LOGIC IN AI

J.-J.Ch. Meyer
Department of Computing Science, Utrecht University, The Netherlands

Keywords: propositional logic, first-order predicate logic, second-order logic, deduction theorem, (semi-)decidability, modal logic, possible world semantics, intelligent agent, BDI (belief, desire, intention), dynamic logic, deontic logic, temporal logic, epistemic logic, nonmonotonic logic, default logic, frame problem, qualification problem, closed world assumption, circumscription, preferential semantics, multi-valued logic, fuzzy logic.

Contents

1. Introduction
2. Classical Logic
   2.1 Classical propositional logic
   2.2 First-Order Predicate Logic
3. Modal Logic
   3.1 Dynamic logic
   3.2 Deontic logic
   3.3 Temporal logic
   3.4 Epistemic logic
4. Nonmonotonic Logic
   4.1 Default Logic
   4.2 Circumscription
5. Multi-Valued Logic and Fuzzy Logic
6. Conclusion
Glossary
Bibliography
Biographical Sketch

Summary

In this article we will give an overview of the role of logic in artificial intelligence. After a general discussion of this role we turn to a more detailed treatment of classical propositional and first-order logic, and next to the most popular nonclassical logics in AI, viz. modal logic, nonmonotonic logic and multi-valued logic.

1. Introduction

The discipline of artificial intelligence (AI) studies the question of how artifacts can be ascribed or endowed with intelligence. In other words, AI concerns questions such as to how to 'implement intelligence into artificial systems'. Of course, since the very concept of intelligence is not fully understood, this means that in order to do something sensible, one has to use some kind of working definition. In his (philosophical) introduction to AI Copeland defines an agent (artificial or otherwise) to be intelligent if it is 'massively adaptable', i.e. if the agent is flexible to the degree that it can cope with all kinds of
changes of situations in the world it's inhabiting. The latter may, of course, be the real physical world we all live in, but one may also think of a more artificial world, such as the internet or some computer-generated virtual reality.

Since intelligence often seems to involve some kind of reasoning it becomes clear that logic, the science of reasoning, may play an important role in AI. This is true to the extent that one adheres to the view that indeed intelligence has to do with or even can be described in terms of symbolic means. At the moment there appear to be at least two tendencies among AI researchers, one which holds that intelligence is to be described and implemented in a symbolic way, and one which maintains that this view is inadequate for implementing intelligence onto an artificial system. The former group of researchers ('the symbolists') indeed ascribe an important role to logic. Here the idea is to lay down intelligent behaviour in formal/logical rules which can then be programmed (by means of some programming language such as LISP or PROLOG) into a system. The latter group does not believe that this can be done (since it is too complex, for example), and claims that one has to resort to other ('non-symbolic' or 'subsymbolic') means like techniques inspired by biological organisms (like neural networks and evolutionary computing mechanisms).

As to the symbolistic approach to AI one may again have different views as to the exact role of logic in this enterprise. For example, is logic itself to be considered / employed as a programming language, or at least as a kind of executable specification language, or does logic 'merely' serve as an intermediary to get the concepts right and precise, after which one may implement these by means of a procedural programming language? Also, there is the question which logic is to be employed. Contrary to what one may think, there are many different logics. There is the familiar classical (prepositional and first-order predicate) logic, but especially in the last century there have been developed many 'non-classical' logics, which typically focus around some particular feature of reasoning.

In this chapter we will briefly sketch some of these logics and indicate their role / use in AI. (More about the role of logic in AI and knowledge representation can be found in the references.)

2. Classical Logic

The logic (or rather logics) usually referred to as classical logic comprises classical prepositional and first-order logic. Although these logics may sometimes be regarded as insufficient for AI purposes (as we will see below in subsequent sections), they nevertheless have had a tremendous impact on later developments, and, moreover, within AI these classical logics are still widely employed in many applications. For example, many ways of knowledge representation in knowledge-based systems are still using classical logic, and also the influential logic programming paradigm (such as the AI language PROLOG) is primarily based upon classical logic. For this reason, as well as for the reason that it provides the basis for many 'non-classical' logics, we here give a succinct treatment of classical prepositional and first-order logic. (More can be found in the many textbooks and handbooks on this subject, such as listed in the bibliography.)
2.1 Classical prepositional logic

As indicated by its name prepositional logic is about reasoning about propositions, i.e. assertions that can be true or false. The logic is therefore built on a set of primitive or atomic propositions, sometimes also called atoms. Let’s call this set \( \varphi \). Now complex propositions may be constructed from the primitive ones by using connectives, such as 'logical and' (\( \land \)), 'logical or' (\( \lor \)), 'logical negation' (\( \neg \)), 'logical implication' (\( \rightarrow \)) and 'logical equivalence' or 'logical bi-implication' (\( \leftrightarrow \)). We denote formulas by \( \varphi, \psi, \chi \), possibly endowed with marks and indexes. So complex formulas may have the following form: \( \varphi \land \psi \), \( \varphi \lor \psi \), \( \neg \varphi \), \( \varphi \rightarrow \psi \), \( \varphi \leftrightarrow \psi \).

The meaning of formulas in classical prepositional logic is given by assigning truth values to these formulas on the basis of an assignment of truth values to the primitive propositions. So let the valuation function \( \pi \) be a function that assigns truth values to the primitive atoms, i.e. \( \pi : \varphi \rightarrow T \), where \( T = \{ tt, ff \} \). This means that the function \( \pi \) assigns a truth values for any atom \( p \in \varphi \):

\[ \pi(p) \in T. \]

The prepositional meaning (i.e. truth value) \( V \) of a complex formula is now obtained by considering the way such a formula is constructed. For convenience, we order the truth values in \( T \) as follows: \( ff \leq ff \), \( ff \leq tt \), \( tt \leq tt \), and use functions \( min \) and \( max \) on truth values, taking the minimum and maximum values, respectively, according to this ordering. Furthermore, we have a function \( compl \) which takes the complement of a truth value, i.e. \( compl(tt) = ff \) and \( compl(ff) = tt \).

- \( V(p) = \pi(p) \text{ if } p \in \varphi \);
- \( V(\varphi \land \psi) = min(V(\varphi), V(\psi)) \);
- \( V(\varphi \lor \psi) = max(V(\varphi), V(\psi)) \);
- \( V(\neg \varphi) = compl(V(\varphi)) \);
- \( V(\varphi \rightarrow \psi) = max(compl(V(\varphi)), V(\psi)) \);
- \( V(\varphi \leftrightarrow \psi) = min(V(\varphi \rightarrow \psi), V(\psi \rightarrow \varphi)) \)

So, for example, if it is given that the atoms \( p \) is true and \( q \) is false (i.e. for \( p, q \in \varphi \) we have that \( \pi(p) = tt \) and \( \pi(q) = ff \)), then we have that the formula \( p \rightarrow q \) is false, since \( V(p \rightarrow q) = max(compl(V(p)), V(q)) = max(compl(\pi(p)), \pi(q)) = max(compl(tt), ff) = max(ff, ff) = ff \).

An alternative way of formulating the truth of a formula, which will easily generalize to some of the other logics we will see in the sequel, is the following. We consider the notion of a model that makes a formula true. Since in prepositional logic a model is just a valuation function \( V \), we will write \( V \models \varphi \) to denote that model \( V \) makes \( \varphi \) true. Strictly speaking the valuation function \( V \) depends on the underlying truth assignment function \( \pi \) and should be written \( V^{\pi} \). However, we omit this superscript for notational convenience.

Now we have that:

- \( V \models p \text{ iff } \pi(p) = tt \text{ for } p \in \varphi \);
- \( V |\ = \varphi \land \psi \) iff \( V |\ = \varphi \) and \( V |\ = \psi \);
- \( V |\ = \varphi \lor \psi \) iff \( V |\ = \varphi \) or \( V |\ = \psi \);
- \( V |\ = \neg \varphi \) iff not \( V |\ = \psi \);
- \( V |\ = \varphi \rightarrow \psi \) iff \( V |\ = \varphi \) implies \( V |\ = \psi \);
- \( V |\ = \varphi \leftrightarrow \psi \) iff \( V |\ = \varphi \) bi-implies \( V |\ = \psi \).

Formulas \( \varphi \) that obtain the truth value \( \text{tt} \) no matter what the truth values of the atoms occurring in it are (or, more formally, \( V(\varphi) = \text{tt} \) for all functions \( \pi \) assigning truth values to the atoms), are called \textit{valid} or \textit{tautological}. If \( \psi \) is valid, we denote this by \( |= \varphi \). These formulas express 'necessary' truth under all circumstances. Examples of tautologies are:

- \( |= \varphi \rightarrow \varphi \)
- \( |= (\varphi \land \psi) \rightarrow \varphi \)
- \( |= \varphi \rightarrow (\varphi \lor \psi) \)
- \( |= \varphi \leftrightarrow \neg \varphi \)
- \( |= \varphi \lor \neg \varphi \)
- \( |= \neg (\varphi \land \psi) \leftrightarrow (\neg \varphi \lor \neg \psi) \)
- \( |= \neg (\varphi \lor \psi) \leftrightarrow (\neg \varphi \land \neg \psi) \)
- \( |= \neg \varphi \rightarrow (\varphi \rightarrow \psi) \)
- \( |= \varphi \rightarrow (\psi \lor \neg \psi) \)

The first tautology should be obvious. The second one expresses a property of conjunction: a conjunction implies its conjuncts. The third one states a a property of disjunction: one can always weaken a logical assertion by adding a disjunct. The fourth one is the law of double negation: in classical logic double negation can be eliminated. The fifth one is the well-known law of the excluded middle. The sixth and seventh ones are the rules of De Morgan. The eighth and ninth ones are versions of the so-called paradoxes of the material implication: they state that from a false assertion anything follows, and that anything implies a true assertion, respectively.

Another important notion is that of (semantic) entailment: \( \Phi |= \psi \) (where \( \Phi \) is a set of formulas and \( \psi \) is a formula) expresses that if all formulas \( \varphi \in \Phi \) are true with respect to a valuation function, also the formula \( \psi \) is true with respect to that valuation function: for any \( V \) it holds that if \( V(\varphi) = \text{tt} \) for all \( \varphi \in \Phi \), then also \( V(\psi) = \text{tt} \). Another way of saying the same is: "all models of the formulas \( \varphi \in \Phi \), are also models of the formula \( \psi \). (Recall that models are here the valuation functions \( V \).) Examples are: \( \varphi |= \varphi \) and \( \varphi \land \psi |= \varphi \). There is an elegant relation between semantic entailment and validity, which is called the (semantic version of the) \textit{deduction theorem}:

**Theorem 2.1** For any formulas \( \varphi_1, \ldots, \varphi_m \) it holds that:

\[
\varphi_1, \ldots, \varphi_m |= \psi \iff |= (\varphi_1 \land \ldots \land \varphi_m) \rightarrow \psi
\]

In order to establish whether a formula is valid one may perform a systematic enumeration of all possibilities for truth assignments for the atoms ("truth tables"), but there are also other methods of answering this question, such as 'tableaux' or employing...
an axiomatic system. In the latter method one uses axioms and rules to try and derive the formula at hand, starting with an axiom (or several of them) and then applying rules. If one succeeds one knows that the formula is indeed a valid formula. For this to work it is necessary to employ an axiom system that is so-called sound and complete. Soundness of an axiomatic system means that every axiom is indeed valid and every rule preserves validity, while completeness of an axiomatic system says that every valid formula can be derived by means of the system (by using the axioms and rules) within some finite number of steps (if one is clever enough!).

An example of a validity-preserving rule is the well-known modus ponens rule: if $\models \varphi$ and $\models \varphi \rightarrow \psi$, then also $\models \psi$. This rule is often written as

$$\varphi, \varphi \rightarrow \psi \quad \psi$$

There are many sound and complete axiom systems known for classical prepositional logic. A disadvantage of the axiomatic method is that if one does not succeed in deriving the formula at hand, it is not clear whether the formula is indeed non-derivable (and thus not valid by the completeness of the system) or it is the case that one has not yet tried enough to obtain it. In fact, one can only really show that a formula $\varphi$ is not valid by giving a so-called countermodel, i.e. a truth assignment $\pi$ for which $V(\varphi) = \text{ff}$.

### 2.2 First-Order Predicate Logic

First-order predicate logic is a first extension of classical prepositional logic in the sense that now the primitive propositions from prepositional logic get more structure. For instance, in first-order predicate logic one may express that all objects in a room are red. To this end the syntax of the logic is equipped with so-called predicate, constant and function symbols. The predicate symbols are used to denote properties of elements in the domain of discourse and relations between them. The constant symbols are used to refer to elements in the domain. The function symbols are used to denote particular mathematical functions on the domain and thus may be used to refer indirectly to particular objects in that domain as well.

Both predicate symbols and function symbols have a so-called *arity*, that is the number of arguments they have. For instance, a predicate symbol '>$' that will be used for denoting the relation 'greater than' on a domain of numbers will typically have arity 2: $>(m_1, m_2)$, or mostly written as $m_1 > m_2$. A predicate symbol $R$ denoting a property like 'Red' will typically have arity 1: $R(x)$. But in general we may have predicate symbols with arity $n$. A function symbol '+' denoting the mathematical function of addition will have arity 2: $(m_1, m_2)$, mostly written as $m_1 + m_2$, whereas a function symbol '√' for the square root function will have arity 1: $\sqrt{x}$. By the way, constant symbols may be regarded as function symbols with arity 0, while predicate symbols with arity 0 may be viewed as prepositional primitives in the sense of prepositional logic!

We also have a set $\text{Var}$ of (first-order) variables in our language which we will use in (universally or existentially) quantified expressions. Terms are either variables, constants or functions applied to terms (as many as the arity of the function). Atomic
Formulas are of the form \( P(t_1, t_2, \ldots, t_n) \), where \( P \) is a predicate symbol of arity \( n \), and \( t_1, t_2, \ldots, t_n \) are terms. Formulas are either atomic or formulas connected by the usual logical connectives \( \wedge, \lor, \neg, \rightarrow, \leftrightarrow \), or by using the universal and existential quantifier: \( \forall x \varphi \) and \( \exists x \varphi \). (Here \( x \) is a variable, which typically (but not necessarily) will also occur in the formula \( \varphi \).) The reading of a formula of the form \( \forall x P(x) \) is "for all elements in the domain it holds that they have property \( P \)", while \( \exists x P(x) \) reads "there exists an element in the domain for which property \( P \) holds."

We call an occurrence of a variable \( x \) free in a formula \( \varphi \) if that occurrence does not occur in the scope of a quantifier, or put in other words, if that occurrence is not bound by some quantifier. The models for an first-order language is a structure \( M = (D, I) \), where \( D \) is a non-empty set, the domain of discourse, the objects which we talk about (e.g. the natural numbers, but it can be anything), and \( I \) is an interpretation function that interprets the predicate, constant and function symbols as relations, elements and mathematical functions on the domain \( V \), respectively. E.g. if \( D \) is the set of natural numbers, the interpretation \( I(\cdot) \) of a predicate symbol '\( > \)' may be the greater than relation on the natural numbers, the interpretation \( I(0) \) of a constant symbol '0' may be the number zero, and the interpretation \( I(\cdot) \) of the function symbol '\( + \)' may be the mathematical function of addition.

To interpret variables we need a valuation function \( v \) of type \( Var \rightarrow V \), yielding for each variable from the set \( Var \) a value in the domain \( V \). In the sequel we will use the notation \( v[d/x] \) for the modified valuation which is like \( v \), but with \( x \) set to \( d \). More formally,

\[
v[d/x](y) = \begin{cases} d, & \text{if } y = x; \\ v(y), & \text{otherwise} \end{cases}
\]

The interpretation of terms is now given inductively by a function \( I^M_v \) as follows:

- \( I^M_v(c) = I(c) \) if \( c \) is a constant symbol;
- \( I^M_v(x) = v(c) \) if \( c \) is a variable in \( Var \);
- \( I^M_v(F)(t_1, \ldots, t_n) = I(F)(I^M_v(t_1), \ldots, I^M_v(t_n)) \) if \( F \) is a function symbol of arity \( n \)

Finally the truth of formulas given a model \( M \) and a valuation \( v \), denoted \( M, v \models \varphi \) is given inductively by:

- \( M, v \models P(t_1, \ldots, t_n) \text{ iff } I(P)(I^M_v(t_1), \ldots, I^M_v(t_n)) \) holds.
- \( M, v \models \varphi \wedge \psi \text{ iff } M, v \models \varphi \text{ and } M, v \models \psi \)
- \( M, v \models \neg \varphi \text{ iff } \text{not } M, v \models \varphi \)
- \( M, v \models \varphi \rightarrow \psi \text{ iff } M, v \models \varphi \text{ implies } M, v \models \psi \)
- \( M, v \models \varphi \leftrightarrow \psi \text{ iff } M, v \models \varphi \text{ bi-implies } M, v \models \psi \)
- \( M, v \models \forall x \varphi \text{ iff } M, v[d/x] \models \varphi \text{ for all } d \in D \)
- \( M, v \models \exists x \varphi \text{ iff } M, v[d/x] \models \varphi \text{ for some } d \in D \)

A formula \( \varphi \) is called (universally) valid (denoted \( \models \varphi \)) iff it is true in every model and for every valuation, i.e. \( M, v \models \varphi \) for any first-order model \( M \) and every valuation.
function $v$. A few examples of valid formulas are:

- $|$ $\forall x \varphi \iff \exists x \neg \varphi$
- $|$ $\exists x \varphi \iff \forall x \neg \varphi$
- $|$ $\forall x (\varphi \land \psi) \iff (\forall x \varphi \land \forall x \psi)$
- $|$ $\exists x (\varphi \lor \psi) \iff (\exists x \varphi \lor \exists x \psi)$
- $|$ $\forall x \forall y \varphi \iff \forall y \forall x \varphi$
- $|$ $\exists x \exists y \varphi \iff \exists y \exists x \varphi$
- $|$ $\forall x \varphi \rightarrow \exists x \varphi$
- $|$ $(\forall x \varphi \lor \forall x \psi) \rightarrow \forall x (\varphi \lor \psi)$
- $|$ $\exists x (\varphi \land \psi) \rightarrow (\exists x \varphi \land \exists x \psi)$
- $|$ $\exists x \forall y \varphi \rightarrow \forall y \exists x \varphi$
- $|$ $\forall x (\varphi \rightarrow \psi) \rightarrow (\forall x \varphi \rightarrow \forall x \psi)$
- $|$ $\forall x (\varphi \rightarrow \psi) \leftrightarrow (\exists x \varphi \rightarrow \psi)$ if $x$ does not occur free in the formula $\varphi$
- $|$ $\forall x (\varphi \rightarrow \psi) \leftrightarrow ((\exists x \varphi) \rightarrow \psi)$ if $x$ does not occur free in the formula $\psi$

Again we have that modus ponens is a validity-preserving rule in the sense that if $|= \varphi$ and $|= \varphi \rightarrow \psi$, then also $|= \psi$. Also we can define a notion of (semantic) entailment for which the deduction theorem holds. We will not pursue this here. As for classical propositional logic there are also again sound and complete proof systems for first-order predicate logic. However, there is not a mechanical method to decide the validity of a formula as was the case in propositional logic: first-order predicate logic is only semi-decidable. If a formula is a validity one is able to derive it by means of a complete proof system. However, while trying to derive a formula there is no way to tell whether this procedure will ever end. Technically this means, although the validities of first-order logic are recursively enumerable (by means of an algorithm, more or less deriving all theorems of a complete proof system in a systematic way), the set of validities is not decidable: a formula is not to be decided as a validity within a finite amount of time.

Finally a remark on higher-order logic. It is possible to extend first-order logic with high-order variables ranging over sets of domain elements (thus ranging over properties and relations), but this results in such an expressive logic that there is not even a complete proof system any more. So the validity in this logic is not even semi-decidable any more. However, we'll see in the sequel that higher-order logic may come in handy when we are interested in certain forms of non-monotonic logic, viz. circumscription.
Bibliography

General note on the references. The references contain a number of (articles in) handbooks. These generally give excellent surveys, but mostly they are written for an advanced audience. Also a number of "Readings in" books are referenced. These are mostly collections of seminal papers in the field (sometimes really defining or starting it), and are thus worthwhile for further reading.


D.J. Israel, The Role(s) of Logic in Artificial Intelligence, in: D.M. Gabbay, C.J. Hogger & J.A. Robinson (eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 1, Logical
ARTIFICIAL INTELLIGENCE – Logic in AI - J.-J. Ch. Meyer

R. Reiter, A Logic for Default Reasoning, Artificial Intelligence 13, 1980, pp. 81-132. [Seminal paper on the first forms of monotonic logic, viz. circumscription and default logic]
R. Turner, Logics for Artificial Intelligence, Ellis Horwood/Wiley. Chich-ester/New York, 1984. [A nice little book which may spark your interest in nonclassical logics as applied in AI (it did so in my case)]
on dynamic and epistemic logic]


**Biographical Sketch**

**Prof. Dr. John-Jules Ch. Meyer** is head of the Intelligent Systems Group of the Institute of Information and Computing Sciences at Utrecht University, and currently research director of this institute and vice-dean of the Faculty of Mathematics and Computer Science at this university. He holds a MSc in Mathematics from Leyden University, and a PhD in Computer Science from the Free University Amsterdam on a thesis, entitled "Programming Calculi Based on Fixed Point Transformations", on the semantics of programming languages.

From 1985 to 1993 he was successively an assistant, associate and full professor at the Computer Science Department of the Free University Amsterdam, the last on a chair "Logic for Distributed Systems and Artificial Intelligence". From 1989 to 1993 he was also a (part-time) full professor of Theoretical Computer Science at Nijmegen University. Since 1993 he has been a (full) professor at Utrecht University. Furthermore, he is currently the scientific director of the Dutch Graduate School in Information and Knowledge-based Systems (SIKS), chairman of the Dutch Association of Logic and Philosophy of Science, member of the boards of the Dutch National Platform for Computer Science Research and the Dutch Association of Theoretical Computer Science. His research interests include artificial intelligence, agent technology, cognitive robotics, applied logic, and semantics of programming languages.

He has published over 150 papers on these subjects in international journals and conference proceedings. He is an editor of the Journal of Applied Non-Classical Logic(s), Data and Knowledge Engineering, and the Journal of Intelligent Agents & Multi-Agent Systems. He has co-authored and co-edited several books on agent technology, deontic logic, epistemic logic and nonmonotonic reasoning.

He has been a member of over 100 (national and international) PhD committees in the area of computer science and AI, at which he acted 19 times as a promotor.