Chapter 7
First-Order Logic: Formulas, Models, Tableaux

7.1 Relations and Predicates

The axioms and theorems of mathematics are defined on sets such as the set of integers $\mathscr{Z}$. We need to be able to write and manipulate logical formulas that contain relations on values from arbitrary sets. *First-order logic* is an extension of propositional logic that includes predicates interpreted as relations on a domain.

Before continuing, you may wish to review Appendix on set theory.

Example 7.1 $\mathscr{P}(x) \subset \mathscr{N}$ is the unary relation that is the subset of natural numbers that are prime: $\{2, 3, 5, 7, 11, \ldots\}$. ■

Example 7.2 $\mathscr{S}(x, y) \subset \mathscr{N}^2$ is the binary relation that is the subset of pairs (x, y) of natural numbers such that $y = x^2$: $\{(0, 0), (1, 1), (2, 4), (3, 9), \ldots\}$. ■

It would be more usual in mathematics to define a unary function $f(x) = x^2$ which maps a natural number x into its square. As shown in the example, functions are special cases of relations. For simplicity, we limit ourselves to relations in this chapter and the next; the extension of first-order logic to include functions is introduced in Sect. 9.1.

Definition 7.3 Let $\mathscr{R}$ be an n-ary relation on a domain D, that is, $\mathscr{R}$ is a subset of D^n. The relation $\mathscr{R}$ can be *represented* by the Boolean-valued function $P_{\mathscr{R}} : D^n \mapsto \{T, F\}$ that maps an n-tuple to T if and only if the n-tuple is an element of the relation:

$$P_{\mathscr{R}}(d_1, \ldots, d_n) = T \text{ iff } (d_1, \ldots, d_n) \in \mathscr{R},$$
$$P_{\mathscr{R}}(d_1, \ldots, d_n) = F \text{ iff } (d_1, \ldots, d_n) \notin \mathscr{R}.$$

■

Example 7.4 The set of primes $\mathscr{P}$ is represented by the function $P_{\mathscr{P}}$:

$$P_{\mathscr{P}}(0) = F,\ P_{\mathscr{P}}(1) = F,\ P_{\mathscr{P}}(2) = T,$$
$$P_{\mathscr{P}}(3) = T,\ P_{\mathscr{P}}(4) = F,\ P_{\mathscr{P}}(5) = T,$$
$$P_{\mathscr{P}}(6) = F,\ P_{\mathscr{P}}(7) = T,\ P_{\mathscr{P}}(8) = F,\ \ldots$$

■

DOI 10.1007/978-1-4471-4129-7_7, © Springer-Verlag London 2012

Example 7.5 The set of squares $\mathscr{S}$ is represented by the function $P_{\mathscr{S}}$:

$$P_{\mathscr{S}}(0,0) = P_{\mathscr{S}}(1,1) = P_{\mathscr{S}}(2,4) = P_{\mathscr{S}}(3,9) = \cdots = T,$$

$$\begin{aligned} &P_{\mathscr{S}}(0,1) = P_{\mathscr{S}}(1,0) = P_{\mathscr{S}}(0,2) = P_{\mathscr{S}}(2,0) = \\ &P_{\mathscr{S}}(1,2) = P_{\mathscr{S}}(2,1) = P_{\mathscr{S}}(0,3) = P_{\mathscr{S}}(2,2) = \cdots = F. \end{aligned}$$

■

This correspondence provides the link necessary for a logical formalization of mathematics. All the logical machinery—formulas, interpretations, proofs—that we developed for propositional logic can be applied to predicates. The presence of a domain upon which predicates are interpreted considerably complicates the technical details but not the basic concepts.

Here is an overview of our development of first-order logic:

- Syntax (Sect. 7.2): Predicates are used to represent functions from a domain to truth values. Quantifiers allow a purely syntactical expression of the statement that the relation represented by a predicate is true for *some* or *all* elements of the domain.
- Semantics (Sect. 7.3): An interpretation consists of a domain and an assignment of relations to the predicates. The semantics of the Boolean operators remains unchanged, but the evaluation of the truth value of the formula must take the quantifiers into account.
- Semantic tableaux (Sect. 7.5): The construction of a tableau is potentially infinite because a formula can be interpreted in an infinite domain. It follows that the method of semantic tableaux is not decision procedure for satisfiability in first-order logic. However, if the construction of a tableau for a formula A terminates in a closed tableau, then A is unsatisfiable (soundness); conversely, a systematic tableau for an unsatisfiable formula will close (completeness).
- Deduction (Sects. 8.1, 8.2): There are Gentzen and Hilbert deductive systems which are sound and complete. A valid formula is provable and we can construct a proof of the formula using tableaux, but given an *arbitrary* formula we cannot decide if it is valid and hence provable.
- Functions (Sect. 9.1): The syntax of first-order logic can be extended with function symbols that are interpreted as functions on the domain. With functions we can reason about mathematical operations, for example:

$$((x > 0 \wedge y > 0) \vee (x < 0 \wedge y < 0)) \rightarrow (x \cdot y > 0).$$

- Herbrand interpretations (Sect. 9.3): There are canonical interpretations called Herbrand interpretations. If a formula in *clausal form* has a model, it has a model which is an Herbrand interpretation, so to check satisfiability, it is sufficient to check if there is an Herbrand model for a formula.
- Resolution (Chap. 10): Resolution can be generalized to first-order logic with functions.

7.2 Formulas in First-Order Logic

7.2.1 Syntax

Definition 7.6 Let $\mathscr{P}$, $\mathscr{A}$ and $\mathscr{V}$ be countable sets of *predicate symbols*, *constant symbols* and *variables*. Each predicate symbol $p^n \in \mathscr{P}$ is associated with an *arity*, the number $n \geq 1$ of *arguments* that it takes. p^n is called an n-ary predicate. For $n = 1, 2$, the terms *unary* and *binary*, respectively, are also used. ■

Notation

- We will drop the word 'symbol' and use the words 'predicate' and 'constant' by themselves for the syntactical symbols.
- By convention, the following lower-case letters, possibly with subscripts, will denote these sets: $\mathscr{P} = \{p, q, r\}$, $\mathscr{A} = \{a, b, c\}$, $\mathscr{V} = \{x, y, z\}$.
- The superscript denoting the arity of the predicate will not be written since the arity can be inferred from the number of arguments.

Definition 7.7
$\forall$ is the *universal quantifier* and is read *for all.*
$\exists$ is the *existential quantifier* and is read *there exists.* ■

Definition 7.8 An *atomic formula* is an n-ary predicate followed by a list of n arguments in parentheses: $p(t_1, t_2, \ldots, t_n)$, where each argument t_i is either a variable or a constant. A *formula* in first-order logic is a tree defined recursively as follows:

- A formula is a leaf labeled by an atomic formula.
- A formula is a node labeled by $\neg$ with a single child that is a formula.
- A formula is a node labeled by $\forall x$ or $\exists x$ (for some variable x) with a single child that is a formula.
- A formula is a node labeled by a binary Boolean operator with two children both of which are formulas.

A formula of the form $\forall x A$ is a *universally quantified formula* or, simply, a *universal formula*. Similarly, a formula of the form $\exists x A$ is an *existentially quantified formula* or an *existential formula*. ■

The definition of derivation and formation trees, and the concept of induction on the structure of a formula are taken over unchanged from propositional logic. When writing a formula as a string, the quantifiers are considered to have the same precedence as negation and a higher precedence than the binary operators.

Example 7.9 Figure 7.1 shows the tree representation of the formula:

$$\forall x(\neg \exists y p(x, y) \vee \neg \exists y p(y, x)).$$

The parentheses in $p(x, y)$ are part of the syntax of the atomic formula. ■

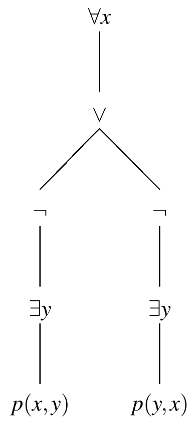

Fig. 7.1 Tree for $\forall x(\neg\exists y p(x,y)\vee\neg\exists y p(y,x))$

Example 7.10 Here are some examples of formulas in first-order logic:

$$\begin{aligned}
&\forall x\forall y(p(x,y)\rightarrow p(y,x)),\\
&\forall x\exists y p(x,y),\\
&\exists x\exists y(p(x)\wedge\neg p(y)),\\
&\forall x p(a,x),\\
&\forall x(p(x)\wedge q(x))\leftrightarrow(\forall x p(x)\wedge\forall x q(x)),\\
&\exists x(p(x)\vee q(x))\leftrightarrow(\exists x p(x)\vee\exists x q(x)),\\
&\forall x(p(x)\rightarrow q(x))\rightarrow(\forall x p(x)\rightarrow\forall x q(x)),\\
&(\forall x p(x)\rightarrow\forall x q(x))\rightarrow\forall x(p(x)\rightarrow q(x)).
\end{aligned}$$

For now, they are just given as examples of the syntax of formulas in first-order logic; their meaning will be discussed in Sect. 7.3.2. ■

7.2.2 The Scope of Variables

Definition 7.11 A universal or existential formula $\forall x A$ or $\exists x A$ is a *quantified formula*. x is the *quantified variable* and its *scope* is the formula A. It is not required that x actually appear in the scope of its quantification. ■

The concept of the scope of variables in formulas of first-order logic is similar to the concept of the scope of variables in block-structured programming languages. Consider the program in Fig. 7.2. The variable `x` is declared twice, once globally and once locally in method `p`. The scope of the global declaration includes `p`, but the local declaration *hides* the global one. Within `p`, the value printed will be `1`, the

Fig. 7.2 Global and local variables

```
class MyClass {
  int x;

  void p() {
    int x;
    x = 1;
    // Print the value of x
  }

  void q() {
    // Print the value of x
  }

  ... void main(...) {
  x = 5;
  p;
  q;
}
```

value of the local variable. Within the method `q`, the global variable `x` is in scope but not hidden and the value `5` will be printed. As in programming, hiding a quantified variable within its scope is confusing and should be avoided by giving different names to each quantified variable.

Definition 7.12 Let A be a formula. An occurrence of a variable x in A is a *free variable* of A iff x is not within the scope of a quantified variable x. A variable which is not free is *bound*.

If a formula has no free variables, it is *closed*. If $\{x_1, \ldots, x_n\}$ are all the free variables of A, the *universal closure* of A is $\forall x_1 \cdots \forall x_n A$ and the *existential closure* is $\exists x_1 \cdots \exists x_n A$.

$A(x_1, \ldots, x_n)$ indicates that the set of free variables of the formula A is a subset of $\{x_1, \ldots, x_n\}$. ■

Example 7.13 $p(x, y)$ has two free variables x and y, $\exists y p(x, y)$ has one free variable x and $\forall x \exists y p(x, y)$ is closed. The universal closure of $p(x, y)$ is $\forall x \forall y p(x, y)$ and its existential closure is $\exists x \exists y p(x, y)$. ■

Example 7.14 In $\forall x p(x) \wedge q(x)$, the occurrence of x in $p(x)$ is bound and the occurrence in $q(x)$ is free. The universal closure is $\forall x (\forall x p(x) \wedge q(x))$. Obviously, it would have been better to write the formula as $\forall x p(x) \wedge q(y)$ with y as the free variable; its universal closure is $\forall y (\forall x p(x) \wedge q(y))$. ■

7.2.3 *A Formal Grammar for Formulas* *

As with propositional logic (Sect. 2.1.6), formulas in first-order logic can be defined as the strings generated by a context-free grammar.

Definition 7.15 The following grammar defines *atomic formulas* and *formulas* in first-order logic:

argument	::=	x	for any $x \in \mathcal{V}$
argument	::=	a	for any $a \in \mathcal{A}$
argument_list	::=	*argument*	
argument_list	::=	*argument*, *argument_list*	
atomic_formula	::=	p (*argument_list*)	for any n-ary $p \in \mathcal{P}, n \geq 1$
formula	::=	*atomic_formula*	
formula	::=	$\neg$*formula*	
formula	::=	*formula* $\vee$ *formula*	similarly for $\wedge, \cdots$
formula	::=	$\forall\, x$ *formula*	for any $x \in \mathcal{V}$
formula	::=	$\exists\, x$ *formula*	for any $x \in \mathcal{V}$

An n-ary predicate p must have an argument list of length n. ■

7.3 Interpretations

In propositional logic, an interpretation is a mapping from atomic propositions to truth values. In first-order logic, the analogous concept is a mapping from atomic formulas to truth values. However, atomic formulas contain variables and constants that must be assigned elements of some domain; once that is done, the predicates are interpreted as relations over the domain.

Definition 7.16 Let A be a formula where $\{p_1, \ldots, p_m\}$ are all the predicates appearing in A and $\{a_1, \ldots, a_k\}$ are all the constants appearing in A. An *interpretation* $\mathcal{I}_A$ for A is a triple:

$$(D,\ \{R_1, \ldots, R_m\},\ \{d_1, \ldots, d_k\}),$$

where D is a *non-empty* set called the *domain*, R_i is an n_i-ary relation on D that is assigned to the n_i-ary predicate p_i and $d_i \in D$ is assigned to the constant a_i. ■

Example 7.17 Here are three interpretations for the formula $\forall x p(a, x)$:

$$\mathcal{I}_1 = (\mathcal{N}, \{\leq\}, \{0\}), \qquad \mathcal{I}_2 = (\mathcal{N}, \{\leq\}, \{1\}), \qquad \mathcal{I}_3 = (\mathcal{Z}, \{\leq\}, \{0\}).$$

The domain is either the $\mathscr{N}$, the set of natural numbers, or $\mathscr{Z}$, the set of integers. The binary relation $\leq$ (*less-than*) is assigned to the binary predicate p and either 0 or 1 is assigned to the constant a.

The formula can also be interpreted over strings:

$$\mathscr{I}_4 = (\mathscr{S}, \{substr\}, \{\text{" "}\}).$$

The domain $\mathscr{S}$ is a set of strings, *substr* is the binary relation such that $(s_1, s_2) \in$ *substr* iff s_1 is a substring of s_2, and " " is the null string. ■

A formula might have free variables and its truth value depends on the assignment of domain elements to the variables. For example, it doesn't make sense to ask if the formula $p(x,a)$ is true in the interpretation $(\mathscr{N}, \{>\}, \{10\})$. If x is assigned 15 the truth value of the formula is T, while if x is assigned 6 the truth value of the formula is F.

Definition 7.18 Let $\mathscr{I}_A$ be an interpretation for a formula A. An *assignment* $\sigma_{\mathscr{I}_A} : \mathscr{V} \mapsto D$ is a function which maps every free variable $v \in \mathscr{V}$ to an element $d \in D$, the domain of $\mathscr{I}_A$.

$\sigma_{\mathscr{I}_A}[x_i \leftarrow d_i]$ is an assignment that is the same as $\sigma_{\mathscr{I}_A}$ except that x_i is mapped to d_i. ■

We can now define the truth value of a formula of first-order logic.

Definition 7.19 Let A be a formula, $\mathscr{I}_A$ an interpretation and $\sigma_{\mathscr{I}_A}$ an assignment. $v_{\sigma_{\mathscr{I}_A}}(A)$, the *truth value* of A *under* $\mathscr{I}_A$ *and* $\sigma_{\mathscr{I}_A}$, is defined by induction on the structure of A (where we have simplified the notation by writing v_σ for $v_{\sigma_{\mathscr{I}_A}}$):

- Let $A = p_k(c_1, \ldots, c_n)$ be an atomic formula where each c_i is either a variable x_i or a constant a_i. $v_\sigma(A) = T$ iff $(d_1, \ldots, d_n) \in R_k$ where R_k is the relation assigned by $\mathscr{I}_A$ to p_k, and d_i is the domain element assigned to c_i, either by $\mathscr{I}_A$ if c_i is a constant or by $\sigma_{\mathscr{I}_A}$ if c_i is a variable.
- $v_\sigma(\neg A_1) = T$ iff $v_\sigma(A_1) = F$.
- $v_\sigma(A_1 \vee A_2) = T$ iff $v_\sigma(A_1) = T$ or $v_\sigma(A_2) = T$, and similarly for the other Boolean operators.
- $v_\sigma(\forall x A_1) = T$ iff $v_{\sigma[x \leftarrow d]}(A_1) = T$ for *all* $d \in D$.
- $v_\sigma(\exists x A_1) = T$ iff $v_{\sigma[x \leftarrow d]}(A_1) = T$ for *some* $d \in D$. ■

7.3.1 Closed Formulas

We define satisfiability and validity only on closed formulas. The reason is both convenience (not having to deal with assignments in addition to interpretations) and simplicity (because we can use the closures of formulas).

Theorem 7.20 *Let A be a* closed *formula and let $\mathscr{I}_A$ be an interpretation for A. Then $v_{\sigma_{\mathscr{I}_A}}(A)$ does not depend on $\sigma_{\mathscr{I}_A}$.*

Proof Call a formula independent of $\sigma_{\mathscr{I}_A}$ if its value does not depend on $\sigma_{\mathscr{I}_A}$. Let $A' = \forall x A_1(x)$ be a (not necessarily proper) subformula of A, where A' is *not* contained in the scope of any other quantifier. Then $v_{\sigma_{\mathscr{I}_A}}(A') = T$ iff $v_{\sigma_{\mathscr{I}_A}[x\leftarrow d]}(A_1)$ for all $d \in D$. But x is the only free variable in A_1, so A_1 is independent of $\sigma_{\mathscr{I}_A}$ since what is assigned to x is replaced by the assignment $[x \leftarrow d]$. A similar results holds for an existential formula $\exists x A_1(x)$.

The theorem can now be proved by induction on the depth of the quantifiers and by structural induction, using the fact that a formula constructed using Boolean operators on independent formulas is also independent. ■

By the theorem, if A is a closed formula we can use the notation $v_{\mathscr{I}}(A)$ without mentioning an assignment.

Example 7.21 Let us check the truth values of the formula $A = \forall x p(a, x)$ under the interpretations given in Example 7.17:

- $v_{\mathscr{I}_1}(A) = T$: For *all* $n \in \mathscr{N}$, $0 \le n$.
- $v_{\mathscr{I}_2}(A) = F$: It is not true that for *all* $n \in \mathscr{N}$, $1 \le n$. If $n = 0$ then $1 \not\le 0$.
- $v_{\mathscr{I}_3}(A) = F$: There is no smallest integer.
- $v_{\mathscr{I}_4}(A) = T$: By definition, the null string is a substring of every string.

The proof of the following theorem is left as an exercise.

Theorem 7.22 *Let $A' = A(x_1, \ldots, x_n)$ be a* (*non-closed*) *formula with free variables $x_1, \ldots, x_n$, and let $\mathscr{I}$ be an interpretation. Then*:

- $v_{\sigma_{\mathscr{I}_A}}(A') = T$ *for* some *assignment* $\sigma_{\mathscr{I}_A}$ *iff* $v_{\mathscr{I}}(\exists x_1 \cdots \exists x_n A') = T$.
- $v_{\sigma_{\mathscr{I}_A}}(A') = T$ *for* all *assignments* $\sigma_{\mathscr{I}_A}$ *iff* $v_{\mathscr{I}}(\forall x_1 \cdots \forall x_n A') = T$.

7.3.2 Validity and Satisfiability

Definition 7.23 Let A be a closed formula of first-order logic.

- A is *true* in $\mathscr{I}$ or $\mathscr{I}$ is a *model* for A iff $v_{\mathscr{I}}(A) = T$. Notation: $\mathscr{I} \models A$.
- A is *valid* if for *all* interpretations $\mathscr{I}$, $\mathscr{I} \models A$. Notation: $\models A$.
- A is *satisfiable* if for some interpretation $\mathscr{I}$, $\mathscr{I} \models A$.
- A is *unsatisfiable* if it is not satisfiable.
- A is *falsifiable* if it is not valid. ■

Example 7.24 The closed formula $\forall x p(x) \rightarrow p(a)$ is valid. If it were not, there would be an interpretation $\mathscr{I} = (D, \{R\}, \{d\})$ such that $v_{\mathscr{I}}(\forall x p(x)) = T$ and $v_{\mathscr{I}}(p(a)) = F$. By Theorem 7.22, $v_{\sigma_{\mathscr{I}}}(p(x)) = T$ for all assignments $\sigma_{\mathscr{I}}$, in particular for the assignment $\sigma'_{\mathscr{I}}$ that assigns d to x. But $p(a)$ is closed, so $v_{\sigma'_{\mathscr{I}}}(p(a)) = v_{\mathscr{I}}(p(a)) = F$, a contradiction. ■

Let us now analyze the semantics of the formulas in Example 7.10.

Example 7.25

- $\forall x \forall y (p(x, y) \rightarrow p(y, x))$
  The formula is satisfiable in an interpretation where p is assigned a symmetric relation like $=$. It is not valid because the formula is falsified in an interpretation that assigns to p a non-symmetric relation like $<$.
- $\forall x \exists y p(x, y)$
  The formula is satisfiable in an interpretation where p is assigned a relation that is a total function, for example, $(x, y) \in R$ iff $y = x + 1$ for $x, y \in \mathscr{Z}$. The formula is falsified if the domain is changed to the negative numbers because there is no negative number y such that $y = -1 + 1$.
- $\exists x \exists y (p(x) \wedge \neg p(y))$
  This formula is satisfiable only in a domain with at least two elements.
- $\forall x p(a, x)$
  This expresses the existence of an element with special properties. For example, if p is interpreted by the relation $\leq$ on the domain $\mathscr{N}$, then the formula is true for $a = 0$. If we change the domain to $\mathscr{Z}$ the formula is false for the same assignment of $\leq$ to p.
- $\forall x (p(x) \wedge q(x)) \leftrightarrow (\forall x p(x) \wedge \forall x q(x))$
  The formula is valid. We prove the forward direction and leave the converse as an exercise. Let $\mathscr{I} = (D, \{R_1, R_2\}, \{\})$ be an arbitrary interpretation. By Theorem 7.22, $v_{\sigma_{\mathscr{I}}}(p(x) \wedge q(x)) = T$ for all assignments $\sigma_{\mathscr{I}}$, and by the inductive definition of an interpretation, $v_{\sigma_{\mathscr{I}}}(p(x)) = T$ and $v_{\sigma_{\mathscr{I}}}(q(x)) = T$ for all assignments $\sigma_{\mathscr{I}}$. Again by Theorem 7.22, $v_{\mathscr{I}}(\forall x p(x)) = T$ and $v_{\mathscr{I}}(\forall x q(x)) = T$, and by the definition of an interpretation $v_{\mathscr{I}}(\forall x p(x) \wedge \forall x q(x)) = T$.
  Show that $\forall$ does not distribute over disjunction by constructing a falsifying interpretation for $\forall x (p(x) \vee q(x)) \leftrightarrow (\forall x p(x) \vee \forall x q(x))$.
- $\forall x (p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x))$
  We leave it as an exercise to show that this is a valid formula, but its converse $(\forall x p(x) \rightarrow \forall x q(x)) \rightarrow \forall x (p(x) \rightarrow q(x))$ is not. ■

7.3.3 An Interpretation for a Set of Formulas

In propositional logic, the concept of interpretation and the definition of properties such as satisfiability can be extended to sets of formulas (Sect. 2.2.4). The same holds for first-order logic.

Definition 7.26 Let $U = \{A_1, \ldots\}$ be a set of formulas where $\{p_1, \ldots, p_m\}$ are all the predicates appearing in all $A_i \in S$ and $\{a_1, \ldots, a_k\}$ are all the constants appearing in all $A_i \in S$. An *interpretation* $\mathscr{I}_U$ for S is a triple:

$$(D, \{R_1, \ldots, R_m\}, \{d_1, \ldots, d_k\}),$$

where D is a *non-empty* set called the *domain*, R_i is an n_i-ary relation on D that is assigned to the n_i-ary predicate p_i and $d_i \in D$ is an element of D that is assigned to the constant a_i. ■

Similarly, an assignment needs to assign elements of the domain to the free variables (if any) in all formulas in U. For simplicity, the following definition is given only for closed formulas.

Definition 7.27 A set of closed formulas $U = \{A_1, \ldots\}$ is *(simultaneously) satisfiable* iff there exists an interpretation $\mathscr{I}_U$ such that $v_{\mathscr{I}_{\mathscr{U}}}(A_i) = T$ for all i. The satisfying interpretation is a *model* of U. U is *valid* iff for every interpretation $\mathscr{I}_U$, $v_{\mathscr{I}_{\mathscr{U}}}(A_i) = T$ for all i. ■

The definitions of unsatisfiable and falsifiable are similar.

7.4 Logical Equivalence

Definition 7.28

- Let $U = \{A_1, A_2\}$ be a pair of closed formulas. A_1 is *logically equivalent* to A_2 iff $v_{\mathscr{I}_U}(A_1) = v_{\mathscr{I}_U}(A_2)$ for all interpretations $\mathscr{I}_U$. Notation: $A_1 \equiv A_2$.
- Let A be a closed formula and U a set of closed formulas. A is a *logical consequence* of U iff for all interpretations $\mathscr{I}_{U\cup\{A\}}$, $v_{\mathscr{I}_{U\cup\{A\}}}(A_i) = T$ for all $A_i \in U$ implies $v_{\mathscr{I}_{U\cup\{A\}}}(A) = T$. Notation: $U \models A$. ■

As in propositional logic, the metamathematical concept $A \equiv B$ is not the same as the formula $A \leftrightarrow B$ in the logic, and similarly for logical consequence and implication. The relations between the concepts is given by the following theorem whose proof is similar to the proofs of Theorems 2.29, 2.50.

Theorem 7.29 *Let A, B be closed formulas and $U = \{A_1, \ldots, A_n\}$ be a set of closed formulas. Then*:

$$\begin{aligned} A \equiv B \quad &\text{iff} \quad \models A \leftrightarrow B, \\ U \models A \quad &\text{iff} \quad \models (A_1 \wedge \cdots \wedge A_n) \rightarrow A. \end{aligned}$$

7.4.1 Logical Equivalences in First-Order Logic

Duality

The two quantifiers are duals:

$$\models \forall x A(x) \leftrightarrow \neg \exists x \neg A(x),$$
$$\models \exists x A(x) \leftrightarrow \neg \forall x \neg A(x).$$

In many presentations of first-order logic, $\forall$ is defined in the logic and $\exists$ is considered to be an abbreviation of $\neg\forall\neg$.

Commutativity and Distributivity

Quantifiers of the same type commute:

$$\models \forall x \forall y A(x,y) \leftrightarrow \forall y \forall x A(x,y),$$
$$\models \exists x \exists y A(x,y) \leftrightarrow \exists y \exists x A(x,y),$$

but $\forall$ and $\exists$ commute only in one direction:

$$\models \exists x \forall y A(x,y) \rightarrow \forall y \exists x A(x,y).$$

Universal quantifiers distribute over conjunction, and existential quantifiers distribute over disjunction:

$$\models \exists x (A(x) \vee B(x)) \leftrightarrow \exists x A(x) \vee \exists x B(x),$$
$$\models \forall x (A(x) \wedge B(x)) \leftrightarrow \forall x A(x) \wedge \forall x B(x),$$

but only one direction holds when distributing universal quantifiers over disjunction and existential quantifiers over conjunction:

$$\models \forall x A(x) \vee \forall x B(x) \rightarrow \forall x (A(x) \vee B(x)),$$
$$\models \exists x (A(x) \wedge B(x)) \rightarrow \exists x A(x) \wedge \exists x B(x).$$

To see that the converse direction of the second formula is falsifiable, let $D = \{d_1, d_2\}$ be a domain with two elements and consider an interpretation such that:

$$v(A(d_1)) = T, \qquad v(A(d_2)) = F, \qquad v(B(d_1)) = F, \qquad v(B(d_2)) = T.$$

Then $v(\exists x A(x) \wedge \exists x B(x)) = T$ but $v(\exists x (A(x) \wedge B(x))) = F$. A similar counterexample can be found for the first formula with the universal quantifiers and disjunction.

In the formulas with more than one quantifier, the scope rules ensure that each quantified variable is distinct. You may wish to write the formulas in the equivalent form with distinct variables names:

$$\models \forall x (A(x) \wedge B(x)) \leftrightarrow \forall y A(y) \wedge \forall z B(z).$$

Quantification Without the Free Variable in Its Scope

When quantifying over a disjunction or conjunction, if one subformula does not contain the quantified variable as a free variable, then distribution may be freely performed. If x is not free in B then:

$$\begin{array}{ll}
\models \exists x A(x) \vee B \leftrightarrow \exists x (A(x) \vee B), & \models \forall x A(x) \vee B \leftrightarrow \forall x (A(x) \vee B), \\
\models B \vee \exists x A(x) \leftrightarrow \exists x (B \vee A(x)), & \models B \vee \forall x A(x) \leftrightarrow \forall x (B \vee A(x)), \\
\models \exists x A(x) \wedge B \leftrightarrow \exists x (A(x) \wedge B), & \models \forall x A(x) \wedge B \leftrightarrow \forall x (A(x) \wedge B), \\
\models B \wedge \exists x A(x) \leftrightarrow \exists x (B \wedge A(x)), & \models B \wedge \forall x A(x) \leftrightarrow \forall x (B \wedge A(x)).
\end{array}$$

Quantification over Implication and Equivalence

Distributing a quantifier over an equivalence or an implication is not trivial.

As with the other operators, if the quantified variable does not appear in one of the subformulas there is no problem:

$$\begin{array}{l}
\models \forall x (A \rightarrow B(x)) \leftrightarrow (A \rightarrow \forall x B(x)), \\
\models \forall x (A(x) \rightarrow B) \leftrightarrow (\exists x A(x) \rightarrow B).
\end{array}$$

Distribution of universal quantification over equivalence works in one direction:

$$\models \forall x (A(x) \leftrightarrow B(x)) \rightarrow (\forall x A(x) \leftrightarrow \forall x B(x)),$$

while for existential quantification, we have the formula:

$$\models \forall x (A(x) \leftrightarrow B(x)) \rightarrow (\exists x A(x) \leftrightarrow \exists x B(x)).$$

For distribution over an implication, the following formulas hold:

$$\begin{array}{l}
\models \exists x (A(x) \rightarrow B(x)) \leftrightarrow (\forall x A(x) \rightarrow \exists x B(x)), \\
\models (\exists x A(x) \rightarrow \forall x B(x)) \rightarrow \forall x (A(x) \rightarrow B(x)), \\
\models \forall x (A(x) \rightarrow B(x)) \rightarrow (\exists x A(x) \rightarrow \exists x B(x)), \\
\models \forall x (A(x) \rightarrow B(x)) \rightarrow (\forall x A(x) \rightarrow \exists x B(x)).
\end{array}$$

To derive these formulas, replace the implication or equivalence by the equivalent disjunction and conjunction and use the previous equivalences.

Example 7.30

$$\begin{array}{rcl}
\exists x (A(x) \rightarrow B(x)) & \equiv & \exists x (\neg A(x) \vee B(x)) \\
& \equiv & \exists x \neg A(x) \vee \exists x B(x) \\
& \equiv & \neg \exists x \neg A(x) \rightarrow \exists x B(x) \\
& \equiv & \forall x A(x) \rightarrow \exists x B(x).
\end{array}$$

■

The formulas for conjunction and disjunction can be proved directly using the semantic definitions.

Example 7.31 Prove: $\models \forall x(A(x) \vee B(x)) \rightarrow \forall x A(x) \vee \exists x B(x)$.

Use logical equivalences of propositional logic (considering each atomic formula as an atomic proposition) to transform the formula:

$$\begin{array}{ll}\forall x(A(x) \vee B(x)) \rightarrow (\forall x A(x) \vee \exists x B(x)) & \equiv \\ \forall x(A(x) \vee B(x)) \rightarrow (\neg \forall x A(x) \rightarrow \exists x B(x)) & \equiv \\ \neg \forall x A(x) \rightarrow (\forall x(A(x) \vee B(x)) \rightarrow \exists x B(x)). & \end{array}$$

By duality of the quantifiers, we have:

$$\exists x \neg A(x) \rightarrow (\forall x(A(x) \vee B(x)) \rightarrow \exists x B(x))).$$

For the formula to be valid, it must be true under all interpretations. Clearly, if $v_{\mathscr{I}}(\exists x \neg A(x)) = F$ or $v_{\mathscr{I}}(\forall x(A(x) \vee B(x))) = F$, the formula is true, so we need only show $v_{\mathscr{I}}(\exists x B(x)) = T$ for interpretations $v_{\mathscr{I}}$ under which these subformulas are true. By Theorem 7.22, for some assignment $\sigma'_{\mathscr{I}}$, $v_{\sigma'_{\mathscr{I}}}(\neg A(x)) = T$ and thus $v_{\sigma'_{\mathscr{I}}}(A(x)) = F$. Using Theorem 7.22 again, $v_{\sigma_{\mathscr{I}}}(A(x) \vee B(x)) = T$ under all assignments, in particular under $\sigma'_{\mathscr{I}}$. By definition of an interpretation for disjunction, $v_{\sigma'_{\mathscr{I}}}(B(x)) = T$, and using Theorem 7.22 yet again, $v_{\mathscr{I}}(\exists x B(x)) = T$. ■

7.5 Semantic Tableaux

Before presenting the formal construction of semantic tableaux for first-order logic, we informally construct several tableaux in order to demonstrate the difficulties that must be dealt with and to motivate their solutions.

First, we need to clarify the concept of constant symbols. Recall from Definition 7.6 that formulas of first-order are constructed from countable sets of predicate, variable and constant symbols, although a particular formula such as $\exists x p(a, x)$ will only use a finite subset of these symbols. To build semantic tableaux in first-order logic, we will need to use the entire set of constant symbols $\mathscr{A} = \{a_0, a_1, \ldots\}$. If a formula like $\exists x p(a, x)$ contains a constant symbol, we assume that it is one of the a_i.

Definition 7.32 Let A be a quantified formula $\forall x A_1(x)$ or $\exists x A_1(x)$ and let a be a constant symbol. An *instantiation of A by a* is the formula $A_1(a)$, where all free occurrences of x are replaced by the constant a. ■

7.5.1 Examples for Semantic Tableaux

Instantiate Universal Formulas with all Constants

Example 7.33 Consider the valid formula:

$$A = \forall x (p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x)),$$

and let us build a semantic tableau for its negation. Applying the rule for the α-formula $\neg (A_1 \rightarrow A_2)$ twice, we get:

$$\begin{array}{c} \neg (\forall x (p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x))) \\ \downarrow \\ \forall x (p(x) \rightarrow q(x)),\ \neg (\forall x p(x) \rightarrow \forall x q(x)) \\ \downarrow \\ \forall x (p(x) \rightarrow q(x)),\ \forall x p(x),\ \neg \forall x q(x) \\ \downarrow \\ \forall x (p(x) \rightarrow q(x)),\ \forall x p(x),\ \exists \neg x q(x) \end{array}$$

where the last node is obtained by the duality of $\forall$ and $\exists$.

The third formula will be true in an interpretation only if there exists a domain element c such that $c \in R_q$, where R_q is the relation assigned to the predicate q. Let us use the first constant a_1 to represent this element and instantiate the formula with it:

$$\begin{array}{c} \forall x (p(x) \rightarrow q(x)),\ \forall x p(x),\ \exists \neg x q(x) \\ \downarrow \\ \forall x (p(x) \rightarrow q(x)),\ \forall x p(x),\ \neg q(a_1). \end{array}$$

The first two formulas are universally quantified, so they can be true only if they hold for *every* element of the domain of an interpretation. Since any interpretation must include the domain element that is assigned to the constant a_1, we instantiate the universally quantified formulas with this constant:

$$\begin{array}{c} \forall x (p(x) \rightarrow q(x)),\ \forall x p(x),\ \neg q(a_1) \\ \downarrow \\ \forall x (p(x) \rightarrow q(x)),\ p(a_1),\ \neg q(a_1) \\ \downarrow \\ p(a_1) \rightarrow q(a_1),\ p(a_1),\ \neg q(a_1). \end{array}$$

Applying the rule to the β-formula $p(a_1) \rightarrow q(a_1)$ immediately gives a closed tableau, which to be expected for the negation of the valid formula A. ■

From this example we learn that existentially quantified formulas must be instantiated with a constant the represents the domain element that must exist. Once a constant is introduced, instantiations of all universally quantified formulas must be done for that constant.

$$\begin{tikzcd}
 & \neg\,(\forall x(p(x)\vee q(x))\rightarrow(\forall xp(x)\vee\forall xq(x))) \arrow[d] & \\
 & \forall x(p(x)\vee q(x)),\ \neg\,(\forall xp(x)\vee\forall xq(x)) \arrow[d] & \\
 & \forall x(p(x)\vee q(x)),\ \neg\forall xp(x),\ \neg\forall xq(x) \arrow[d] & \\
 & \forall x(p(x)\vee q(x)),\ \exists\neg xp(x),\ \exists\neg xq(x) \arrow[d] & \\
 & \forall x(p(x)\vee q(x)),\ \exists\neg xp(x),\ \neg q(a_1) \arrow[d] & \\
 & \forall x(p(x)\vee q(x)),\ \neg\,p(a_1),\ \neg q(a_1) \arrow[d] & \\
 & p(a_1)\vee q(a_1),\ \neg\,p(a_1),\ \neg q(a_1) \arrow[dl] \arrow[dr] & \\
p(a_1),\ \neg\,p(a_1),\ \neg q(a_1) & & q(a_1),\ \neg\,p(a_1),\ \neg q(a_1) \\
\times & & \times
\end{tikzcd}$$

Fig. 7.3 Semantic tableau for the negation of a satisfiable, but not valid, formula

Don’t Use the Same Constant Twice to Instantiate Existential Formulas

Example 7.34 Figure 7.3 shows an attempt to construct a tableau for the negation of the formula:

$$A=\forall x(p(x)\vee q(x))\rightarrow(\forall xp(x)\vee\forall xq(x)),$$

which is satisfiable but not valid. As a falsifiable formula, its negation $\neg\,A$ is satisfiable, but the tableau in the figure is closed. What went wrong?

The answer is that instantiation of $\exists x\neg\,p(x))$ should not have used the constant a_1 once it had already been chosen for the instantiation of $\exists\neg\,xq(x)$. Choosing the same constant means that the interpretation will assign the same domain element to both occurrences of the constant. In fact, the formula A true (and $\neg\,A$ is false) in all interpretations over domains of a single element, but the formula might be satisfiable in interpretations with larger domains.

To avoid unnecessary constraints on the domain of a possible interpretation, a *new* constant must be chosen for every instantiation of an existentially quantified formula:

$$\begin{tikzcd}
\forall x(p(x)\vee q(x)),\ \exists\neg\,xp(x),\ \exists\neg\,xq(x) \arrow[d] \\
\forall x(p(x)\vee q(x)),\ \exists\neg\,xp(x),\ \neg q(a_1) \arrow[d] \\
\forall x(p(x)\vee q(x)),\ \neg\,p(a_2),\ \neg q(a_1).
\end{tikzcd}$$

Instantiating the universally quantified formula with a_1 gives:

$$p(a_1)\vee q(a_1),\ \neg\,p(a_2),\ \neg q(a_1).$$

■

Don't Use Up Universal Formulas

Example 7.35 Continuing the tableau from the previous example:

$$\begin{array}{c}\forall x(p(x) \vee q(x)),\ \neg p(a_2),\ \neg q(a_1)\\ \downarrow\\ p(a_1) \vee q(a_1),\ \neg p(a_2),\ \neg q(a_1)\end{array}$$

we should now instantiate the universal formula $\forall x(p(x) \vee q(x))$ *again* with a_2, since it must be true for *all* domain elements, but, unfortunately, the formula has been used up by the tableau construction. To prevent this, universal formulas will never be deleted from the label of a node. They remain in the labels of all descendant nodes so as to constrain the possible interpretations of every new constant that is introduced:

$$\begin{array}{c}\forall x(p(x) \vee q(x)),\ \neg p(a_2),\ \neg q(a_1)\\ \downarrow\\ \forall x(p(x) \vee q(x)),\ p(a_1) \vee q(a_1),\ \neg p(a_2),\ \neg q(a_1)\\ \downarrow\\ \forall x(p(x) \vee q(x)),\ p(a_2) \vee q(a_2),\ p(a_1) \vee q(a_1),\ \neg p(a_2),\ \neg q(a_1).\end{array}$$

We leave it to the reader to continue the construction the tableau using the rule for β-formulas. Exactly one branch of the tableau will be open. A model can be defined by specifying a domain with two elements, say, 1 and 2. These elements are assigned to the constants a_1 and a_2, respectively, and the relations R_p and R_q assigned to p and q, respectively, hold for exactly one of the domain elements:

$$\mathscr{I} = (\{1, 2\}, \{R_p = \{1\},\ R_q = \{2\}\}, \{a_1 = 1, a_2 = 2\}).$$

As expected, this model satisfies $\neg A$, so A is falsifiable. ■

A Branch May not Terminate

Example 7.36 Let us construct a semantic tableau to see if the formula $A = \forall x \exists y p(x, y)$ is satisfiable. Apparently, no rules apply since the formula is universally quantified and we only required that they had to be instantiated for constants already appearing in the formulas labeling a node. The constants are those that appear in the original formula and those that were introduced by instantiating existentially quantified formulas.

However, recall from Definition 7.16 that an interpretation is required to have a *non-empty* domain; therefore, we can arbitrarily choose the constant a_1 to represent that element. The tableau construction begins by instantiating A and then instantiating the existential formula with a new constant:

$$\begin{array}{c}\forall x \exists y p(x, y)\\ \downarrow\\ \forall x \exists y p(x, y),\ \exists y p(a_1, y)\\ \downarrow\\ \forall x \exists y p(x, y),\ p(a_1, a_2).\end{array}$$

Since $A = \forall x \exists y p(x, y)$ is universally quantified, it is not used up.

The new constant a_2 is used to instantiate the universal formula A again; this results in an existential formula which must be instantiated with a new constant a_3:

$$\begin{array}{c}\forall x \exists y p(x, y),\ p(a_1, a_2)\\ \downarrow\\ \forall x \exists y p(x, y),\ \exists y p(a_2, y),\ p(a_1, a_2)\\ \downarrow\\ \forall x \exists y p(x, y),\ p(a_2, a_3),\ p(a_1, a_2).\end{array}$$

The construction of this semantic tableau will not terminate and an infinite branch results. It is easy to see that there are models for A with infinite domains, for example, $(\mathcal{N}, \{<\}, \{\})$. ■

The method of semantic tableaux is *not* a decision procedure for satisfiability in first-order logic, because we can never know if a branch that does not close defines an infinite model or if it will eventually close, say, after one million further applications of the tableau rules.

Example 7.36 is not very satisfactory because the formula $\forall x \exists y p(x, y)$ is satisfiable in a finite model, in fact, even in a model whose domain contains a single element. We were being on the safe side in always choosing new constants to instantiate existentially quantified formulas. Nevertheless, it is easy to find formulas that have no finite models, for example:

$$\forall x \exists y p(x, y) \ \wedge\ \forall x \neg\, p(x, x) \ \wedge\ \forall x \forall y \forall z (p(x, y) \wedge p(y, z) \rightarrow p(x, z)).$$

Check that $(\mathcal{N}, \{<\}, \{\})$ is an infinite model for this formula; we leave it as an exercise to show that the formula has no finite models.

An Open Branch with Universal Formulas May Terminate

Example 7.37 The first two steps of the tableau for $\{\forall x p(a, x)\}$ are:

$$\begin{array}{c}\{\forall x p(a, x)\}\\ \downarrow\\ \{p(a, a), \forall x p(a, x)\}\\ \downarrow\\ \{p(a, a), \forall x p(a, x)\}.\end{array}$$

There is no point in creating the same node again and again, so we specify that this branch is finite and open. Clearly, $(\{a\}, \{P = (a, a)\}, \{a\})$ is a model for the formula. ■

$$\begin{array}{c}
\forall x \exists y p(x, y) \wedge \forall x (q(x) \wedge \neg q(x)) \\
\downarrow \\
\forall x \exists y p(x, y),\ \forall x (q(x) \wedge \neg q(x)) \\
\downarrow \\
\forall x \exists y p(x, y),\ \exists y p(a_1, y),\ \forall x (q(x) \wedge \neg q(x)) \\
\downarrow \\
\forall x \exists y p(x, y),\ p(a_1, a_2),\ \forall x (q(x) \wedge \neg q(x)) \\
\downarrow \\
\forall x \exists y p(x, y),\ \exists y p(a_2, y),\ p(a_1, a_2),\ \forall x (q(x) \wedge \neg q(x)) \\
\downarrow \\
\forall x \exists y p(x, y),\ p(a_2, a_3),\ p(a_1, a_2),\ \forall x (q(x) \wedge \neg q(x))
\end{array}$$

Fig. 7.4 A tableau that should close, but doesn't

The Tableau Construction Must Be Systematic

Example 7.38 The tableau in Fig. 7.4 is for the formula which is the conjunction of $\forall x \exists y p(x, y)$, which we already know to be satisfiable, together with the formula $\forall x (q(x) \wedge \neg q(x))$, which is clearly unsatisfiable. However, the branch can be continued indefinitely, because we are, in effect, choosing to apply rules only to subformulas of $\forall x \exists y p(x, y)$, as we did in Example 7.36. This branch will never close although the formula is unsatisfiable. A *systematic* construction is needed to make sure that rules are eventually applied to all the formulas labeling a node. ■

7.5.2 The Algorithm for Semantic Tableaux

The following definition extends a familiar concept from propositional logic:

Definition 7.39 A *literal* is a closed atomic formula $p(a_1, \ldots, a_k)$, an atomic formula all of whose arguments are constants, or the negation of a closed atomic formula $\neg p(a_1, \ldots, a_k)$. If A is $p(a_1, \ldots, a_k)$ then $A^c = \neg p(a_1, \ldots, a_k)$, while if A is $\neg p(a_1, \ldots, a_k)$ then $A^c = p(a_1, \ldots, a_k)$. ■

The classification of formulas in propositional logic as α and β formulas (Sect. 2.6.2) is retained and we extend the classification to formulas with quantifiers. γ-formulas are universally quantified formulas $\forall x A(x)$ and the negations of existentially quantified formulas $\neg \exists x A(x)$, while δ-formulas are existentially quantified formulas $\exists x A(x)$ and the negations of universally quantified formulas $\neg \forall x A(x)$. The rules for these formulas are simply instantiation with a constant:

γ	$\gamma(a)$
$\forall x A(x)$	$A(a)$
$\neg \exists x A(x)$	$\neg A(a)$

δ	$\delta(a)$
$\exists x A(x)$	$A(a)$
$\neg \forall x A(x)$	$\neg A(a)$

The algorithm for the construction of a semantic tableau in first-order logic is similar to that for propositional logic with the addition of rules for quantified formulas, together with various constraints designed to avoid the problems were saw in the examples.

Algorithm 7.40 (Construction of a semantic tableau)
Input: A formula ϕ of first-order logic.
Output: A semantic tableau $\mathscr{T}$ for ϕ: each branch may be infinite, finite and marked open, or finite and marked closed.

A semantic tableau is a tree $\mathscr{T}$ where each node is labeled by a pair $W(n) = (U(n), C(n))$, where:

$$U(n) = \{A_{n_1}, \ldots, A_{n_k}\}$$

is a set of formulas and:

$$C(n) = \{c_{n_1}, \ldots, c_{n_m}\}$$

is a set of constants. $C(n)$ contains the list of constants that appear in the formulas in $U(n)$. Of course, the sets $C(n)$ could be created on-the-fly from $U(n)$, but the algorithm in easier to understand if they explicitly label the nodes.

Initially, $\mathscr{T}$ consists of a single node n_0, the root, labeled with

$$(\{\phi\}, \{a_{0_1}, \ldots, a_{0_k}\}),$$

where $\{a_{0_1}, \ldots, a_{0_k}\}$ is the set of constants that appear in ϕ. If ϕ has no constants, take the first constant a_0 in the set $\mathscr{A}$ and label the node with $(\{\phi\}, \{a_0\})$.

The tableau is built inductively by repeatedly *choosing* an unmarked leaf l labeled with $W(l) = (U(l), C(l))$, and applying the *first applicable rule* in the following list:

- If $U(l)$ contains a complementary pair of literals, mark the leaf *closed* $\times$.
- If $U(l)$ is not a set of literals, *choose* a formula A in $U(l)$ that is an α-, β- or δ-formula.
  - If A is an α-formula, create a new node l' as a child of l. Label l' with:

    $$W(l') = ((U(l) - \{A\}) \cup \{\alpha_1, \alpha_2\},\ C(l)).$$

    (In the case that A is $\neg\neg A_1$, there is no α_2.)
  - If A is a β-formula, create two new nodes l' and l'' as children of l. Label l' and l'' with:

    $$\begin{aligned} W(l') &= ((U(l) - \{A\}) \cup \{\beta_1\},\ C(l)), \\ W(l'') &= ((U(l) - \{A\}) \cup \{\beta_2\},\ C(l)). \end{aligned}$$

  - If A is a δ-formula, create a new node l' as a child of l and label l' with:

    $$W(l') = ((U(l) - \{A\}) \cup \{\delta(a')\},\ C(l) \cup \{a'\}),$$

    where a' is some constant that *does not appear in* $U(l)$.

- Let $\{\gamma_{l_1}, \ldots, \gamma_{l_m}\} \subseteq U(l)$ be all the γ-formulas in $U(l)$ and let $C(l) = \{c_{l_1}, \ldots, c_{l_k}\}$. Create a new node l' as a child of l and label l' with

$$W(l') = \left(U(l) \cup \left\{ \bigcup_{i=1}^{m} \bigcup_{j=1}^{k} \gamma_{l_i}(c_{l_j}) \right\},\ C(l) \right).$$

  However, if $U(l)$ consists only of literals and γ-formulas and if $U(l')$ as constructed would be the same as $U(l)$, do not create node l'; instead, mark the leaf l as *open* $\odot$. ∎

Compare the algorithm with the examples in Sect. 7.5.1. The phrase *first applicable rule* ensures that the construction is systematic. For δ-formulas, we added the condition that a *new* constant be used in the instantiation. For γ-formulas, the formula to which the rule is applied is not removed from the set $U(l)$ when $W(l')$ is created. The sentence beginning *however* in the rule for γ-formulas is intended to take care of the case where no new formulas are produced by the application of the rule.

Definition 7.41 A branch in a tableau is *closed* iff it terminates in a leaf marked closed; otherwise (it is infinite or it terminates in a leaf marked open), the branch is *open*.

A tableau is *closed* if all of its branches are closed; otherwise (it has a finite or infinite open branch), the tableau is open. ∎

Algorithm 7.40 is *not* a search procedure for a satisfying interpretation, because it may choose to infinitely expand one branch. Semantic tableaux in first-order logic can only be used to prove the validity of a formula by showing that a tableau for its negation closes. Since all branches close in a closed tableau, the nondeterminism in the application of the rules (choosing a leaf and choosing an α-, β- or γ-formula) doesn't matter.

7.6 Soundness and Completion of Semantic Tableaux

7.6.1 Soundness

The proof of the soundness of the algorithm for constructing semantic tableaux in first-order logic is a straightforward generalization of the one for propositional logic (Sect. 2.7.2).

Theorem 7.42 (Soundness) *Let ϕ be a formula in first-order logic and let $\mathscr{T}$ be a tableau for ϕ. If $\mathscr{T}$ closes, then ϕ is unsatisfiable.*

Proof The theorem is a special case of the following statement: if a subtree rooted at a node n of $\mathscr{T}$ closes, the set of formulas $U(n)$ is unsatisfiable.

The proof is by induction on the height h of n. The proofs of the base case for $h = 0$ and the inductive cases 1 and 2 for α- and β-rules are the same as in propositional logic (Sect. 2.6).

Case 3: The γ-rule was used. Then:

$$U(n) = U_0 \cup \{\forall x A(x)\} \quad \text{and} \quad U(n') = U_0 \cup \{\forall x A(x),\ A(a)\},$$

for some set of formulas U_0, where we have simplified the notation and explicitly considered only one formula.

The inductive hypothesis is that $U(n')$ is unsatisfiable and we want to prove that $U(n)$ is also unsatisfiable. Assume to the contrary that $U(n)$ is satisfiable and let $\mathscr{I}$ be a model for $U(n)$. Then $v_{\mathscr{I}}(A_i) = T$ for all $A_i \in U_0$ and also $v_{\mathscr{I}}(\forall x A(x)) = T$. But $U(n') = U(n) \cup \{A(a)\}$, so if we can show that $v_{\mathscr{I}}(A(a)) = T$, this will contradict the inductive hypothesis that $U(n')$ is unsatisfiable.

Now $v_{\mathscr{I}}(\forall x A(x)) = T$ iff $v_{\sigma_{\mathscr{I}}}(A(x)) = T$ for all assignments $\sigma_{\mathscr{I}}$, in particular for any assignment that assigns the same domain element to x that $\mathscr{I}$ does to a, so $v_{\mathscr{I}}(A(a)) = T$. By the tableau construction, $a \in C(n)$ and it appears in some formula of $U(n)$; therefore, $\mathscr{I}$, a model of $U(n)$, does, in fact, assign a domain element to a.

Case 4: The δ-rule was used. Then:

$$U(n) = U_0 \cup \{\exists x A(x)\} \quad \text{and} \quad U(n') = U_0 \cup \{A(a)\},$$

for some set of formulas U_0 and for some constant a that does not occur in any formula of $U(n)$.

The inductive hypothesis is that $U(n')$ is unsatisfiable and we want to prove that $U(n)$ is also unsatisfiable. Assume to the contrary that $U(n)$ is satisfiable and let:

$$\mathscr{I} = (D, \{R_1, \ldots, R_n\}, \{d_1, \ldots, d_k\})$$

be a model for $U(n)$.

Now $v_{\mathscr{I}}(\exists x A(x)) = T$ iff $v_{\sigma_{\mathscr{I}}}(A(x)) = T$ for some assignment $\sigma_{\mathscr{I}}$, that is, $\sigma_{\mathscr{I}}(x) = d$ for some $d \in D$. Extend $\mathscr{I}$ to the interpretation:

$$\mathscr{I}' = (D, \{R_1, \ldots, R_n\}, \{d_1, \ldots, d_k, d\})$$

by assigning d to the constant a. $\mathscr{I}'$ is well-defined: since a does not occur in $U(n)$, it is not among the constants $\{a_1, \ldots, a_k\}$ already assigned $\{d_1, \ldots, d_k\}$ in $\mathscr{I}$. Since $v_{\mathscr{I}'}(U_0) = v_{\mathscr{I}}(U_0) = T$, $v_{\mathscr{I}'}(A(a)) = T$ contradicts the inductive hypothesis that $U(n')$ is unsatisfiable. ■

7.6.2 Completeness

To prove the completeness of the algorithm for semantic tableaux we define a Hintikka set, show that a (possibly infinite) branch in a tableau is a Hintikka set and then prove Hintikka's Lemma that a Hintikka set can be extended to a model. We begin with a technical lemma whose proof is left as an exercise.

Lemma 7.43 *Let b be an open branch of a semantic tableau, n a node on b, and A a formula in $U(n)$. Then some rule is applied to A at node n or at a node m that is a descendant of n on b. Furthermore, if A is a γ-formula and $a \in C(n)$, then $\gamma(a) \in U(m')$, where m' is the child node created from m by applying a rule.*

Definition 7.44 Let U be a set of closed formulas in first-order logic. U is a *Hintikka set* iff the following conditions hold for all formulas $A \in U$:

1. If A is a literal, then either $A \notin U$ or $A^c \notin U$.
2. If A is an α-formula, then $\alpha_1 \in U$ and $\alpha_2 \in U$.
3. If A is a β-formula, then $\beta_1 \in U$ or $\beta_2 \in U$.
4. If A is a γ-formula, then $\gamma(c) \in U$ for *all* constants c in formulas in U.
5. If A is a δ-formula, then $\delta(c) \in U$ for *some* constant c. ■

Theorem 7.45 *Let b be a (finite or infinite) open branch of a semantic tableau and let $U = \bigcup_{n \in b} U(n)$. Then U is a Hintikka set.*

Proof Let $A \in U$. We show that the conditions for a Hintikka set hold.

Suppose that A is a literal. By the construction of the tableau, once a literal appears in a branch, it is never deleted. Therefore, if A appears in a node n and A^c appears in a node m which is a descendant of n, then A must also appear in m. By assumption, b is open, so either $A \notin U$ or $A^c \notin U$ and condition 1 holds.

If A is not atomic and not a γ-formula, by Lemma 7.43 eventually a rule is applied to A, and conditions 2, 3 and 5 hold.

Let A be a γ-formula that first appears in $U(n)$, let c be a constant that first appears in $C(m)$ and let $k = \max(n, m)$. By the construction of the tableau, the set of γ-formulas and the set of constants are non-decreasing along a branch, so $A \in U(k)$ and $c \in C(k)$. By Lemma 7.43, $\gamma(c) \in U(k') \subseteq U$, for some $k' > k$. ■

Theorem 7.46 (Hintikka's Lemma) *Let U be a Hintikka set. Then there is a (finite or infinite) model for U.*

Proof Let $\mathscr{C} = \{c_1, c_2, \ldots\}$ be the set of constants in formulas of U. Define an interpretation $\mathscr{I}$ as follows. The domain is the same set of symbols $\{c_1, c_2, \ldots\}$. Assign to each constant c_i in U the symbol c_i in the domain. For each n-ary predicate p_i in U, define an n-ary relation R_i by:

$$\begin{array}{lll} (a_{i_1}, \ldots, a_{i_n}) \in R_i & \text{if} & p(a_{i_1}, \ldots, a_{i_n}) \in U, \\ (a_{i_1}, \ldots, a_{i_n}) \notin R_i & \text{if} & \neg\, p(a_{i_1}, \ldots, a_{i_n}) \in U, \\ (a_{i_1}, \ldots, a_{i_n}) \in R_i & \text{otherwise.} & \end{array}$$

The relations are well-defined by condition 1 in the definition of Hintikka sets. We leave as an exercise to show that $\mathscr{I} \models A$ for all $A \in U$ by induction on the structure of A using the conditions defining a Hintikka set. ■

Theorem 7.47 (Completeness) *Let A be a valid formula. Then the semantic tableau for $\neg A$ closes.*

Proof Let A be a valid formula and suppose that the semantic tableau for $\neg A$ does not close. By Definition 7.41, the tableau must contain a (finite or infinite) open branch b. By Theorem 7.45, $U = \bigcup_{n \in b} U(n)$ is a Hintikka set and by Theorem 7.46, there is a model $\mathscr{I}$ for U. But $\neg A \in U$ so $\mathscr{I} \models \neg A$ contradicting the assumption that A is valid. ■

7.7 Summary

First-order logic adds variables and constants to propositional logic, together with the quantifiers $\forall$ (for all) and $\exists$ (there exists). An interpretation includes a domain; the predicates are interpreted as relations over elements of the domain, while constants are interpreted as domain elements and variables in non-closed formulas are assigned domain elements.

The method of semantic tableaux is sound and complete for showing that a formula is unsatisfiable, but it is not a decision procedure for satisfiability, since branches of a tableau may be infinite. When a tableau is constructed, a universal quantifier followed by an existential quantifier can result in an infinite branch: the existential formula is instantiated with a new constant and then the instantiation of the universal formula results in a new occurrence of the existentially quantified formula, and so on indefinitely. There are formulas that are satisfiable only in an infinite domain.

7.8 Further Reading

The presentation of semantic tableaux follows that of Smullyan (1968) although he uses analytic tableaux. Advanced textbooks that also use tableaux are Nerode and Shore (1997) and Fitting (1996).

7.9 Exercises

7.1 Find an interpretation which falsifies $\exists x p(x) \rightarrow p(a)$.

7.2 Prove the statements left as exercises in Example 7.25:

- $\forall x p(x) \wedge \forall x q(x) \rightarrow \forall x(p(x) \wedge q(x))$ is valid.
- $\forall x(p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x))$ is a valid formula, but its converse $(\forall x p(x) \rightarrow \forall x q(x)) \rightarrow \forall x(p(x) \rightarrow q(x))$ is not.

7.3 Prove that the following formulas are valid:

$$\begin{aligned}
&\exists x(A(x) \rightarrow B(x)) \leftrightarrow (\forall x A(x) \rightarrow \exists x B(x)),\\
&(\exists x A(x) \rightarrow \forall x B(x)) \rightarrow \forall x(A(x) \rightarrow B(x)),\\
&\forall x(A(x) \vee B(x)) \rightarrow (\forall x A(x) \vee \exists x B(x)),\\
&\forall x(A(x) \rightarrow B(x)) \rightarrow (\exists x A(x) \rightarrow \exists x B(x)).
\end{aligned}$$

7.4 For each formula in the previous exercise that is an implication, prove that the converse is not valid by giving a falsifying interpretation.

7.5 For each of the following formulas, either prove that it is valid or give a falsifying interpretation.

$$\begin{aligned}
&\exists x \forall y((p(x,y) \wedge \neg p(y,x)) \rightarrow (p(x,x) \leftrightarrow p(y,y))),\\
&\forall x \forall y \forall z(p(x,x) \wedge (p(x,z) \rightarrow (p(x,y) \vee p(y,z)))) \rightarrow \exists y \forall z p(y,z).
\end{aligned}$$

7.6 Suppose that we allowed the domain of an interpretation to be *empty*. What would this mean for the equivalence:

$$\forall y p(y,y) \vee \exists x q(x,x) \equiv \exists x(\forall y p(y,y) \vee q(x,x)).$$

7.7 Prove Theorem 7.22 on the relationship between a non-closed formula and its closure.

7.8 Complete the semantic tableau construction for the negation of

$$\forall x(p(x) \vee q(x)) \rightarrow (\forall x p(x) \vee \forall x q(x)).$$

7.9 Prove that the formula $(\forall x p(x) \rightarrow \forall x q(x)) \rightarrow \forall x(p(x) \rightarrow q(x))$ is not valid by constructing a semantic tableau for its negation.

7.10 Prove that the following formula has no finite models:

$$\forall x \exists y p(x,y) \;\wedge\; \forall x \neg p(x,x) \;\wedge\; \forall x \forall y \forall z(p(x,y) \wedge p(y,z) \rightarrow p(x,z)).$$

7.11 Prove Lemma 7.43, the technical lemma used in the proof of the completeness of the method of semantic tableaux.

7.12 Complete the proof of Lemma 7.46 that every Hintikka set has a model.

References

M. Fitting. *First-Order Logic and Automated Theorem Proving* (*Second Edition*). Springer, 1996.
A. Nerode and R.A. Shore. *Logic for Applications* (*Second Edition*). Springer, 1997.
R.M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968. Reprinted by Dover, 1995.