# APPENDIX B

# First-Order Logic

## B.1 Introduction

First-order logic is a formal system with three fundamental constituents.

1. A *language* $\mathcal{L}$ defines the set of possible statements, usually called (well-formed) *formulas*, e.g., $p \rightarrow q$ (read $p$ implies $q$).
2. The *semantics* is a way to assign a meaning to each statement in $\mathcal{L}$ and to determine whether a formula is true. For instance, if both $p$ and $q$ are true, then the statement $p \rightarrow q$ is true.
3. A *proof theory* provides rules to transform statements of $\mathcal{L}$ and derive new statements, e.g., the modus ponens rule that from $p$ and $p \rightarrow q$ derives $q$. We write $p, p \rightarrow q \vdash q$.

In this appendix, we first introduce propositional logic and then extend its expressiveness to first-order logic.

## B.2 Propositional Logic

The language $\mathcal{L}$ of propositional logic is the set $P$ of propositions. $P$ is defined inductively starting from an enumerable set of atomic propositions $P_0$.

1. If $p \in P_0$, then $p \in P$.
2. If $p \in P$, then $\neg p \in P$.
3. If $p \in P$ and $q \in P$, then $p \wedge q \in P$.
4. Nothing else is a propositional formula.

We define $p \vee q$ as $\neg(\neg p \wedge \neg q)$, and $p \rightarrow q$ as $\neg p \vee q$. The symbols $\wedge$ (and), $\vee$ (or), and $\rightarrow$ (if ... then, or implies) are called *connectives*. Given a propositional formula, its atomic propositions are called *propositional variables*.

**Example B.1** Let $p$, $q$, and $r$ be atomic propositions. Examples of propositional formulas are $p$, $\neg p$, $p \wedge q$, and $(p \rightarrow q) \vee \neg r$. The propositional variables of $(p \rightarrow q) \vee \neg r$ are $p$, $q$, and $r$. ■

If $\mathcal{L}$ is a propositional language, then the usual way to assign a meaning to each formula in $\mathcal{L}$ is to assign a truth value (either true or false) to the propositional variables and to take into account the connectives. Thus $\neg p$ evaluates to true if and only if $p$ is false and $p \wedge q$ evaluates to true if and only if both $p$ and $r$ are true. We call an assignment of truth values to the propositional variables of a proposition an *interpretation.* A *model* of a proposition is an interpretation for which the formula evaluates to true. A *satisfiability problem* is the problem of determining whether a formula has a model. We say that a formula is *satisfiable* if there exists a model of the formula. A formula is *valid* if and only if any possible interpretation is a model. In other words, a formula is valid if and only if it evaluates to true for any interpretation. If a formula $p$ is valid, then we write $\models p$. Valid propositional formulas are called *tautologies.*

**Example B.2** We have the following models of $(p \rightarrow q) \vee \neg r$: all the interpretations that assign *false* to $r$, all the interpretations that assign *false* to $p$, and all the interpretations that assign *true* to $q$. Examples of tautologies are $p \vee \neg p$, $p \rightarrow p$, and $(p \rightarrow (q \rightarrow p))$. ■

A widely used proof theory is *resolution.* It is based on two main ideas. The first idea is to work on formulas with a standard syntactic structure (called *normal form*). The second idea is to apply just a unique and powerful inference rule (called *resolution*). A propositional formula is in *Conjunctive Normal Form* (CNF) if and only if it is a conjunction of clauses, where a *clause* is a disjunction of literals. A *literal* is a propositional variable (*positive literal*) or its negation (*negative literal*). Intuitively, two literals are complementary if and only if one is the negation of the other. For instance, $p$ and $\neg p$ are two complementary literals. A *unit clause* is a clause with one literal. A CNF formula can thus be represented as a set of clauses, where each clause is represented as a set of literals. The *resolution inference rule* is the following.

> *For any two clauses $C_1$ and $C_2$, if there is a literal $L_1$ in $C_1$ that is complementary to a literal $L_2$ in $C_2$, then delete $L_1$ and $L_2$ from $C_1$ and $C_2$, respectively, and construct the disjunction of the remaining clauses. The constructed clause is called a* resolvent *of $C_1$ and $C_2$.*

The reasoning underlying the resolution principle is that, if we have $A$ in $C_1$ and $\neg A$ in $C_2$, then we can have two cases.

1. $A$ is true, and therefore $C_1$ is true and can be eliminated, while $\neg A$ can be eliminated from $C_2$.

2. $A$ is false, and therefore $C_2$ is true and can be eliminated, while $A$ can be eliminated from $C_1$.

In the limiting case, in which $C_1$ and $C_2$ are unit clauses and the two literals are complementary, the resolution principle generates the *empty clause* $\emptyset$.

If $C$ is a resolvent of $C_1$ and $C_2$, we say that $C$ *logically follows* from $C_1$ and $C_2$. We can now define the notion of deduction based on resolution. Given a set of clauses $S$, a *deduction* of the clause $C$ from $S$ is a finite sequence $C_1, \ldots, C_n$ of clauses such that each $C_i$ either is a clause in $S$ or is a resolvent of clauses preceeding $C_i$, and $C_n = C$. A deduction of the empty clause $\emptyset$ from $S$ is called a *refutation* of $S$.

**Example B.3** Let us build a deduction of $p \rightarrow r$ from $p \rightarrow q$ and $q \rightarrow r$, i.e., let us show that $p \rightarrow r$ logically follows from $p \rightarrow q$ and $q \rightarrow r$. First, we transform in CNF. We have $S = \{\neg p \vee q, \neg q \vee r\}$ and $C = \neg p \vee r$. We have a one-step deduction: $C_1 = \{\neg p \vee r\}$. Now let us suppose we want to prove $p \rightarrow p$. The CNF version of $p \rightarrow p$ is $\{\neg p \vee p\}$. We take its negation: $S = \{p \wedge \neg p\}$, which resolves in one step to the empty clause. ■

It can be shown that resolution is correct and complete, i.e., if a clause $C$ logically follows from a set of clauses $S$, then $C$ is valid if any clause in $S$ is valid, and vice versa.

## B.3 First-Order Logic

The nice thing about propositional logic is its simplicity. Its weakness is its lack of expressivity. Many ideas cannot be treated in this simple way, such as the fact that any object that has a given property also has another property, or the fact that there exists an object that satisfies some property. A first-order language $\mathcal{L}$ is based on four types of symbols: constant, variable, function, and predicate symbols. A *term* is defined recursively as follows.

1. A constant is a term.
2. A variable is a term.
3. If $t_1, \ldots t_n$ are terms and $f$ is an $n$-place function symbol, then $f(t_1, \ldots, t_n)$ is a term.
4. Nothing else is a term.

We define *atoms* (or atomic formulas) as the basic building blocks for constructing formulas in first-order logic. If $t_1, \ldots, t_n$ are terms and $P$ is an $n$-place predicate, then $P(t_1, \ldots, t_n)$ is an atom. We can now define *formulas* in first-order logic.

1. An atom is a formula.
2. If $\phi$ and $\psi$ are formulas, then $\neg\phi$ and $\phi \wedge \psi$ are formulas.
3. If $\phi$ is a formula and $x$ is a variable, then $\forall x \phi$ is a formula.
4. Nothing else is a formula.

We define $\exists x\ \phi$ as $\neg\forall x\ \neg\phi$. An occurrence of a variable in a formula is *bound* if and only if the occurrence is within the scope of a quantifier employng the variable or is the occurrence in that quantifier. An occurrence of a variable is *free* if and only if this occurrence of the variable is not bound. A variable is free in a formula if at least one occurrence of it is free in the formula, and similarly for bound variables. Formulas without free variables are called *closed formulas.*

If $\mathcal{L}$ is a first-order language, then the usual way to assign a meaning to each statement in $\mathcal{L}$ is to define an *interpretation* that maps each constant symbol, function symbol, and predicate symbol of $\mathcal{L}$ into an object, a function, or a relation in some world $W$, where a *world* is a set of objects.[1]

A very common interpretation is the *Herbrand interpretation,* in which the objects of $W$ are the constant symbols of $\mathcal{L}$, and the interpretation maps each constant symbol into itself. If $\mathcal{L}$ is function-free, then a state $s$ can be viewed as a Herbrand interpretation that assigns *true* to all ground atoms in $s$, and *false* to all ground atoms not in $s$. From this, a truth value can be computed for every closed formula of $\mathcal{L}$ (i.e., every formula of $\mathcal{L}$ that contains no free variables) using the usual rules for logical composition. For example, a conjunction $\phi_1 \wedge \phi_2$ is true in $s$ (written $s \models \phi_1 \wedge \phi_2$) if and only if both $\phi_1$ and $\phi_2$ are true in $s$. Similarly, a quantified formula $\forall x\ \phi$ is true in $s$ if and only if for every substitution $\theta$ that substitutes a constant symbol for $x$, $\theta(\phi)$ is true in $s$. We will use the notation $s \models \phi$ to mean that $\phi$ is true in $s$. Also, if $\theta = \{x \leftarrow c\}$ is a substitution and $\phi$ is a formula, then we will use the notation $\phi[x \leftarrow c]$ to refer to $\theta(\phi)$. Thus, we have that $s \models \forall x\ \phi$ if and only if $s \models \phi[x \leftarrow c]$ for every constant symbol $c$ of $\mathcal{L}$.

The notion of literal, clause, and CNF is easily extended to first-order logic. The resolution principle is then extended to first-order logic. Recall the resolution principle for propositional logic: a resolvent is obtained by eliminating two complementary literals. The notion of complementary literals in propositional logic is very simple. For clauses that contain variables, this operation is more complicated. For example, consider these clauses:

$$C_1 : P(x) \vee Q(x)$$
$$C_2 : \neg P(f(x)) \vee R(x)$$

There are no complementary literals in $C_1$ and $C_2$. However, if we substitute $f(c)$ (where $f$ is a function symbol and $c$ is a constant) for $x$ in $C_1$ and $c$ for $x$ in $C_2$,

1. The world of an interpretation is often called the *domain* of the interpretation.

we obtain:

$$C_1' : P(f(c)) \vee Q(f(c))$$
$$C_2' : \neg P(f(c)) \vee R(c)$$

Now we have two complementary literals, and we can obtain the resolvent:

$$C_3' : Q(f(c)) \vee R(c)$$

More generally, if we substitute $f(x)$ for $x$ in $C_1$, we obtain a resolvent for $C_1$ and $C_2$:

$$C_3 : Q(f(x)) \vee R(x)$$

By substituting appropriate terms for the variables in $C_1$ and $C_2$ as shown here, we can resolve them. Furthermore, clause $C_3$ is the "most general clause" in the sense that all the other clauses that can be generated by this process are instances of $C_3$. In general, a substitution $\theta$ is a finite set of the form $\theta = \{t_1 \leftarrow x_1, \ldots, t_n \leftarrow x_n\}$, where every $x_i$ is a variable, every $t_i$ is a term different from $x_i$, and $x_i \neq x_j$ for any $i \neq j$. When $t_1, \ldots, t_n$ are ground terms (i.e., terms containing no variables), then the substitution is called *ground substitution.* Let $\theta$ be a substitution and $e$ be an expression of first-order logic, i.e., a term or a formula. Then $\theta(e)$ is an expression obtained from $e$ by replacing simultaneously each occurrence of the variable $x_i$ with the term $t_i$. $\theta(e)$ is called an *instance* of $e$.

If we want to apply the resolution principle to a set of clauses, then in order to identify a pair of complementary literals, we have to match or *unify* two or more expressions. That is, we have to find a substitution that can make several expressions identical. A substitution $\theta$ is called a *unifier* for a set of expressions $\{e_1, \ldots, e_k\}$ if and only if $\theta(e_1) = \theta(e_2) = \ldots = \theta(e_k)$. The set $\{e_1, \ldots, e_k\}$ is said to be *unifiable* if and only if there exists a unifier for it.

A unifier $\sigma$ for a set $\{e_1, \ldots, e_k\}$ of expressions is a *most general unifier* if and only if for each unifier $\theta$ for the set there is a substitution $\rho$ such that $\theta = \sigma \circ \rho$, where $\circ$ is the usual composition of substitutions.

It is possible to define an algorithm for finding a most general unifier for a finite unifiable set of nonempty expressions. When the set is not unifiable, the algorithm returns failure.

We can now define the resolution principle for first-order logic. We start with some definitions. If two literals of a clause $C$ have a most general unifier $\sigma$, then $\sigma(C)$ is a *factor* of $C$. Let $C_1$ and $C_2$ be two clauses with no variables in common. Let $L_1$ and $L_2$ be two literals in $C_1$ and $C_2$, respectively. If $L_1$ and $\neg L_2$ have a most general unifier $\sigma$, then the clause

$$(\sigma(C_1) - \sigma(L_1)) \cup (\sigma(C_2) - \sigma(L_2))$$

is a *binary resolvent* of $C_1$ and $C_2$.

We can finally define the notion of a resolvent. A *resolvent* of the clauses $C_1$ and $C_2$ is one of the following binary resolvents:

1. A binary resolvent of $C_1$ and $C_2$
2. A binary resolvent of $C_1$ and a factor of $C_2$
3. A binary resolvent of a factor of $C_1$ and $C_2$
4. A binary resolvent of a factor of $C_1$ and a factor of $C_2$

It is possible to show that the resolution principle is *complete*, i.e., it is guaranteed to generate the empty clause $\emptyset$ from an unsatisfiable set of clauses.