae and open ones. The property of a formula of being open or closed depends on whether there are variables in it, and in case there are, on the kind of variables that occurs in the formula. The *closed* formulae are those in which all the variables (if there are any at

[2] The symbols F, a, Q etc. used in defining formulae are not used as above, i.e. they are not predicates, constants etc. but rather variables used to talk about the predicates, constants etc. of the language of FOL. These variables are not part of FOL.

all) are *bound* by a quantifier. A variable x is bound if it is preceded by – or, equivalently, is in the scope of – a quantifier which contains an occurence of the same variable x, as in $\exists x Fx$. The variables that are not bound are *free*. E.g. the variable x in Fx is free, because there is no quantifier that has in the variable x in its scope. The formulae containing free variables are not closed, i.e. are open. All the other formulae are closed (including the formulae which don't have variables at all, e.g. Fa). The free variables are not used in FOL. Hence, the open formulae are not wffs of FOL.

Let's look at some more interesting translations:

1. Everyone loves something.
2. $(\forall x)(Px \rightarrow (\exists y)Lxy)$
3. If you love something you think about it.
4. $(\forall x)(Px \rightarrow (\exists y)Txy)$
5. Everyone thinks about something.
6. $(\forall x)(\forall y)((Pxy \wedge Lxy) \rightarrow Txy)$

NOTE The natural language words are not always translated by the same logical symbol. For instance, *something* expresses an existential quantifier in "I like *something*" and an universal quantifier in "If I like *something*, I think about *it*". In many cases, the structure of the entire natural language sentence should help you decide on a translation.

Chapter 6

Trees for FOL

We shall learn how to develop trees for first-order logic. We shall use trees to decide on whether a first-order formula is a tautology or whether a first-order inference is valid.

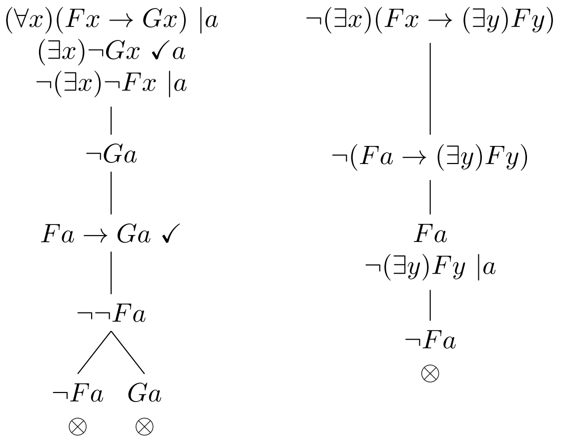
6.1 Tree-Rules for FOL

6.2 Tautology and Validity

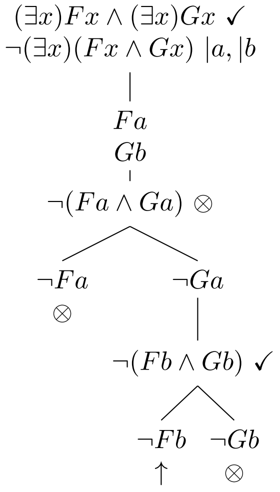
Let's use the new tree-rules for FOL by first checking the validity of the argument $(\forall x)(Fx \to Gx), (\exists x)\neg Gx, \therefore (\neg\exists)\neg Fx$, and then verifying that $(\forall x)Fx \to (\exists)yFy$.

Validity Test

Tautology Test


$(\forall x)(Fx \to Gx)\ |a$
$(\exists x)\neg Gx\ \checkmark a$
$\neg(\exists x)\neg Fx\ |a$
$\neg Ga$
$Fa \to Ga\ \checkmark$
$\neg\neg Fa$
$\neg Fa$
Ga
$\otimes$
$\otimes$
$\neg(\exists x)(Fx \to (\exists y)Fy)$
$\neg(Fa \to (\exists y)Fy)$
Fa
$\neg(\exists y)Fy\ |a$
$\neg Fa$
$\otimes$

The definition of completeness for an open branch in FOL differs from the corresponding definition in propositional logic.



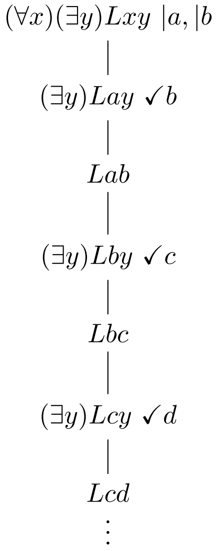
Complete Open Branch

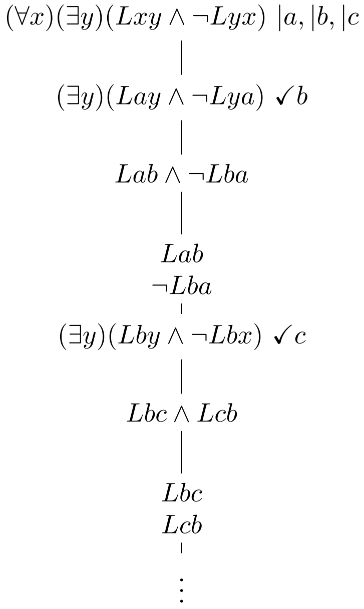
An open branch is complete iff

(i) every resolvable formula has been resolved; and

(ii) every constant in the branch has been substituted into every *general* formula in the branch.

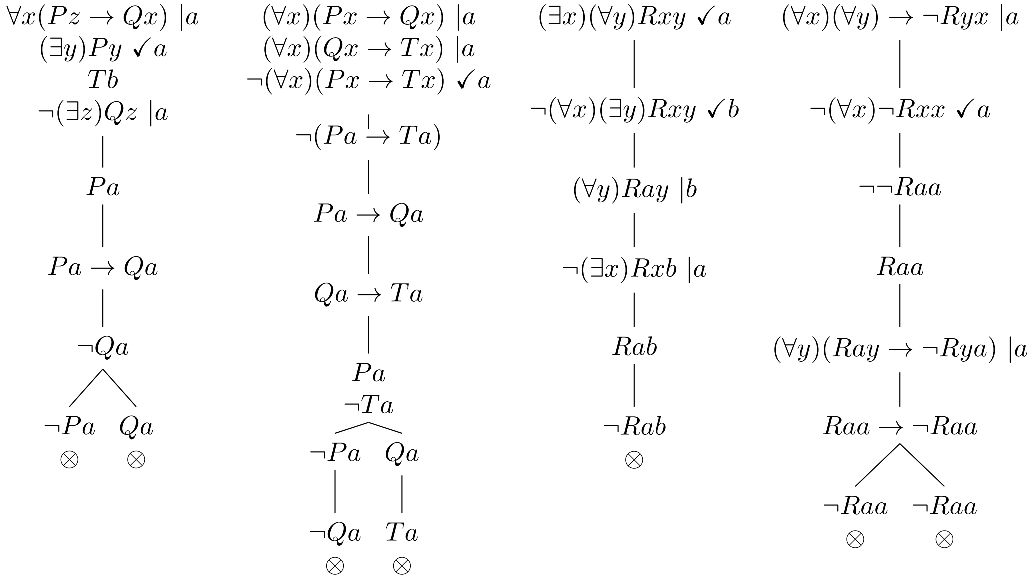
Furthermore, by contrast to the propositional logic, FOL admits trees that go on forever.



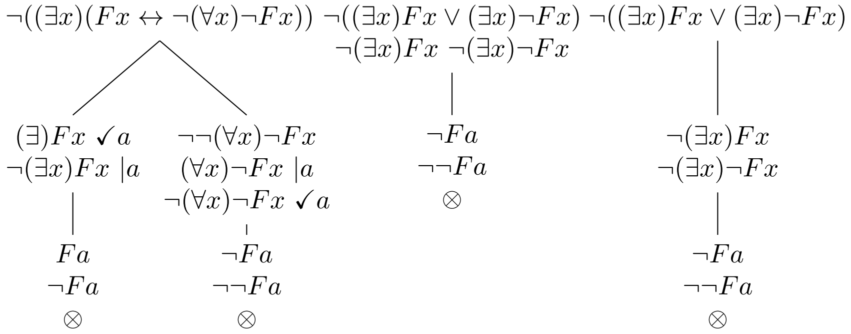

$(\forall x)(\exists y)(Lxy \wedge \neg Lyx)$ $|a, |b, |c$
$(\exists y)(Lay \wedge \neg Lya)$ ✓b
$Lab \wedge \neg Lba$
Lab
$\neg Lba$
$(\exists y)(Lby \wedge \neg Lbx)$ ✓c
$Lbc \wedge Lcb$
Lbc
Lcb
⋮

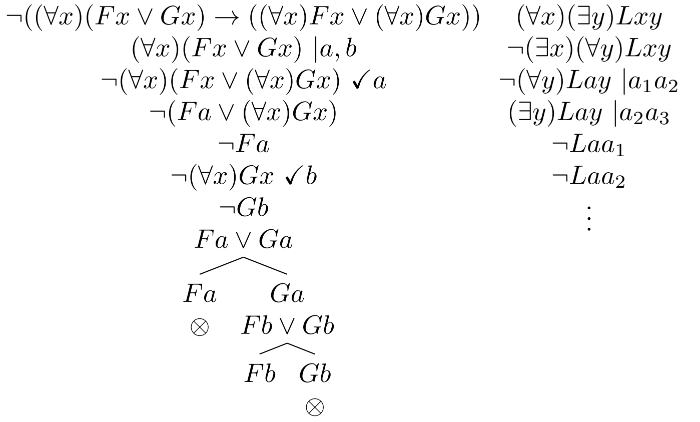
6.3 More Trees

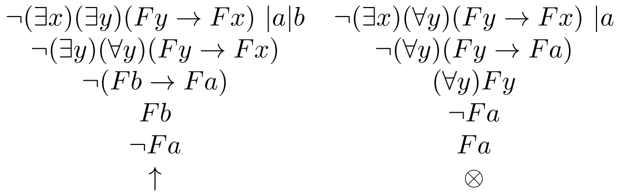
The following trees are due to Howson, chapter 6.

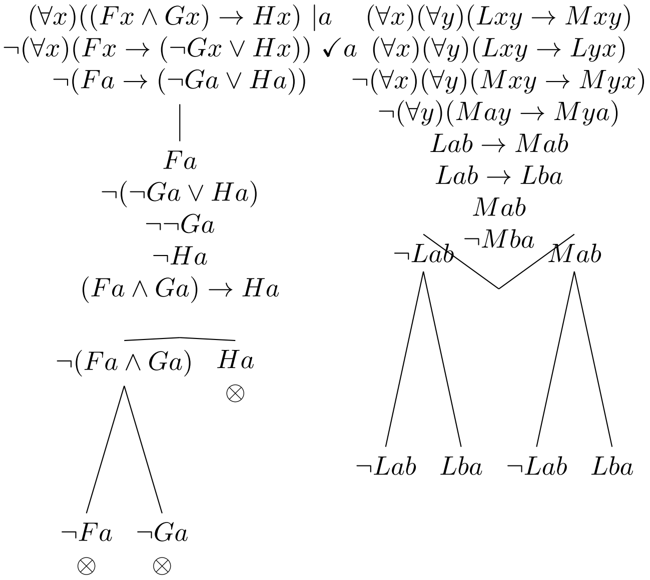

$\forall x(Pz \to Qx)\ |a$
$(\exists y)Py\ \checkmark a$
Tb
$\neg(\exists z)Qz\ |a$
Pa
$Pa \to Qa$
$\neg Qa$
$\neg Pa$ Qa
$\otimes$ $\otimes$
$(\forall x)(Px \to Qx)\ |a$
$(\forall x)(Qx \to Tx)\ |a$
$\neg(\forall x)(Px \to Tx)\ \checkmark a$
$\neg(Pa \to Ta)$
$Pa \to Qa$
$Qa \to Ta$
Pa
$\neg Ta$
$\neg Pa$ Qa
$\neg Qa$ Ta
$\otimes$ $\otimes$
$(\exists x)(\forall y)Rxy\ \checkmark a$
$\neg(\forall x)(\exists y)Rxy\ \checkmark b$
$(\forall y)Ray\ |b$
$\neg(\exists x)Rxb\ |a$
Rab
$\neg Rab$
$\otimes$
$(\forall x)(\forall y) \to \neg Ryx\ |a$
$\neg(\forall x)\neg Rxx\ \checkmark a$
$\neg\neg Raa$
Raa
$(\forall y)(Ray \to \neg Rya)\ |a$
$Raa \to \neg Raa$
$\neg Raa$ $\neg Raa$
$\otimes$ $\otimes$

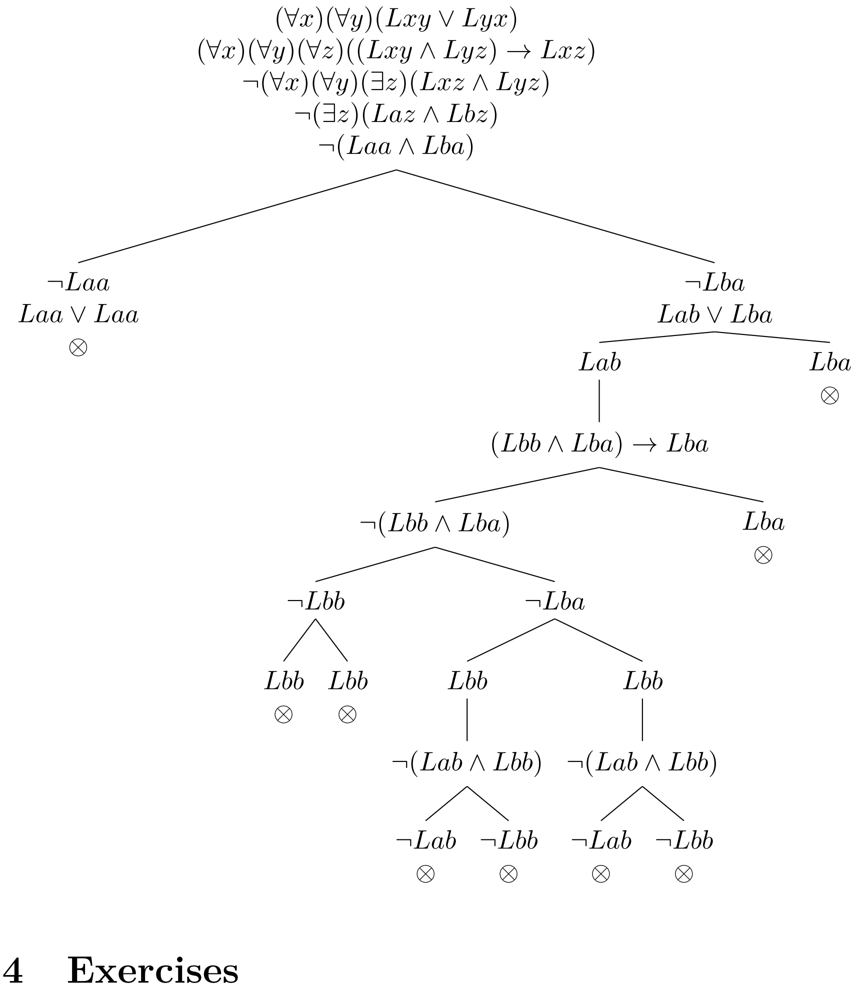
Let's consider the following trees that test validity and tautologousness, and which are from Restall' exercises in chapter 10.


$\neg((\exists x)(Fx \leftrightarrow \neg(\forall x)\neg Fx))$
$(\exists)Fx\ \checkmark a$
$\neg(\exists x)Fx\ |a$
Fa
$\neg Fa$
$\otimes$
$\neg\neg(\forall x)\neg Fx$
$(\forall x)\neg Fx\ |a$
$\neg(\forall x)\neg Fx\ \checkmark a$
$\neg Fa$
$\neg\neg Fa$
$\otimes$
$\neg((\exists x)Fx \vee (\exists x)\neg Fx)$
$\neg(\exists x)Fx\ \neg(\exists x)\neg Fx$
$\neg Fa$
$\neg\neg Fa$
$\otimes$
$\neg((\exists x)Fx \vee (\exists x)\neg Fx)$
$\neg(\exists x)Fx$
$\neg(\exists x)\neg Fx$
$\neg Fa$
$\neg\neg Fa$
$\otimes$


$\neg((\forall x)(Fx \vee Gx) \to ((\forall x)Fx \vee (\forall x)Gx))$
$(\forall x)(Fx \vee Gx)\ |a, b$
$\neg(\forall x)(Fx \vee (\forall x)Gx)\ \checkmark a$
$\neg(Fa \vee (\forall x)Gx)$
$\neg Fa$
$\neg(\forall x)Gx\ \checkmark b$
$\neg Gb$
$Fa \vee Ga$
Fa Ga
$\otimes$ $Fb \vee Gb$
Fb Gb
$\otimes$
$(\forall x)(\exists y)Lxy$
$\neg(\exists x)(\forall y)Lxy$
$\neg(\forall y)Lay\ |a_1a_2$
$(\exists y)Lay\ |a_2a_3$
$\neg Laa_1$
$\neg Laa_2$
$\vdots$


$\neg(\exists x)(\exists y)(Fy \to Fx)\ |a|b$
$\neg(\exists y)(\forall y)(Fy \to Fx)$
$\neg(Fb \to Fa)$
Fb
$\neg Fa$
$\uparrow$
$\neg(\exists x)(\forall y)(Fy \to Fx)\ |a$
$\neg(\forall y)(Fy \to Fa)$
$(\forall y)Fy$
$\neg Fa$
Fa
$\otimes$





6.4 Exercises

1. Test the following formulae using trees. Are any of the formulae tautologies? For those that are not, present counterexamples (cf. *Restall* 164: 10.1):

1. $(\forall x)(Fx \vee Gx) \rightarrow ((\forall x)Fx \vee (\forall x)Gx)$

2. $(\forall x)(Fx \rightarrow Gx) \rightarrow ((\forall x)Fx \rightarrow (\forall x)Gx)$
3. $(\forall x)(Fx \vee Gx) \rightarrow ((\forall x)Fx \vee (\exists x)Gx)$
4. $(\exists x)(Fx \rightarrow (\forall y)Fy)$
5. $((\exists x)((\exists y)Fy \rightarrow Fx)$
6. $(\exists x)(\forall y)(Fy \rightarrow Fx)$

2. Test these argument forms for validity, using trees. Present counterexamples to any invalid argument forms you find. (cf. *Restall* 164: 10.2)

1. $(\forall x)((Fx \wedge Gx) \rightarrow Hx) \vdash (\forall x)(Fx \rightarrow (\neg Gx \vee Hx))$
2. $(\forall x)(\forall y)(Lxy \rightarrow Mxy), (\forall x)(\forall y)(Lxy \rightarrow Lyx) \vdash (\forall x)(\forall y)(Mxy \rightarrow Myx)$
3. $(\forall x)(\exists y)Lxy \vdash ((\exists x)(\forall y)Lxy$
4. $(\forall x)(\forall y)(Lxy \rightarrow Lyx), (\forall x)(\forall y)(\forall z)((Lxy \wedge Lyz) \rightarrow Lxz) \vdash (\forall x)((\exists y)Lxy \rightarrow Lxx)$
5. $(\forall x)(\forall y)(Lxy \vee Lyx), (\forall x)(\forall y)(\forall z)((Lxy \wedge Lyz) \rightarrow Lxz) \vdash (\forall x)(\forall y)(\exists z)(Lxz \wedge Lyz)$

3. A dyadic relation R is said to have no dead ends iff $(\forall x)(\exists y)Rxy$. Show that all symmetric, transitive relations without dead ends are also reflexive by showing that:

- $(\forall x)(\forall y)(Rxy \rightarrow Ryx), (\forall x)(\forall y)(\forall z)((Rxy \wedge Ryz) \rightarrow Rxz), (\forall x)(\exists y)Rxy \vdash (\forall x)Rxx.$

6.5 Models and Countermodels

Exercises - 22 November

1. Check for validity the following arguments:

1. $(\forall x)(Fx \to Gx) \vdash (\forall x)(Gx \to Fx)$
2. $(\forall x)(Fx \to Gx) \vdash (\forall x)(\neg Gx \to \neg Fx)$
3. $(\forall x)(\exists y)Lxy \vdash (\forall y)(\exists x)Lyx$
4. $(\forall x)((Fx \land Gx) \to Hx) \vdash (\forall x)(Fx \to (Gx \to Hx))$
5. $(\forall x)Fx \lor (\forall x)Gx \vdash (\forall x)(Fx \lor Gx)$
6. $(\forall x)(Fx \to Gx), (\forall x)(\neg Gx \to Hx) \vdash (\forall x)(Fx \to \neg Hx)$
7. $(\exists x)(Fx \land Gx), (\forall x)(\neg Hx \to \neg Gx) \vdash (\exists x)(Fx \land Hx)$
8. $(\forall x)(\exists y)(Fy \to Gx) \vdash (\forall y)(\exists x)(Gx \to Fy)$
9. $(\forall x)((\forall y)(Lxy \to Lyx)) \vdash (\forall x)Lxx$

2. Take the following model:

- The domain is {Romeo, Juliet, Benedick, Beatrice}.
- Constants are assigned references as follows:
  - m: Romeo
  - n: Juliet
- Predicates are assigned extensions as follows:
  - F: {Romeo, Benedick}
  - G: {Juliet, Beatrice}
  - L: {(Romeo, Juliet),(Juliet, Romeo),(Benedick, Beatrice),(Beatrice, Benedick),(Benedick, Benedick)}

What are the truth-values of the following formulae?

1. $(\exists x)Lmx$
2. $(\forall x)Lxm$
3. $(\exists x)Lmx \to Lmn$
4. $(\forall x)(Fx \leftrightarrow \neg Gx)$
5. $(\forall x)(Gx \to (Lxm \lor \neg Lmx))$
6. $(\forall x)(Gx \to (\exists y)Lxy)$
7. $(\exists x)(Fx \land (\forall y)(Gy \to Lxy))$

Now take the following model:

- The domain is {4, 7, 8, 11, 12 }
- Constants are assigned references as follows:
  - m: 7
  - n: 12
- Predicates are assigned extensions as follows:
  - F: the even numbers in the domain
  - G: the odd numbers in the domain
  - L: the set of pairs (m, n) where m and n are in the domain and m is less than n.

What are the truth-values of the formulae (1) to (7) above?

3. Test the following formulae in two-element domains, using any technique you wish. Which are true in each model? Which are false in some? For those that have counterexamples, present a counterexample as a table of values for F and G.

1. $(\forall x)(Fx \to Gx) \lor (\forall x)(Fx \to \neg Gx)$
2. $(\forall x)(Fx \to (Gx \lor \neg Gx))$
3. $(\exists x)Fx \lor (\exists x)\neg Fx$
4. $(\forall x)(Fx \lor \neg Fx)$
5. $(\forall x)Fx \lor (\forall x)\neg Fx$
6. $(\forall x)Fx \lor (\exists x)\neg Fx$
7. $(\forall x)Fx \to (\exists x)Fx$
8. $(\exists x)(Fx \to (\forall y)Fy)$
9. $(\forall x)(Fx \to (\forall y)Fy)$
10. $(\forall x)(Fx \to (\exists y)Fy)$

4. Test the following arguments in three-element worlds. For any that are not valid, present the counterexamples in a table.

1. $(\forall x)(Gx \to Fx) \vdash (\forall x)(\neg Gxto\neg Hx)$
2. $(\forall x)Gx \to (\forall x)Hx \vdash (\forall x)(Gx \to Hx)$
3. $(\forall x)Gx \land (\forall x)Hx \vdash (\forall x)(Gx \land Hx)$
4. $(\exists x)(Gx \leftrightarrow Hx) \vdash \neg(\forall x)(Gx \to Hx)$
5. $(\exists x)(Gx \land Hx) \vdash (\exists x)Gx \land (\exists x)Hx$
6. $(\forall x)(Gx \to (\forall y)Hy) \vdash (\forall x)(Gx \to Hx)$
7. $(\forall x)(Gx \to Hx), (\forall x)Gx \vdash (\forall x)Hx$

Bibliography

[1] Barwise, J. and Etchemendy, J. *The language of first-order logic.* CSLI, Stanford, CA, (1991).

[2] Boolos, G., Burgess, J. P., and Jeffrey, R. C. *Computability and logic.* Cambridge University Press, Cambridge; New York, (2007).

[3] Forbes, G. *Modern logic: a text in elementary symbolic logic.* Oxford University Press, (1994).

[4] Halbach, V. *The logic manual.* Oxford University Press, Oxford; New York, (2010).

[5] Howson, C. *Logic with trees an introduction to symbolic logic.* Routledge, (1997).

[6] Jeffrey, R. C. and Burgess, J. P. *Formal logic : its scope and limits.* Hackett Pub. Co., Indianapolis, (2006).

[7] Restall, G. *Logic: An Introduction.* Routledge, (2005).

[8] Smith, P. *An introduction to formal logic.* Cambridge University Press, Cambridge, UK; New York, (2003).

[9] Smullyan, R. M. *First-order logic.* Dover, New York, (1995).