General References

Lloyd (1987) and Maier and Warren (1988) are good general references.

H. Abramson and M. H. Rogers, eds., *Meta-Programming for Logic Programming*, MIT Press, Cambridge, Mass., 1989.

J. A. Campbell, ed., *Implementations of PROLOG*, Ellis Horwood, Chichester, UK, 1984.

K. L. Clark, "Logic Programming Schemes," in *Proceedings of the International Conference on Fifth Generation Computer Systems 88*, North-Holland, Amsterdam, The Netherlands, 1988.

H. Gallaire and J. Minker, eds., *Logic and Databases*, Plenum Press, New York, 1978.

C. J. Hogger, *Introduction to Logic Programming*, Academic Press, Inc., New York, 1984.

C. J. Hogger, *Essentials of Logic Programming*, Oxford University Press, Oxford, UK, 1990.

IEEE Symposia on Logic Programming, IEEE Computer Society Press.

Journal of Logic Programming, published by Elsevier Science Publishing Co., New York.

R. A. Kowalski, "Predicate Logic as a Programming Language," in *Proceedings of IFIP-74*, North-Holland, Amsterdam, The Netherlands, 1974, pp. 569–574.

R. A. Kowalski, *Logic for Problem Solving*, Elsevier, Amsterdam, The Netherlands, 1979.

U. Nilsson and J. Maluszynski, *Logic, Programming and Prolog*, John Wiley & Sons, Inc., New York, 1990.

Proceedings of the International Conferences on Logic Programming, MIT Press, Cambridge, Mass.

Proceedings of the North Atlantic Conferences on Logic Programming, MIT Press, Cambridge, Mass.

L. Sterling and E. Y. Shapiro, *The Art of Prolog*, MIT Press, Cambridge, Mass., 1986.

C. J. HOGGER
R. A. KOWALSKI
University of London

LOGIC, PROPOSITIONAL

Propositional logic is the study of inferences that can be made from propositions. Roughly, propositions are the "meanings" or "thoughts" expressed by declarative sentences (Church, 1956; Gale, 1967). Secondarily, it is also the study of the representation of information by propositions. Other names for it are propositional calculus, sentential logic, and—when its subject matter is taken to be those things that can have truth values (ie, that are either true or false)—it is often called truth-functional logic. The bearers of truth values are sometimes considered to be propositions, (declarative) sentences, or truth functions. Typically, propositional logics are distinguished from first-order logics by their lack of internal analysis of propositions (eg, they do not distinguish between subject and predicate (see LOGIC, PREDICATE). Such logics also exist for other types of sentences (such as imperatives) and for more than two truth values. For details, see the logics of imperatives presented by Castañeda (1975) and Rescher (1966), the many-valued logics discussed by Rescher (1969), and the various *Proceedings of the International Symposium on Multiple-Valued Logic*.

The representational system of propositional logic is its underlying language. This consists of propositions and propositional (or, in the case of truth-functional propositional logic, truth-functional) connectives; a syntax that specifies the grammar of propositions; and a semantics that provides the "meanings" of propositions in terms of their truth conditions. The deductive system of propositional logic consists of rules that only permit inferences that lead from truths to truths (thereby preventing inferences that would lead from truths to falsehoods). Its (deductive) syntax consists of such rules and axioms, and its semantics characterizes these rules in terms of truth values. (This entry is concerned primarily with truth-functional propositional logic, except where indicated.)

LANGUAGE OF PROPOSITIONAL LOGIC

There are two kinds of propositions: atomic and molecular (also called simple and compound). Molecular propositions are formed from (one or more) atomic ones by means of truth-functional connectives (or truth-functional operators). For instance,

Andrea is a philosopher

is atomic;

Andrea is a philosopher and Mike is not French

is molecular, formed from the two atomic propositions

Andrea is a philosopher
Mike is French

by the connective *and* and the connective (or operator) *not*. But

Ruth believes that Marvin is a logician

is not molecular (rather, it is atomic), since the operator *Ruth believes that* is not truth-functional. [A branch of logic that does treat the latter proposition as molecular is modal logic, in particular, doxastic modal logic (see LOGIC, MODAL; BELIEF REPRESENTATION SYSTEMS).] In propositional logic, atomic propositions are considered to be unanalyzable; the branch of nonmodal logic that analyzes atomic propositions is called predicate logic (see LOGIC, PREDICATE).

In truth-functional propositional logic a molecular proposition must be such that its truth value is a function of the truth values of its atomic parts. The particular function is determined by the connectives. In addition to the one-place "connective" or operator, negation (usually expressed in English by *not* or *it is not the case that*), there are 16 two-place truth-functional connectives. Of these, the most common are given in Table 1. For a list of others, see Church (1956) and Schagrin, Rapaport, and Dipert (1985). There are, of course, other n-place connectives (for $n \neq 2$), for example, the three-place connective *if . . . then . . . else* (see Manna, 1974); but these are either trivial (when $n = 1$) or (for $n > 2$) expressible in terms of two-place connectives, as discussed below (Minimal Sets of

Table 1. Common Two-Place Truth-Functional Connectives

Conjunction	and
Inclusive disjunction	or
Material conditional	if . . . then . . .
Material biconditional	if and only if
Exclusive disjunction	xor; ie, either . . . or . . . but not both
Joint denial	nand
Disjoint (or alternative) denial	nor

Connectives). (For interesting generalizations of these connectives as operators on sets of propositions, see Shapiro (1979) for a discussion in an AI context and McCawley (1981) for a discussion in a linguistic context.)

Syntax

A formal syntax for a language of propositional logic can be presented by a recursive definition of well-formed proposition. One such definition, using the most common connectives, follows.

1. The letters $A, \ldots, Z$ and these letters subscripted with positive-integer numerals (eg, A_1, B_{27}) are well-formed (atomic) propositions.
2. If **P** and **Q** are well-formed propositions, then so are
   - $\neg$ **P** (the negation of **P**)
   - (**P** $\wedge$ **Q**) (the conjunction of **P** with **Q**)
   - (**P** $\vee$ **Q**) (the inclusive disjunction of **P** with **Q**)
   - (**P** $\rightarrow$ **Q**) (the material conditional whose antecedent is **P** and whose consequent is **Q**)
   - (**P** $\leftrightarrow$ **Q**) (the material biconditional of **P** with **Q**)
   - (**P** + **Q**) (the exclusive disjunction of **P** with **Q**)
   - (**P**|**Q**) (the joint denial of **P** with **Q**)
   - (**P** $\downarrow$ **Q**) (the disjoint denial of **P** with **Q**)
3. Nothing else is a well-formed proposition.

The boldface letters in the second clause of this recursive definition are metavariables ranging over propositions. The outer parentheses in clause 2 prevent ambiguity. For instance, **P** $\wedge$ **Q** $\vee$ **R** is ill-formed; instead, one must write either ((**P** $\wedge$ **Q**) $\vee$ **R**) or (**P** $\wedge$ (**Q** $\vee$ **R**)), depending on which is wanted. On occasion, parentheses can be dropped for purposes of readability. Other systems can be defined by using different symbols or by using different conventions for disambiguation—such as precedence of connectives.

There are several ways to express the connectives in English, many of which have non-truth-functional connotations; however, propositional logic studies only the truth-functional properties of such phrases. Thus, propositional logic ignores the important distinctions between

Marie is a vice-president *and* Ben is a clerk

and

Marie is a vice-president *but* Ben is a clerk

(the latter suggesting, perhaps, that Ben is *merely* a clerk) as well as the important distinctions between

I got into bed and I fell asleep

and

I fell asleep and I got into bed

(the latter suggesting, perhaps, that I sleepwalk).

There are also several other families of symbols used for the connectives, most notably Polish (or prefix) notation. Polish notation has the advantage of not requiring parentheses for disambiguation. The first five connectives of clause 2 expressed in Polish notation are

*N***P**
*K***PQ**
*A***PQ**
*C***PQ**
*E***PQ**

As an example, the ambiguous proposition in infix notation discussed earlier cannot be written in Polish notation. Instead, one is forced to write either *AK***PQR** or *KA***PQR**. (It is standard practice in Polish notation to use the letters *N*, *K*, *A*, *C*, and *E* instead of the connective symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$, respectively. For a thorough discussion of these notational issues, consult a standard introductory textbook, such as Copi (1979), Schagrin, Rapaport, and Dipert (1985), and especially Schagrin (1979).

It is also important to recall that propositional logic does not provide an analysis of the internal structure of propositions; thus, it does not provide a way to represent or reason about individuals or classes; that is the province of first-order (or predicate) logic.

Semantics

The semantics of molecular propositions can be given by means of the equivalences in Table 2.

The semantics can also be given by means of truth tables. Typically, these have two sets of columns, one for the atomic propositions (the "input") and one for the molecular proposition (the "output"); and 2^n rows, where n is

Table 2. Semantics of Molecular Propositions

$\neg$ **P** is true	if and only if	**P** is false.
(**P** $\wedge$ **Q**) is true	if and only if	both of **P** and **Q** are true.
(**P** $\vee$ **Q**) is false	if and only if	both of **P** and **Q** are false.
(**P** $\rightarrow$ **Q**) is true	if and only if	**P** is false or **Q** is true (or both).
(**P** $\leftrightarrow$ **Q**) is true	if and only if	**P** and **Q** have the same truth value.
(**P** + **Q**) is true	if and only if	**P** and **Q** have opposite truth values.
(**P**|**Q**) is false	if and only if	both of **P** and **Q** are true.
(**P** $\downarrow$ **Q**) is true	if and only if	both of **P** and **Q** are false.

Table 3. Sample Truth Tables

Input	Output	Input		Output	Input		Output	Input		Output
P	¬ P	P	Q	(P ∧ Q)	P	Q	(P ∨ Q)	P	Q	(P → Q)
T	F	T	T	T	T	T	T	T	T	T
F	T	T	F	F	T	F	T	T	F	F
		F	T	F	F	T	T	F	T	T
		F	F	F	F	F	F	F	F	T

Table 4. Truth Table with Intermediate Computations

Input		Computations			Output
P	Q	(P ∨ Q)	(P ∧ Q)	¬ (P ∧ Q)	((P ∨ Q) ∧ ¬ (P ∧ Q))
T	T	T	T	F	F
T	F	T	F	T	T
F	T	T	F	T	T
F	F	F	F	T	F

Table 5. Some Important Logical Equivalences

Double negation	P	is logically equivalent to	¬ ¬ P
Idempotency	P	is logically equivalent to	(P ∧ P)
	P	is logically equivalent to	(P ∨ P)
Commutative laws	(P ∧ Q)	is logically equivalent to	(Q ∧ P)
	(P ∨ Q)	is logically equivalent to	(Q ∨ P)
Associative laws	(P ∧ (Q ∧ R))	is logically equivalent to	((P ∧ Q) ∧ R)
	(P ∨ (Q ∨ R))	is logically equivalent to	((P ∨ Q) ∨ R)
Distributive laws	(P ∧ (Q ∨ R))	is logically equivalent to	((P ∧ Q) ∨ (P ∧ R))
	(P ∨ (Q ∧ R))	is logically equivalent to	((P ∨ Q) ∧ (P ∨ R))
De Morgan's laws	¬ (P ∧ Q)	is logically equivalent to	(¬ P ∨ ¬ Q)
	¬ (P ∨ Q)	is logically equivalent to	(¬ P ∧ ¬ Q)
Contraposition	(P → Q)	is logically equivalent to	(¬ Q → ¬ P)
Material conditional	(P → Q)	is logically equivalent to	(¬ P ∨ Q)
	(P → Q)	is logically equivalent to	¬ (P ∧ ¬ Q)
Exportation	(P → (Q → R))	is logically equivalent to	((P ∧ Q) → R)

the number of distinct atomic propositions, one for each possible combination of truth values of the atomic propositions. Samples are given in Table 3 (T and F stand for true and false, respectively).

Truth tables can also be used to compute the truth values of more complicated molecular propositions. Sometimes this is done using a third set of columns for intermediate computations of "subpropositions," as in Table 4. (Algorithms for computing with truth tables are given in Schagrin, Rapaport, and Dipert, 1985).

Two propositions are logically equivalent if they have the same truth values for all possible combinations of truth values of their atomic parts. Table 5 lists some of the important logical equivalences.

Minimal Sets of Connectives. The choice of which connectives to use depends on one's purposes. Generally, if the language of propositional logic is to be used in a representational system, especially one for natural language, then a large set of connectives is appropriate. This permits distinguishing between distinct but logically equivalent propositions. However, for deductive purposes, a smaller number of connectives is better, both because fewer inference rules are then needed and because metatheoretic proofs about propositional logic then become easier.

It can be shown that all n-place truth-functional connectives can be expressed using only negation and conjunction, or else negation and disjunction, or else negation and the material conditional. They can also all be expressed using only one connective, either joint denial or disjoint denial. (For further discussion and proofs, see, eg, Copi, 1979 and Mendelson, 1979.) Usually, a compromise is found between the extremes of using all of the connectives (for representational adequacy) and only one or two (for elegance or metatheoretic simplicity): It is common to use negation, disjunction, and conjunction to express a proposition in either conjunctive normal form (CNF) or disjunctive normal form (DNF). In the former a proposition is expressed as a (logically equivalent) conjunction of disjunctions of atomic propositions and negations; in the latter, as a (logically equivalent) disjunction of conjunctions of atomic propositions and negations. For example, the proposition

$$(((\mathbf{P} \rightarrow \mathbf{Q}) \vee \mathbf{Q}) \rightarrow (\mathbf{R} \wedge \mathbf{Q}))$$

is logically equivalent to the CNF proposition

$$(\neg \mathbf{P} \vee \neg \mathbf{Q} \vee \mathbf{R}) \wedge (\mathbf{P} \vee \neg \mathbf{Q} \vee \mathbf{R}) \wedge (\mathbf{P} \vee \mathbf{Q} \vee \neg \mathbf{R}) \wedge (\mathbf{P} \vee \mathbf{Q} \vee \mathbf{R})$$

as well as to the DNF proposition

$$(\neg \mathbf{P} \wedge \mathbf{Q} \wedge \mathbf{R}) \vee (\mathbf{P} \wedge \neg \mathbf{Q} \wedge \neg \mathbf{R}) \vee (\mathbf{P} \wedge \neg \mathbf{Q} \wedge \mathbf{R}) \vee (\mathbf{P} \wedge \mathbf{Q} \wedge \mathbf{R})$$

Algorithms for converting a proposition into a logically equivalent proposition in CNF or DNF may be found in Copi (1979) and Schagrin, Rapaport, and Dipert (1985).

Tautologies, Contradictions, and Contingent Propositions. Propositions that are true for all possible combinations of truth values of their atomic parts are called tautologies; those that are false for all possible combinations are called contradictions; and the others are said to be contingent. For example, $((\mathbf{P} \wedge (\mathbf{P} \rightarrow \mathbf{Q})) \rightarrow \mathbf{Q})$ is a tautology; $(\mathbf{P} \wedge \neg \mathbf{P})$ is a contradiction; and $(\mathbf{P} \rightarrow \mathbf{Q})$ is contingent. Because of the semantics of negation, the negation of a tautology is a contradiction, and vice versa. All tautologies are logically equivalent to each other, as are all contradictions. This fact is of some significance for representational issues since all tautologies clearly do not "say" the same thing. For example, $(\mathbf{P} \vee \neg \mathbf{P})$ and $((\mathbf{P} \wedge (\mathbf{P} \rightarrow \mathbf{Q})) \rightarrow \mathbf{Q})$ are both tautologies and hence logically equivalent; yet, in an important sense, they do not "mean" the same thing.

The Paradox of the Material Conditional. There are other limitations on the use of the language of propositional logic as a representational system for natural-language sentences. Most notably, the semantics of the material conditional do not correspond to the ordinary English use of if–then. For instance, $((\mathbf{P} \wedge \neg \mathbf{P}) \rightarrow \mathbf{Q})$ is a tautology simply because its antecedent is a contradiction and, hence, false. But a corresponding English sentence such as "If $1 + 1 = 2$ and $1 + 1 \neq 2$, then Bertrand Russell is the Pope" does not seem to be true even though it is a tautology. For this reason, such phenomena are called "paradoxes of the material conditional." Attempts to overcome these "paradoxes" have generally taken the form of introducing new, non-truth-functional operators and connectives whose semantics are closer to their natural-language counterparts. The two main kinds of logic that have been developed along these lines are modal logic and relevance logic. (For the former, see LOGIC, MODAL, and Hughes and Cresswell, 1968; for the latter, see Anderson and Belnap, 1975.)

DEDUCTIVE SYSTEMS OF PROPOSITIONAL LOGIC

Syntax

A deductive system for any logic can be presented in one of two ways: as an axiomatic system or by means of a natural deduction system.

Axiomatic Propositional Logic. An axiomatic system typically has several axioms (which ought to be tautologies) and a minimal number of rules of inference (which ought to lead from truths to truths). To present propositional logic axiomatically, the well-formed propositions (WFPs) are restricted here to those whose only connectives are $\neg$ and $\rightarrow$. All WFPs of the following three tautological forms, called axiom schemata, may be taken as axioms (others are possible; note, again, that boldface letters are metavariables ranging over propositions):

(A1) $(\mathbf{P} \rightarrow (\mathbf{Q} \rightarrow \mathbf{P}))$ (confirmation of the consequent)

(A2) $((\mathbf{P} \rightarrow (\mathbf{Q} \rightarrow \mathbf{R})) \rightarrow ((\mathbf{P} \rightarrow \mathbf{Q}) \rightarrow (\mathbf{P} \rightarrow \mathbf{R})))$ (self-distribution)

(A3) $((\neg \mathbf{P} \rightarrow \neg \mathbf{Q}) \rightarrow (\mathbf{Q} \rightarrow \mathbf{P}))$ (contraposition)

There is one rule of inference:

(MP) From $\mathbf{P}$ and $(\mathbf{P} \rightarrow \mathbf{Q})$, infer $\mathbf{Q}$ (modus ponens)

A *proof* of a WFP $\mathbf{P}_n$ is a sequence $\mathbf{P}_1, \ldots, \mathbf{P}_n$ of WFPs such that for each $k (1 \leq k \leq n)$, either $\mathbf{P}_k$ is an axiom or there are $i, j < k$ such that $\mathbf{P}_i = (\mathbf{P}_j \rightarrow \mathbf{P}_k)$. (Note: $\mathbf{P}_i$ is not merely logically equivalent to $(\mathbf{P}_j \rightarrow \mathbf{P}_k)$; it *is* $(\mathbf{P}_j \rightarrow \mathbf{P}_k)$.) A *theorem* of our propositional logic is a WFP $\mathbf{P}$ such that there is a proof of $\mathbf{P}$ (viz, $\mathbf{P}_1, \ldots, \mathbf{P}_{n-1}, \mathbf{P}$). Finally, $\mathbf{P}$ is *provable* means: $\mathbf{P}$ is a theorem—the notation for this is: $\vdash \mathbf{P}$. (Sometimes, the turnstile, $\vdash$, is subscripted by the name of the system of logic of which $\mathbf{P}$ is a theorem.)

As an example, a proof of $(\mathbf{P} \rightarrow \mathbf{P})$ is given in Table 6. Comments, preceded by semicolons, are not formally part of the proof. Note, however, that they would be formally part of a proof that the proposition is a theorem of propositional logic, that is, of a proof that $\vdash(\mathbf{P} \rightarrow \mathbf{P})$.

The notion of "proof" can be extended to "proof from hypotheses," where the hypotheses are nonlogical principles (or *postulates*) typically belonging to some particular subject matter (eg, laws of physics or "world knowledge"). They would not usually be tautologies. Formally, a sequence of WFPs $\mathbf{P}_1, \ldots, \mathbf{P}_n$ is a *proof of* $\mathbf{P}_n$ *from a set of hypotheses* $\mathbf{H}$ iff for all k $(1 \leq k \leq n)$, either $\mathbf{P}_k$ is an axiom, or $\mathbf{P}_k \in \mathbf{H}$, or $\mathbf{P}_k$ is inferred by (MP) from previous WFPs in the sequence. The notation for this is: $\mathbf{H} \vdash \mathbf{P}_n$; if $\mathbf{H} = \{\mathbf{H}_1, \ldots, \mathbf{H}_m\}$, then the notation is $\mathbf{H}_1, \ldots, \mathbf{H}_m \vdash \mathbf{P}_n$ [for complete details of an axiomatic propositional logic, see Mendelson (1979) and Kleene (1950)].

A Natural Deduction System for Propositional Logic. A natural deduction system typically has no axioms but has several rules of inference, and it allows for the possibility of introducing *assumptions* in the middle of a proof. These can be viewed as "temporary axioms" that are "discharged" when no longer needed.

To present propositional logic in this fashion, the WFPs are restricted here to those whose only connectives are $\neg$ and $\wedge$. The following may be used as rules of inference:

($\wedge$ Introduction) (a) From $\mathbf{P}$ and $\mathbf{Q}$, infer $(\mathbf{P} \wedge \mathbf{Q})$.
(b) From $\mathbf{P}$ and $\mathbf{Q}$, infer $(\mathbf{Q} \wedge \mathbf{P})$.

($\wedge$ Elimination) (a) From $(\mathbf{P} \wedge \mathbf{Q})$, infer $\mathbf{P}$.
(b) From $(\mathbf{P} \wedge \mathbf{Q})$, infer $\mathbf{Q}$.

Table 6. A Proof from Axioms

1. $((\mathbf{P} \rightarrow ((\mathbf{P} \rightarrow \mathbf{P}) \rightarrow \mathbf{P})) \rightarrow ((\mathbf{P} \rightarrow (\mathbf{P} \rightarrow \mathbf{P})) \rightarrow (\mathbf{P} \rightarrow \mathbf{P})))$	; this is an axiom, since it is a ; WFP with the form of axiom ; schema (A2), with **P** and **R** both ; replaced by **P**, and **Q** replaced ; by '$(\mathbf{P} \rightarrow \mathbf{P})$'
2. $(\mathbf{P} \rightarrow (\mathbf{P} \rightarrow \mathbf{P}))$	; (A1), with **Q** replaced by **P**
3. $(\mathbf{P} \rightarrow ((\mathbf{P} \rightarrow \mathbf{P}) \rightarrow \mathbf{P}))$	; (A1), with **Q** replaced by ; '$(\mathbf{P} \rightarrow \mathbf{P})$'
4. $((\mathbf{P} \rightarrow (\mathbf{P} \rightarrow \mathbf{P})) \rightarrow (\mathbf{P} \rightarrow \mathbf{P}))$	; from 3, 1 by (MP)
5. $(\mathbf{P} \rightarrow \mathbf{P})$	; from 2, 4 by (MP)

($\neg$ Introduction) If both **Q** and $\neg$ **Q** can be inferred from an assumption **P**, then infer $\neg$ **P**.

($\neg$ Elimination) If both **Q** and $\neg$ **Q** can be inferred from an assumption $\neg$ **P**, then infer **P**.

A notion of *subproofs* is needed for the last two rules, along with rules allowing propositions to be "sent" into the subproofs and "returned" from them under certain restrictions. Subproofs can be indicated by prefixing stars to the lines of a subproof, where the number of stars indicates the level of nesting of the subproof [for details, see Schagrin, Rapaport, and Dipert (1985)]. As an example, Table 7 contains a natural deduction proof of the argument

$$\neg (A \wedge B), A \vdash \neg B$$

Each line of the proof has a comment following the semicolon, and subproofs have "begin" and "end" comments; these are not formally part of the proof, but they play the same role that comments do in computer programs. For details and for rules of inference for other connectives, see Copi (1979); Kalish, Montague, and Mar (1980); Schagrin, Rapaport, and Dipert (1985). Natural deduction systems have been extensively investigated (see Szabo, 1969).

AI and Propositional Logic. Newell, Shaw, and Simon's Logic Theorist program (1963), considered by some to be the first AI program, used a breadth-first search procedure to prove theorems of propositional logic. It successfully proved 38 of the first 52 theorems of Whitehead and Russell's *Principia Mathematica* (1927). A successor program, the General Problem Solver (Ernst and Newell, 1969; Newell, Shaw, and Simon, 1960) used means-ends analysis to solve problems in a variety of domains, including propositional logic. Discussions of these programs may be found in Barr and Feigenbaum (1981) and Slagle (1971).

Another important propositional logic program is Wang's algorithm, which is a more efficient method for determining whether a given argument is valid than using truth tables. This algorithm attempts to interpret the premises of the argument as all true and the conclusion as false. If it succeeds in this attempt, the argument is shown to be invalid; if it fails, the argument is shown to be valid. For details, see Schagrin, Rapaport and Dipert, 1985.

A rule of inference that has proved to be of importance in AI and automated theorem proving, in part because most of the introduction and elimination rules can be shown to be instances of it, is

(Resolution) From $(\neg \mathbf{P} \vee \mathbf{Q})$ and $(\mathbf{P} \vee \mathbf{R})$, infer $(\mathbf{Q} \vee \mathbf{R})$

For a discussion of AI systems that use propositional logic inference techniques based on Resolution, see RESOLUTION; THEOREM PROVING; as well as such AI texts as Manna (1974), Nilsson (1971, 1980), Raphael (1976), Rich (1983), and Winston (1984).

Table 7. A Natural-Deduction Proof

1. $\neg (A \wedge B)$	; this is the first premise
2. A	; this is the second premise
; BEGIN subproof using $\neg$ Introduction to prove $\neg$ B	
* 3. B	; an assumption for use by $\neg$ Introduction
* 4. A	; sent in from line 2 of main proof ; (similar to parameter passing in ; procedures)
* 5. $(A \wedge B)$	; $\wedge$ Introduction using lines 3, 4
* 6. $\neg (A \wedge B)$	; sent in from line 1 of the main proof
* 7. $\neg$ B	; $\neg$ Introduction, from lines 3, 5, 6
; END of subproof that proved $\neg$ B by $\neg$ Introduction	
8. $\neg$ B	; returned from line 7 of subproof ; (similar to a procedure returning a ; value)

Semantics

An *argument* is any inference from *hypotheses* (or *premises*) to a *conclusion.* Thus, rules of inference are essentially forms of argument. A rule of inference or an argument should never lead from truth to falsehood. To say that a rule of inference or an argument is *valid* means: if the hypotheses are true, then the conclusion must be true. Validity, thus, can be construed as a notion of truth *relative* to the premises. An argument is said to be *sound* (in one sense) iff it is valid and its hypotheses are, in fact, true.

Truth tables can be used for semantic inference—as opposed to the syntactic inference discussed in the previous section. Here, a truth table is constructed whose "input" columns contain the premises and whose "output" column contains the conclusion. The argument is valid iff there is no line of the truth table with T in all premise columns and F in the conclusion column.

Propositional logic is also said to be *sound* in the sense that all of its theorems are tautologies. It is also *complete:* all propositional tautologies are theorems of propositional logic. There is a link between a proposition's being a tautology and an argument's being valid: For any proof $\mathbf{P}_1, \ldots, \mathbf{P}_{n-1} \vdash \mathbf{P}_n$, there corresponds the material-conditional proposition: $((\mathbf{P}_1 \wedge \cdots \wedge \mathbf{P}_{n-1}) \rightarrow \mathbf{P}_n)$. The former is valid iff the latter is a tautology [for details on these topics, see Mendelson (1979) and Kleene (1950)]. This follows (by soundness and completeness) from the Deduction Theorem, which states that the former is valid iff the latter is a theorem. Finally, propositional logic is also *decidable:* There is an algorithm such that for any WFP, the algorithm decides whether the WFP is a theorem (ie, one can use a truth table to determine whether the WFP is a tautology). However, the decidability of propositional logic is an NP-complete problem and hence computationally "intractable"; this fact has been employed in philosophical arguments to the effect that such logics are not well-suited to computational models of rationality (Cherniak, 1984).

BIBLIOGRAPHY

A. R. Anderson and N. D. Belnap, Jr., *Entailment: The Logic of Relevance and Necessity,* Princeton University Press, Princeton, New Jersey, 1975.

A. Barr and E. A. Feigenbaum, eds., *The Handbook of Artificial Intelligence,* Vol. 1, William Kaufmann, Los Altos, Calif., 1981.

H.-N. Castañeda, *Thinking and Doing,* D. Reidel, Dordrecht, 1975.

C. Cherniak, "Computational Complexity and the Universal Acceptance of Logic," *J. Philos.* **81,** 739–758 (1984).

A. Church, *Introduction to Mathematical Logic,* Princeton University Press, Princeton, New Jersey, 1956, pp. 23–31.

I. M. Copi, *Symbolic Logic,* 5th ed., Macmillan, New York, 1979.

G. W. Ernst and A. Newell, *GPS: A Case Study in Generality and Problem Solving,* Academic Press, New York, 1969.

R. M. Gale, "Propositions, Judgments, Sentences, and Statements," in P. Edwards, ed., *Encyclopedia of Philosophy,* Vol. 6, Macmillan and Free Press, New York, 1967, pp. 494–505.

G. E. Hughes and M. J. Cresswell, *An Introduction to Modal Logic,* Methuen, London, 1968.

D. Kalish, R. Montague, and G. Mar, *Logic: Techniques of Formal Reasoning,* 2nd ed., Harcourt Brace Jovanovich, New York, 1980.

S. C. Kleene, *Introduction to Metamathematics,* Van Nostrand, Princeton, N.J., 1950.

Z. Manna, *Mathematical Theory of Computation,* McGraw-Hill, New York, Chapt. 2, 1974.

J. D. McCawley, *Everything that Linguists Have Always Wanted to Know about Logic but Were Ashamed to Ask,* University of Chicago Press, Chicago, 1981.

E. Mendelson, *Introduction to Mathematical Logic,* 2nd ed., Van Nostrand, New York, 1979.

A. Newell, J. C. Shaw, and H. Simon, "A Variety of Intelligent Learning in a General Problem-Solver," in M. C. Yovits and S. Cameron, eds., *Self-Organizing Systems,* Pergamon, New York, 1960, pp. 153–189.

A. Newell, J. C. Shaw, and H. Simon, "Empirical Explorations of the Logic Theory Machine," in E. Feigenbaum and J. Feldman, eds., *Computers and Thought,* McGraw-Hill, New York, 1963, pp. 109–133.

N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence,* McGraw-Hill, New York, 1971.

N. J. Nilsson, *Principles of Artificial Intelligence,* Tioga, Palo Alto, Calif., 1980.

B. Raphael, *The Thinking Computer: Mind Inside Matter,* W. H. Freeman, San Francisco, 1976.

N. Rescher, *The Logic of Commands,* Routledge and Kegan Paul, London and Dover, New York, 1966.

N. Rescher, *Many-Valued Logic,* McGraw-Hill, New York, 1969.

E. Rich, *Artificial Intelligence,* McGraw-Hill, New York, 1983.

M. L. Schagrin, *The Language of Logic: A Self-Instruction Text,* 2nd ed., Random House, New York, 1979.

M. L. Schagrin, W. J. Rapaport, and R. R. Dipert, *Logic: A Computer Approach,* McGraw-Hill, New York, 1985.

S. C. Shapiro, "The SNePS Semantic Network Processing System," in N. V. Findler, ed., *Associative Networks,* Academic Press, New York, 1979, pp. 179–203.

J. R. Slagle, *Artificial Intelligence: The Heuristic Programming Approach,* McGraw-Hill, New York, 1971.

M. E. Szabo, ed., *Collected Papers of Gerhard Gentzen,* North-Holland, Amsterdam, 1969.

A. N. Whitehead and B. Russell, *Principia Mathematica,* 2nd ed., Cambridge University Press, Cambridge, UK, 1927.

P. H. Winston, *Artificial Intelligence,* 2nd ed., Addison-Wesley, Reading, Mass., 1984.

General References

A. E. Blumberg, "Logic, Modern," in P. Edwards, ed., *Encyclopedia of Philosophy,* Vol. 5, Macmillian and Free Press, New York, 1967, pp. 12–34.

R. C. Jeffrey, *Formal Logic: Its Scope and Limits,* 2nd ed., McGraw-Hill, New York, 1981.

W. Kneale and M. Kneale, *The Development of Logic,* Oxford University Press, Oxford, 1962.

W. V. O. Quine, *Mathematical Logic,* rev. ed., Harper & Row, New York, 1951.

W. V. O. Quine, *Elementary Logic,* rev. ed., Harvard University Press, Cambridge, Mass., 1980.

W. V. O. Quine, *Methods of Logic,* 4th ed., Harvard University Press, Cambridge, Mass., 1982.

W. J. RAPAPORT
SUNY at Buffalo

LOGO

LOGO is a programming language in the spirit of LISP invented in the MIT AI Lab to teach mathematical concepts to little children [see S. Papert, "Teaching Children to be Mathematicians versus Teaching about Mathematics," *Int. J. Math. Educ. Sci. Technol.* **3,** 249–262 (1972)]. A program in LOGO manipulates a little device called the "turtle." This turtle moves on a large flat surface. With two commands, PENUP and PENDOWN, it is possible to create a trace of the turtle movements. The goal of a program is usually to draw a certain figure; therefore, programming in LOGO is referred to as "turtle geometry." Typically, commands would be FORWARD 100, RIGHT 60, and BACK 100. It is possible to define procedures. In later research, LOGO has been used to help people learn about powerful ideas and to study the acquisition of computational skills by young children (see C. J. Solomon and S. Papert, "A Case Study of a Young Child Doing Turtle Graphics in LOGO," *Proceedings of the National Computer Conference,* AFIPS, pp. 1049–1056, 1976). More information can be found in the MIT AI Lab LOGO Memos.

J. GELLER
New Jersey Institute of Technology

LOOPS

LOOPS was one of the first multi-paradigm programming environments. Developed at Xerox PARC in Interlisp, it added an object-oriented programming system similar to Smalltalk to the procedure-oriented programming of LISP. It also incorporated access-oriented programming allowing changes in objects to trigger computation (useful for monitoring), rule-oriented programming often used for simple expert systems, and a visual programming interface that supported graphic exploration and modification of program and data structures. The integration of paradigms was designed to support the rapid development of knowledge-based systems. [See M. Stefik, D. G. Bobrow, S. Mittal, and L. Conway, "Knowledge Programming in LOOPS: Report on an Experimental Course," *AI Magazine* **4**(3), 3–13 (1983).] Recent versions of Common LISP integrate CLOS (qv), a new standard object-oriented programming substrate that carries over many of the ideas from LOOPS.

DANIEL G. BOBROW
Xerox PARC

LOPS

An approach to program synthesis, based on transformation of logical formulas and guided by powerful fundamental strategies [see W. Bibel, "Syntax-Directed, Semantics-Supported Program Synthesis," *Artif. Intell.* **14,** 243–261 (1980)]. A LISP implementation of this approach is presented in W. Bibel and K. M. Hörnig, "LOPS—A System Based on a Strategical Approach to Program Synthesis," in A. Biermann, G. Guiho, and Y. Kodratoff, eds., *Automatic Program Construction Techniques,* MacMillan, New York, 1984. A more elaborate and logic programming oriented implementation is described in G. Neugebauer, B. Fronhöfer, and C. Kreitz, "XPRTS—An Implementation Tool for Program Synthesis," in D. Metzing, ed., *Proceedings of the Thirteenth German Workshop on Artificial Intelligence,* Geseke, Germany, Sept. 1989, Informatik-Fachberichte 216, Springer, Berlin, 1989, pp. 348–357.

BERTRAM FRONHÖFER
Technical University Munich

LUNAR

LUNAR is a natural language question-answering system developed by W. Woods and his colleagues at BBN in the early 1970s for the NASA Manned Spacecraft Center. The system answered questions about the chemical composition of the Apollo 11 moon rocks. LUNAR was one of the first successful natural language understanding systems and pioneered the concept of natural language front ends to databases. Its linguistic fluency was substantial: it handled relative clauses, passive sentences, verb complement constructions, complex quantification, and pronominal references. It could specify and perform averaging calculations and included capabilities for both document retrieval and fact retrieval. LUNAR was demonstrated to the public at the Second Annual Lunar Sciences Conference in Houston, Texas in 1971. [See W. A. Woods, R. M. Kaplan, and B. Nash-Webber, *The LUNAR Sciences Natural Language Information System: Final Report,* BBN Report No. 2378, Bolt, Beranek and Newman, Cambridge, Mass., 1972 (available from NTIS as N72-28984); see also W. A. Woods, "Progress in Natural Language Understanding: An Application to Lunar Geology," *AFIPS Conference Proceedings,* Vol. 42, National Computer Conference and Exposition, 1973; and W. A. Woods, *Semantics for a Question-Answering System,* Garland Publishing Co., New York, 1979.]

W. A. WOODS
Harvard University