# Logic and Artificial Intelligence: Divorced, Still Married, Separated . . . ?*

Selmer Bringsjord//David A. Ferrucci

February 13, 1998

**Abstract**

Though it's difficult to agree on the exact date of their union, logic and artificial intelligence (AI) were married by the late 1950s, and, at least during their honeymoon, were *happily* united. What connubial permutation do logic and AI find themselves in now? Are they still (happily) married? Are they divorced? Or are they only separated, both still keeping alive the promise of a future in which the old magic is rekindled? This paper is an attempt to answer these questions via a review of six books. Encapsulated, our answer is that (i) logic and AI, despite tabloidish reports to the contrary, still enjoy matrimonial bliss, and (ii) only their future robotic offspring (as opposed to the children of connectionist AI) will mark real progress in the attempt to understand cognition.

## 1 The Logic–AI Marriage

Though it's difficult to agree on the exact date of their union, logic and artificial intelligence (AI) were married by the late 1950s, and, at least during their honeymoon, were *happily* married. But what connubial permutation do logic and AI find themselves in now? Are they still (happily) married? Are they divorced? Or are they only separated, both still keeping alive the promise of a future in which the old magic is rekindled? This paper is an attempt to answer these questions via a review of the following six books:

- Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, eds., *Knowledge Representation*, Special Issues of *Artificial Intelligence: An International Journal*, Cambridge, MA: MIT Press, 1992, 408 pp., $29.00 (paper), ISBN 0-262-52168-7.

*We are indebted to Stevan Harnad, Pat Hayes, Drew McDermott and Ron Noel for helpful conversations about some of the issues discussed herein.

- Clark Glymour, *Thinking Things Through: An Introduction to Philosophical Issues and Achievements*, Cambridge, MA: MIT Press, 1992, xi + 382 pp., $35.00 (cloth), ISBN 0-262-07141-X.

- J. E. Hayes, D. Michie, and E. Tyugu, eds., *Machine Intelligence 12: Towards an Automated Logic of Human Thought*, Oxford: Oxford University Press, 1991, x + 342 pp. $120.00 (cloth), ISBN 0-19-853823-5.

- Steven H. Kim, *Knowledge Systems Through PROLOG: An Introduction*, New York: Oxford University Press, 1991, xvi + 341 pp., $29.95 (paper), ISBN 0-19-507241-3.

- André Thayse, ed., *From Modal Logic to Deductive Data Bases: Introducing A Logic Based Approach to Artificial Intelligence*, Chichester, U.K.: John Wiley & Sons, 1989, xxiv + 380 pp., £25.00 (paper), ISBN 0-471-92345-1.

- André Thayse, ed., *From Natural Language Processing to Logic for Expert Systems: A Logic Based Approach to Artificial Intelligence*, Chichester, U.K.: John Wiley & Sons, 1991, xx + 535 pp., £29.95 (paper), ISBN 0-471-92431-8.

Encapsulated, our answer is that logic and AI, despite tabloidish reports to the contrary, still enjoy matrimonial bliss—and they had better continue to if the future is to bring on stage both robots able to match us and scientists able to explain how they do so.

One side-effect of the investigation undergirding our answer is a taxonomy for parsing logicist AI and, possibly, the start of a taxonomy for parsing connectionist AI as well. We also intend the following to be useable as an introduction to logicist AI (for readers with a modicum of previous exposure to logic and computer science), as an antidote to caricatures of logicist AI,[1] and, to some degree, as a rallying cry for the logicist approach to understanding and replicating (or at least simulating) the human mind.

We should note at the outset that the marriage in question may not be monogamous. The connectionist would claim that AI has been polygamous from the outset, that AI was long ago married to non-logical (= sub-symbolic) architectures and approaches (yielding CAI, for connectionist AI; pronounced so as to rhyme with 'fly'), that a separation took place (courtesy, so the tale goes, of Minsky and Papert), and that the passion returned starting in the early 1980s. The CAInik would also claim, no doubt, that this marriage will eventuate in intelligent offspring of a type beyond the reach of LAI (pronounced—gleefully, by the connectionist—as 'lie'), the logic and AI couple. The second of these claims, for the record, is one we reject, but we make no effort to defend our rejection herein.[2]

## 2 A Taxonomy for Analyzing LAI

Modern first-order logic (denoted, hereafter, following tradition, by '$\mathcal{L}_I$') is the cornerstone of any respectable exposition of LAI; we give here a quick review of it in accordance with Glymour's Chapter 6 ("Symbolic Logic") and Thayse's 1989, Chapter 1 ("Languages and Logic"):[3]

Given an **alphabet** (of **variables** $x, y, \ldots$; **constants** $c_1, c_2, \ldots$; $n$-ary **relation symbols** $R, G, \ldots$; **functors** $f_1, f_2, \ldots$; **quantifiers** $\exists, \forall$; and the familiar **truth-functional connectives** $(\neg, \vee, \wedge, \rightarrow, \leftrightarrow)$, one uses standard **formation rules** (e.g., if $\phi$ and $\psi$ are well-formed formulas (wffs), then $\phi \wedge \psi$ is a wff as well) to build "atomic" formulas and then more complicated "molecular" formulas. ¿From sets of these formulas (say $\Phi$), given certain **rules of inference** (e.g., *modus ponens*: from $\phi$ and $\phi \rightarrow \psi$, infer $\psi$), individual formulas (say $\phi$) can be inferred; such a situation is expressed by meta-expressions like $\Phi \vdash \phi$. First-order logic includes a semantic side that systematically provides meaning for formulas involved. In first-order logic, formulas are said to be true (or false) on an **interpretation**, often written as $\mathcal{I} \models \phi$. (This is often read, "$\mathcal{I}$ satisfies $\phi$", or "$\mathcal{I}$ models (or is a model of) $\phi$".) For example, the formula $\forall x \exists y G y x$ might mean, on the standard interpretation $\mathcal{R}$ for arithmetic, that for every natural number $n$, there is a natural number $m$ such that $m > n$. In this case, the domain of $\mathcal{R}$ is $\mathbf{N}$, the natural numbers, and $G$ is the binary relation $> \subset \mathbf{N} \times \mathbf{N}$; i.e., $>$ is a set of ordered pairs $(i, j)$, where $i, j \in \mathbf{N}$ and $i$ is greater than $j$. Some formulas of $\mathcal{L}_I$ are **valid**, in the sense that they are true on *all* interpretations. For example, any formula $\delta$ of the form $\forall x(\phi \wedge \psi) \rightarrow (\forall x \phi) \wedge (\forall x \psi)$ is valid; such a fact is customarily written $\models \delta$. A formula that is true on at least one interpretation is **satisfiable**.

It is easy enough to generalize from this account to the broader concept of a **logical system**, and doing so is crucial for understanding LAI and the six books under consideration.[4] A logical system includes an **alphabet** (the symbols from which well-formed formulas are to be built); a **grammar** specifying how wffs are to be generated; a **semantic relation**, a binary relation (e.g., $\models$ as presented in the previous paragraph) holding between interpretations (or other structures designed to formally model aspects of "reality") and formulas; and a **proof theory** that makes precise how reasoning in the system is to proceed.

It is also usually important that a logical system have associated with it a **metatheory**, which would address questions such as whether the system in question is sound, complete, decidable, and so on. Such metaproperties are determined by bringing mathematical tools to bear on the system in question.

To anchor things a bit, consider first a system even simpler than first-order logic, viz., the propositional calculus, $\mathcal{L}_{PC}$, a familiar and simple logical system, and one introduced in solid fashion in Thayse 1989.[5] $\mathcal{L}_{PC}$ has for an alphabet propositional variables $p_1, p_2, \ldots$, and the truth-functional connectives; its grammar is the grammar of $\mathcal{L}_I$ without rules covering the quantifiers; its semantics are based on truth-tables (so that, e.g., $\models \phi$ holds iff $\phi$ is true on every

truth-value assignment in an exhaustive truth-table for $\phi$); and the proof-theory of $\mathcal{L}_{PC}$ can be given by familiar natural-deduction rules (e.g., *modus ponens*). Another logical system, this one "beyond" $\mathcal{L}_I$, is second-order logic, $\mathcal{L}_{II}$, which adds **relation variables** $X, Y, Z, \ldots$, to the alphabet of $\mathcal{L}_I$, which in turn allows (via the associated grammar) for formulas expressing such things as Leibniz's Law (two things are identical iff they have precisely the same properties):

$$\forall x \forall y (x = y \leftrightarrow \forall X (Xx \leftrightarrow Xy)).$$

Of course, logical systems are mathematical creatures. How does LAI end up with working programs? Suppose that we want to implement a logical system able to play the role of a guidance counselor in advising a high school student about which colleges to apply to. And suppose that we want to essentially work in $\mathcal{L}_I$. The first thing we need to do is create an ontology of the domain in question; accordingly, we may posit categories that include SATs, small liberal-arts colleges, national universities, grade-point averages, etc. All of these things must be representable by symbols from, and strings over, an alphabet of our devising (though part of the alphabet is unalterable; e.g., we know that we have the quantifiers). To focus the situation, suppose that we want a rule in such a system that says "If a student has low SATs and a low GPA, then none of the top twenty-five national universities ought to be applied to by this student." Assume that we have the following interpreted predicates: $Sx$ iff $x$ is a student, $L_s x$ for $x$ has low SATs, $L_g x$ for $x$ has a low GPA, $Tx$ for $x$ is a top-twenty-five national university, and $Axy$ for $x$ ought to apply to $y$. Then the rule in question, in first-order logic, becomes

$$\forall x \forall y [(Sx \wedge L_s x \wedge L_g x \wedge Ty) \rightarrow \neg Axy]. \tag{1}$$

Let's suppose, in addition, that Steve is a student denoted by the constant $s$ in the system and that he, alas, has low SATs and a low GPA. Assume also that $v$ is a constant denoting Vanderbilt University (which happens to be a top twenty-five national university according the *U. S. News & World Report*'s annual rankings). These facts are represented in the system by

$$Ss \wedge L_s s \wedge L_g s \tag{2}$$

and

$$Tv. \tag{3}$$

Our implemented logical system, based as it is on first-order logic, can verify

$$\{(1), (2), (3)\} \vdash \neg Asv;$$

that is, it can deduce that Steve ought not to apply to Vanderbilt.

At this point, we have identified, with respect to LAI, the following four fundamental categories: historical, philosophical, and mathematical foundations;

| Book | Foundations | System Specification | Metatheory | Implementation |
|---|---|---|---|---|
| Brachman et al. | .25 | .75 | 1 | .25 |
| Glymour | 1 | .25 | .25 | 0 |
| Hayes et al. | 0 | .5 | .5 | .5 |
| Kim | .25 | .25 | 0 | 1 |
| Thayse 1989 | .5 | 1 | 1 | 0 |
| Thayse 1991 | .5 | 1 | 1 | .25 |

Table 1: Breakdown of Coverage under Four Fundamental Categories

creation and specification of logical systems themselves; metatheory; and ways of getting to an implementation. Given these categories, our books fall out as indicated in Table 1, where we have a range from 0 for "no coverage" to 1 for "much coverage".

The remainder of the paper consists of selected discussion, via our six books, of these four categories, and then one final, brief section centered around a thought-experiment designed to display one of LAI's matchless virtues.

# 3 Foundations

## 3.1 Historical roots

When it comes to the history of AI, there may be no more informative book, ounce for ounce, than Glymour; certainly it would be difficult to find one stocked with as much cocktail-party ammunition. What thinker first articulated a serious (= genuinely rigorous) version of the AIish doctrine that the mind is a computational system? Did you answer "Hobbes" or "Leibniz"? Those, as Glymour explains, are not indefensible answers. But, as Glymour informs us, it was really Carnap, via his 1929 *The Logical Construction of the World*, who first expressed, in a form suitable, in principle, for capture in (say) Lisp, the vision that the mind is a program. All of this, we grant, may be knowledge possessed by cognoscenti—but even they, we wager, will be surprised to learn that it was the colorful philosopher Ramon Lull who first planted the computationalist seeds that flowered into Leibniz's dream for a "calculus of thought" and Hobbes's conviction that thinking is mere "calculation". Lull? That's right. As we read in Glymour:

> [Lull] spent his time with games and pleasantries and is reputed to have made great efforts to seduce the wives of other courtiers. Accounts have it that after considerable effort to seduce a particular lady, she finally let him into her chambers and revealed a withered breast. Taking this as a sign from God, Lull gave up the life of a courtier and joined the Franciscan order. (p. 70.)

As Glymour explains, not long after his singular repentance, Lull had built certain *machines* that he believed could be used to demonstrate to Moslems the truth of Christianity. Hidden in these intricate devices (which were composed of disks with common spindles displaying letters standing for Latin words signifying divine attributes) is the notion that reasoning can be mechanical and that it can be based on representations more robust than that expressible in Aristotle's theory of the syllogism.

We should point out that Glymour's historical discussion revolves around a record of attempts to answer the question "What is a proof?"—a question that the likes of Aristotle, Leibniz, and Boole failed to answer. It was Frege, the man Glymour rightly says "stands to logic as Newton stands to physics" (p. 120), who finally cracked the puzzle; his work led directly to the orthodox answer, namely that a proof is that which can be formalized as a derivation in first-order logic.[6]

## 3.2 Searle's Chinese Room

Perhaps no foundational take on AI is complete without at least mention of John Searle's infamous Chinese Room Argument (CRA), and Glymour doesn't disappoint. In fact, Glymour gives a new objection to CRA, one based on temporal factors:

> An important aspect of many actions is *the time it takes*. A system that executes the very same computational procedures that I carry out in catching a baseball or reading a story or solving a problem but that takes considerably longer than I do to carry out those procedures *does not do what I do*. (p. 350.)

Glymour goes on to claim that what Searle imagines himself doing (e.g., looking up appropriate Chinese strings for output in response to certain Chinese strings as input) is something he can't do "fast enough to simulate a Chinese speaker" (p. 350). The problem, it seems to us, is that Searle could be given, say, an electronic rolodex in order to accelerate the time needed for giving feedback. Since, as Glymour would apparently agree, a rolodex (and the like) surely doesn't carry *bona fide* understanding with it into the Chinese Room, Glymour's objection would seem to be easily surmounted. Interestingly, Glymour does intimate problems for AI that he believes may *be* fatal. One of these, left here in mere nutshell form, is that cognition may require continuous, analog information-processing beyond the reach of Turing machines, which are traditionally taken by AIniks to define the limits of computation *and* cognition. (On this issue, see Siegelman 1995.) This aspect of Glymour is of course relevant to the LAI–CAI clash, since connectionists often see themselves as working with continuous, analog systems.

# 4 System Specification

Most of Thayse 1989 and Thayse 1991, and a good deal of Brachman et al. are devoted to specifying logical systems central to AI. After $\mathcal{L}_{PC}$ and $\mathcal{L}_I$ are established (and we assume they now are for our readers, by virtue of the relevant material presented above), the standard route is to move beyond them by describing **propositional modal logic** (which we'll denote by $\mathcal{L}_{PML}$), pausing to consider what this new logical system might be good for, and then proceeding to richer logical systems. This route is competently followed in Thayse 1989 and Thayse 1991 essentially as follows.

## 4.1 Propositional modal logic

$\mathcal{L}_{PML}$, syntactically, is simply the propositional calculus with the addition of the two modal operators $\Diamond$ and $\Box$, which will be familiar to many readers. With the two new operators, of course, come newly admissible wffs, by way of the following straightforward formation rule:

If $\phi$ is a wff, then so are $\Diamond\phi$ and $\Box\phi$.

At this point, we have the alphabet and grammar for $\mathcal{L}_{PML}$. This system's semantical side is founded upon the notion of a Kripkean frame $\mathcal{F} = (W, R)$, where $W$ is simply a non-empty set (of **points** as Thayse 1989 says, or of **possible worlds** as it is often said), and $R$ is a binary relation on $W$; i.e., $R \subseteq W \times W$. $R$ is traditionally called the **accessibility relation**. Now, let $P$ contain all the atomic wffs generable from the logic's alphabet and grammar. We define a function $V$ from $P$ to $2^W$; i.e., $V$ assigns, to every atomic wff, the set of points at which it's true. Now, a model $\mathcal{M}$ (the analogue to 'interpretation' in our exposition of first-order logic) is simply a pair $(\mathcal{F}, V)$, and we can move on to define the key locution 'formula $\phi$ is true at point $w$ in model $\mathcal{M}$', written

$$\mathcal{M} \models_w \phi.$$

The definition of this phrase for the truth-functional connectives parallels the account for first-order logic (e.g., $\mathcal{M} \models_w \neg\phi$ iff it's not true that $\mathcal{M} \models_w \phi$). The important new information is that which semantically characterizes our two new modal operators, viz.,

- $\mathcal{M} \models_w \Box\phi$ iff for all $w'$ such that $wRw'$, $\mathcal{M} \models_{w'} \phi$.
- $\mathcal{M} \models_w \Diamond\phi$ iff there is a $w'$ such that $wRw'$, where $\mathcal{M} \models_{w'} \phi$.

But what, many of our readers are no doubt asking, do $\Box$ and $\Diamond$ *mean*? What are they good for? One way to answer such questions is to read $\Box$ as 'knows'. (An extensive catalogue of useful readings of the two operators is provided in Thayse 1989.) Such a reading very quickly allows for some rather tricky ratiocination to be captured in the logic. For example, Thayse 1989 has us consider the famous "wise man puzzle", which can be put as follows:

> A king, wishing to know which of his three advisers is the wisest, paints a white spot on each of their foreheads, tells them the spots are black or white and that there is at least one white spot, and asks them to tell him the color of their own spots. After a time the first wise man says, "I do not know whether I have a white spot." The second, hearing this, also says he does not know. The third (truly!) wise man then responds, "My spot must be white."

Thayse 1989 shows that the third wise man is indeed correct, by formalizing his formidable reasoning. The formalization (which is only given for the two-man version of the puzzle) starts with three propositions, viz.,

1. $A$ knows that if $A$ doesn't have a white spot, $B$ will know that $A$ doesn't have a white spot.
2. $A$ knows that $B$ knows that either $A$ or $B$ has a white spot.
3. $A$ knows that $B$ doesn't know whether or not $B$ has a white spot.

and proceeds to bring to bear on them the machinery of $\mathcal{L}_{PML}$.[7]

## 4.2 Temporal logic

A straightforward extension of $\mathcal{L}_{PML}$ leads to what Thayse 1989 calls **linear-time temporal logic**, which is not only useful in AI for modeling domains that include phenomena that vary with time (and what robust domain fails to include such phenomena?) but is also a tool for verifying the behavior of computer programs that have nothing to do with AI. The extension comes via two new operators, $\circ$ and $\mu$; this yields a total of four operators now, which are read as follows:

- $\circ$ – at the next time
- $\Box$ – always
- $\Diamond$ – eventually
- $\mu$ – until

A semantical account starts with a **temporal** frame $\mathcal{F} = (S, R)$, where $S$ is an at-most-countable, non-empty set of states, and $R$ is a binary relation on $S$ satisfying a condition that makes it a *functional* relation, viz.,

$$\forall t \exists! t' (tRt').$$

(The quantifier $\exists! x$ essentially says 'there is exactly one $x$ such that'.) Now, using $P$ again as the set of all atomic wffs from the propositional calculus, define a function $I$ analogously to $V$ from above; specifically, let $I$ be a mapping from

$S \times P$ to the truth-values 'true' and 'false', $\{\mathbf{T}, \mathbf{F}\}$. ($I$ tells us whether or not a given atomic formula $p$ is true at a given state $s$.) A **temporal interpretation** $\mathcal{I}$ is then a pair $(\mathcal{F}, I)$ that can give precise "meaning" to expressions in the logic, as follows. For the propositional case, a conjunction $p \wedge q$ is true at a state $s$ iff both $p$ and $q$ are true at $s$. This is written as $\mathcal{I}(s, \phi \wedge \psi)$ iff $\mathcal{I}(s, \phi)$ and $\mathcal{I}(s, \psi)$. For the interesting cases, the operators, we let $R^i(s)$ denote the $(i+1)$st member of the sequence

$$s, R(s), R(R(s)), \ldots.$$

Then, for example,

- $\mathcal{I}(s, \circ\phi) = \mathbf{T}$ iff $\mathcal{I}(R(s), \phi) = \mathbf{T}$, and
- $\mathcal{I}(s, \Box\phi) = \mathbf{T}$ iff $\mathcal{I}(R^i(s), \phi) = \mathbf{T}$, $\forall i \geq 0$.

As Thayse 1989 shows, formulas that, intuitively, we would want to come out valid on this scheme—e.g., $\circ(\phi \rightarrow \psi) \rightarrow (\circ\phi \rightarrow \circ\psi)$, a counterpart to axiom **K** in $\mathcal{L}_{PML}$—*do* come out valid, formally speaking. Thayse 1989 and Thayse 1991 also consider, in informative detail, how to go about proving things in linear-time temporal logic (and in enhancements of it). We return to some of the metatheory for this logic below.

We should mention that temporal logic is by no means the only tool used in AI for enabling reasoning about time and change. In fact, AI has often tended to refer to the entities posited in temporal logic (e.g., events, possible worlds, or histories) directly in modifications of $\mathcal{L}_I$. The earliest such formalism was the **situation calculus**, invented by John McCarthy (1968); a good presentation of this formalism, modernized from the form it had when first presented, can be found in Genesereth & Nilsson 1987. The paper "Nonmonotonic Reasoning in the Framework of the Situation Calculus", by Andrew Baker, found in the Brachman et al. volume, provides a quick but serviceable introduction to the situation calculus at work. Another well-known approach in AI to reasoning about time and change comes by way of James Allen's "interval approach" (1984). Rina Dechter, Itay Meiri, and Judea Pearl's "Temporal Constraint Networks", in Brachman et al. shows another (advanced) approach to time and action, one based on networks; the authors compare their approach to Allen's.

When temporal logic is deployed to systematize reasoning about the behavior of computer programs, it is often recast as **dynamic logic**, wherein the operators we have identified are associated with commands in a programming language. For a short but elegant description of dynamic logic as reinterpretation of temporal logic, see Thayse 1989 itself.

## 4.3 Defeasible logic

The next cluster of logical systems we discuss fall under **defeasible logic** (or revisable, or nonmonotonic, ... logic), an area that has for a long while justifiably received the attention of many LAIniks. Such systems mark attempts

to circumvent **monotonicity**, a property that, put in terms of $\mathcal{L}_I$, amounts to the fact that if $\phi$ is provable from $\Phi$, then adding new information to $\Phi$, no matter what that information is, never invalidates $\Phi \vdash \phi$. Clearly, our everyday reasoning isn't monotonic. Nearly all readers, for example, will at one time or another have affirmed the proposition that birds can fly. Expressed in straight first-order logic, this fact could be captured by

$$\forall x(Bx \rightarrow Fx). \tag{4}$$

But such readers will also have confronted the fact that ostriches *can't* fly. Clearly, there is a need to revise what had earlier been believed. But, just as clearly, it's very implausible that humans, in everyday reasoning, employ modifications of (4) like

$$\forall x((Bx \wedge \neg Ox) \rightarrow Fx),$$

because, for starters, there are an infinite number of exceptions to (4). The solution, in general, would seem to be that we use a system that allows us to make *defeasible* or *revisable* inferences. The challenge is to find and implement a correlate to this system.

There are a number of different families of attacks on the revisable reasoning challenge, among which are the following.

- Reiter's default logic (1980). The key idea here is domain-dependent inference rules – "defaults" – which permit rules with exceptions to be expressed without the exceptions having to be enumerated.
- Drew McDermott's non-monotonic modal logic (1982), built upon the predicate modal logic and designed to provide a domain-independent proof theory allowing for the characterization of maximal sets of consistent assertions that can be inferred from a knowledge base.
- Robert Moore's autoepistemic logic (1985), designed to model the introspective ability of an ideal agent whose beliefs can be revised in light of this introspection.
- Minimization techniques, which include the **closed-world assumption** (treated briefly below in §6) and **circumscription**, a rather tricky technique introduced by McCarthy (1980) and refined by, among others, Vladimir Lifschitz (1987).
- Oscar, a theory of defeasible reasoning devised by Pollock (1995), which is somewhat similar to default logic, but which, in our opinion, is superior (technically and epistemologically) to all other approaches to the problem. ('Oscar' is the name of the "artilect" who reasons using Pollock's theory.)

Chapter 5 of Thayse 1989, "Formalization of Revisable Reasoning", provides what is in our opinion one of the best introductions anywhere to minimization techniques. The treatment explains the intuitions behind minimization techniques and then proceeds to formalize the approach, gradually moving from the simplest incarnation of minimization techniques to circumscription.

Though Drew McDermott once claimed that only two people on the planet understood circumscription, the basic idea behind the technique, as Thayse 1989 shows, is really quite simple: Suppose some agent has knowledge represented by a set $\Phi$ of formulas in $\mathcal{L}_I$. We partition the relation symbols, or predicates, deployed in $\Phi$ into three sets, $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$. The predicates in $\mathcal{P}_1$ contain those to be circumscribed, or "minimized"; the predicates in $\mathcal{P}_2$ are the ones whose extensions (= the things in the domain picked out by the predicates) are permitted to change during the circumscription process; and $\mathcal{P}_3$ holds all remaining predicates. Now, let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two interpretations that model $\Phi$. (An interpretation $\mathcal{I}$ models a set of formulas $\Phi$ just in case it models every formula in the set.) $\mathcal{I}_1$ is said to be **inferior** to $\mathcal{I}_2$ exactly when the following three conditions are satisfied:

1. $\mathcal{I}_1$ and $\mathcal{I}_2$ have the same domain and interpret function symbols and variables in exactly the same way.

2. $\mathcal{I}_1$ and $\mathcal{I}_2$ interpret the predicates in $\mathcal{P}_3$ in exactly the same way.

3. For every predicate $R$ in the set $\mathcal{P}_1$ of circumscribed predicates, the extension of $R$ in $\mathcal{I}_1$ is a subset of the extension of $R$ in $\mathcal{I}_2$.

These three conditions can be understood intuitively as follows: The two interpretations in question, as it is said in Thayse 1989, interpret our agents knowledge, $\Phi$, "almost identically". But interpretation $\mathcal{I}_1$ interprets the predicates in $\mathcal{P}_1$ as true "less often" than interpretation $\mathcal{I}_2$ does. The key interpretations are those that model $\Phi$ and are minimal with respect to the ordering relation produced by these three conditions. A formula can be inferred from a circumscribed knowledge base when it is true in all minimal models. The hope is that the formulas that can be inferred will be precisely those that are desirable in cases like the ostrich one we presented above. Thayse 1989 shows, in fact, that

$$\forall x((Bx \wedge \neg Ox) \rightarrow Fx)$$

can be deduced in the ostrich example. The demonstration follows the traditional proof-theoretic route through circumscription: It's shown that if a certain second-order formula $\gamma$, (i.e., a formula from the logical system $\mathcal{L}_{II}$ described at the outset of this paper) is added to the knowledge in this case, the desired deduction can be performed. Thayse 1989 then explain the problems plaguing circumscription, at least in its traditional form (e.g., partitioning the predicates seems to require knowledge of the desired outcome).

Readers who assimilate the material we presented earlier in §2 and then the presentation of circumscription in Thayse 1989 will be able to proceed to the many papers in Brachman et al. that relate to revisable reasoning. We mention a number of these papers below under the "Metatheory" category of our four-fold breakdown of LAI.

## 4.4 Other logical systems

There are a number of other logical systems competently specified in the six books in question. Some of these (e.g., the sorted logic featured in Alan Frisch's "The Substitutional Framework for Sorted Deduction", in Brachman et al. which marries classical logic to semantic networks) serve narrow but important purposes. Others are systems that anyone seriously interested in LAI (and related mathematical, linguistic, and philosophical matters) ought to be familiar with. One family of such "required reading" systems are those formed in the attempt to genuinely come to grips with the expressive power of natural language. Invariably, members of this family build upon the logical systems already described in this paper; the perfect example is Montague semantics, and perhaps the perfect introduction to Richard Montague's work (and related matters) comes by way of Thayse 1989 and Thayse 1991.

Montague, in a series of three seminal papers (all of which are conveniently reprinted in Thomason 1974), initiated a program intended to formulate a syntax and semantics for natural language having the same rigor and precision as the syntax and semantics for formal languages. The introduction to this program found in Thayse 1989 and Thayse 1991 is first rate. The Thayse 1989 part of this treatment starts with an extensional, type-theoretic logic; proceeds to consider Montague's intensional logic; pauses to engagingly discuss the historical, philosophical, and linguistic context of Montague's program; then presents for converting natural language to formal language; and, finally, briefly discusses some alternative approaches to reaching Montague's dream. Thayse 1991 is an articulate discussion of some of the limitations of Montague's approach (e.g., anaphora isn't accommodated) and some of the proposed solutions – solutions that show LAI making genuine progress.[8] We recommend that interested readers begin with the initial portion of the advanced material in Thayse 1991, which constitutes a quick introduction to Montague's logic; from there the reader can move back to the coverage in Thayse 1989, and then back to the advanced topics treated in Thayse 1991. We attempt now to give a sense of Montague's work.

Montague's approach borrows numerous elements from the logical systems visited above. For example, his logic is "multimodal", in that the logic combines operators for possibility and necessity with temporal operators:

- $\Box$ – necessarily
- $\Diamond$ – possibly

- [F] – always in the future
- $\langle$F$\rangle$ – sometimes in the future
- [P] – always in the past
- $\langle$P$\rangle$ – sometimes in the past

Belief and knowledge, which we formalized above using the two basic operators from modal logic, are formalized by Montague with a two-place predicate for each concept – e.g., $believe(a, e)$ – the first place occupied by a symbol referring to the agent, the second to an event. There is insufficient space here for even a compressed presentation of how the step-by-step process of Montague's translation scheme, when applied to natural language, yields a formal counterpart. But central aspects of the process are as follows: At the heart of the translation is an isomorphism between the analysis of an expression in natural language and its associated analysis in the intensional logic. This isomorphism comes by way of a **type-theoretic syntax** for the logic and a **translation function** that maps each syntactic category of natural language to a type of the intensional logic. Each natural-language expression $A$ of syntactic category $\mathcal{C}$ is assigned a logical translation $A'$ in the logic; and $f(\mathcal{C})$ is the logical type of category $\mathcal{C}$. The approach assumes that most sentences of natural language can be analyzed according to two different moods: *de re* and *de dicto*.[9] Two additional operators handle the *de re* and *de dicto* moods; these are the intension and extension operators – $\triangleleft$ and $\triangleright$, respectively the first indicating *de re*, the second *de dicto*. The final tool used in Montague's translation is the **lambda operator**. Our readers are doubtless familiar with such "set builder" statements as

$$\{x : x \text{ is a person able to beat Deep Blue}\},$$

that is, the set of all people able to beat Deep Blue, a famous chess-playing computer program. The $\lambda$-operator allows us to say similar things in the logic itself. For example, if the two-place relation $Wxy$ is true if and only if $x$ has written $y$, and if $u$ is a constant denoting Umberto Eco, then intuitively speaking the following represents the set of works that Eco has written.

$$\lambda x(W\ x\ f)$$

In order to perhaps whet your appetite for Montague's program, here is what all of this machinery produces in the logic for the English sentences "John examines and solves a problem" and "John will not talk", respectively:

$$exm'(\triangleleft\lambda Y[\exists x(prob'(x) \wedge \triangleright Y(x))])(j) \wedge solv'(\triangleleft\lambda Y[\exists x(prob'(x) \wedge \triangleright Y(x))])(j)$$

$$\langle F\rangle(\neg talk'(j)).$$

Those interested not only in Montague's work, but in the full range of proposals for how to genuinely deal with the formidable power of natural language (especially natural language's ability to allow reference to nonexistent objects), are encouraged to read Graeme Hirst's "Existence Assumptions in Knowledge Representation", in Brachman et al. (We strongly recommend that readers searching for answers to the problems lurking in this area also investigate SNePS (Shapiro & Rapaport 1987).)

# 5 Metatheory

In our taxonomy, we use the term 'metatheory' in a slightly more inclusive sense than is standard. For us, the term covers (i) theorems establishing that a logical system has (or lacks) certain mathematical properties (the customary sense), as well as (ii) general discussions whose vocabulary includes terms making reference to logical systems and their mathematical properties. (Metatheory in the sense of (ii) may correspond to what Jon Doyle has called "rational psychology" (1988), the attempt to "logicize" the mind.) The discussion in (ii), of course, often includes straight metatheory from (i).

## 5.1 Metatheory in the standard sense

The first interesting metatheoretical result that students of LAI are likely to assimilate is that which implies that $\mathcal{L}_I$ has the properties of both **soundness** and **completeness**.[10] The next significant metatheoretical result usually presented concerns **decidability**: First-order logic is not decidable; i.e., there is no algorithm for determining whether or not a first-order formula is valid and, hence, no algorithm for determining whether or not a given set $\Phi$ of first-order formulas can deductively produce a given formula $\phi$. (The propositional calculus, on the other hand, *is* decidable: There is an algorithm (which can simply construct a truth-table) that can tell if a wff of $\mathcal{L}_{PC}$ is true on all truth-value assignments.)

There are many results along these lines in the volumes under review. Our favorite cluster of such results concerns the decidability of the linear-time temporal logic we considered above. The decision procedure for this logic is a fascinating one that views formulas in the logic as infinitely long strings and then proceeds to classify such strings with help from a variant on finite-state automata – Büchi automata – which "accept" them.[11]

The key idea in moving from formulas of linear time temporal logic to their corresponding Büchi automata is that such formulas can be viewed as true on certain infinite histories. For example, the formula $\Diamond p$, recall, is true of every history in which $p$ is eventually true; so the Büchi automaton is the one seen in Figure 2. In this flow graph, acceptance happens only if the accept state (the double circle) is "hit" infinitely many times. But that cannot be

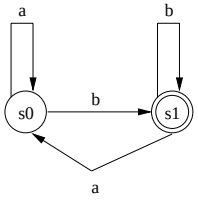



Figure 1: Büchi automaton accepting $(a^\star bb^\star)^\omega$.

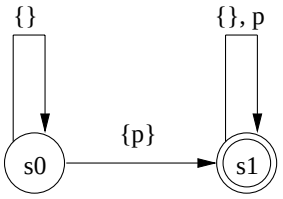



Figure 2: Büchi automaton for $\Diamond p$.

accomplished unless $p$ is hit at least once; i.e., acceptance happens only if $p$ is eventually hit, and then history moves on forever, with the question of whether $p$ remains true or becomes true again left as an open question. The algorithm for deciding whether or not a formula from linear-time temporal logic is satisfiable first generates from a formula $\phi$ the Büchi automaton $A_\phi$ corresponding to this formula, and then checks to see if there are strings accepted by $A_\phi$. If so, then $\phi$ is satisfiable; if not, $\phi$ is not satisfiable.[12]

## 5.2 Metatheory in a broader sense

A number of penetrating papers in Brachman et al. fall under our broader construal of metatheory:

Andrew Baker's "Non-monotonic Reasoning in the Framework of Situation Calculus" presents a modification of standard circumscription allowing this technique to solve the famous "Yale Shooting Problem".[13]

Doyle and Michael Wellman, in their fascinating "Impediments to Universal Preference-Based Default Theories", investigate the prospect of a formal theory of rational inference and preference that could unify the various families of approaches to revisable reasoning – and find that such unification is probably not possible. (This paper will be of special interest to philosophers: It ranges over topics as diverse as Pascal's wager and "mental societies", composite agents composed of individual selves.)

No discussion of the LAI approach would be complete without noting that this approach is in large part driven by certain gem-like problems – problems that, upon analysis, explode with inspiring and guiding challenges. A perfect

example of such a problem is the Lottery Paradox, discussed by David Poole in his "The Effect of Knowledge on Belief: Conditioning, Specificity and the Lottery Paradox in Default Reasoning". The paradox runs as follows:[14]

> Suppose you hold one ticket ($t_k$, for some $k \geq 1$) in a fair lottery consisting of 1 million tickets, and suppose it is known that one and only one ticket will win. Since the probability is only .000001 of $t_k$'s being drawn, it seems reasonable to believe that $t_k$ will not win. (After all, isn't it true that you believe that your head won't be mashed by a meteorite when you walk outside today? And don't you so believe because the probability that your head will be mashed is very low?) By the same reasoning, it seems reasonable to believe that $t_1$ will not win, that $t_2$ will not win, ..., and that $t_{1000000}$ will not win. Therefore, it is reasonable to believe
>
> $$\neg\exists t_i(t_i \text{ will win}).$$
>
> But we know that
>
> $$\exists t_i(t_i \text{ will win}).$$
>
> So we have an outright contradiction.

Poole does an excellent job of systematically laying out the various maneuvers taken by LAIniks in an attempt to resolve the Lottery Paradox, and the costs associated with these moves. (How would *you* design an artificial reasoner capable of escaping the paradox? Grappling with focussed metatheoretical questions such as this one are really, at bottom, the best way to truly understand the LAI approach.)[15]

# 6 Implementation

In this section, we briefly review the history of logic programming (LP), and then give an account of what we call **implemented LAI systems**. (For a detailed account of the history of LP, see Robinson 1992.) Armed with this history and this account, we proceed to explore the topic of implementation through Kim, Hayes et al., and Thayse 1991.

## 6.1 Historical roots of logic programming (LP)

Logic can be thought of as a proposed means to precisely communicate knowledge and its underlying justification. Aristotle, as Glymour eloquently explains, set a 2,300-year-old standard for what constituted a precise expression of, and justification for, knowledge, including the mathematical knowledge produced by Euclid. (Glymour provides a superlative treatment of Euclidean reasoning and Aristotle's theory of the syllogism.) As Glymour explains (in Chapter 4, "The

Laws of Thought"), the Aristotelian scheme started to genuinely change when, in the middle of the nineteenth century, George Boole attempted to realize Leibniz's conception of an "algebra of thought", but even Boole's proposals failed to account for even simple geometric and number-theoretic proofs of the sort given by Euclid. Everything changed, as we noted above, when Frege hit the scene. He laid the foundation for erecting $\mathcal{L}_I$ (and for a number of other things – e.g., aspects of modern economics).

For our purposes at the moment, it's important to note that the predicate calculus is the foundation of LP, because it provided, for the first time, a precise characterization of the concept of proof. Furthermore, proofs cast in $\mathcal{L}_I$ are themselves mathematical objects that can be formally and symbolically represented, analyzed, and manipulated by algorithms.

In 1934, Alfred Tarski provided a semantic theory for the predicate calculus that established a link between its syntax and the notion of truth in a given interpretation (see Tarski 1956). (Our presentation above of the semantics of first-order logic follows Tarski closely.) In 1936, Gerhard Gentzen showed that, if a predicate calculus proof for a sentence exists, it can be produced from the concepts in the sentence without the *creative* introduction of extraneous concepts (see Gentzen 1969).

Gentzen's results sparked the search for algorithms that can find proofs for sentences in predicate calculus – theorem-provers. This quest, inspired by theoretical advances in the 1930's, laid the foundation for modern LP.

The first attempts at theorem-proving algorithms, however, were exhaustive searches through the exponential space of possible proofs. Despite the power of the computers introduced in the 1950s, it was obvious that the exponential complexity of the proposed predicate-calculus proof procedures was overwhelming.

In the 1960s, a simpler form of predicate logic called **clausal logic** was introduced. This form proved very important for producing efficient computational procedures. Standard predicate logic is redundant in that there are often alternative ways to express the same thing: Two propositions may be syntactically distinct but semantically equivalent. While this might be an advantage for humans wanting to represent knowledge with predicate logic, it is a source of inefficiency for the automated procedures designed to manipulate these expressions in search of proofs. The simpler clausal logic led to different, more efficient approaches to manipulating proofs.

The 1960s also saw the abandonment of approaches using proof procedures based on human-oriented inference rules like *modus ponens*. These rules were supplanted with machine-oriented reasoning patterns that could take more productive steps in the production of a proof without getting bogged down with the conceptual details that a human reasoning process might require. In 1963, J. Alan Robinson invented a clause logic based on a single, very effective inference rule that he called **resolution**. Though not an intuitive inference rule, resolution allowed the proof procedure to take large steps toward a goal. Reso-

lution relied on a process called 'unification'. This process, based on work done by Jacques Herbrand, enabled the direct computation of clause instantiations, which eliminated the need for much of the detailed search required in the earlier theorem provers.

By 1963, largely due to Robinson, we had a resolution-based algorithm for producing clausal predicate-calculus proofs that was significantly more effective than any of its predecessors. In fact, it was effective enough to form the basis of fully functional programming systems based on logic.

The declarative approach to building AI systems based in logic was under scrutiny contemporaneously with the strides made in LP research. In the late 1960s, the AI community was boiling with a debate over whether knowledge is better represented for AI purposes declaratively or procedurally. The fate of LP as an accepted basis for the implementation side of AI was at stake during this heated controversy. In 1974, Robert Kowalski showed the equivalence of a Horn-clause representation and a procedural representation of knowledge. (A **clausal representation** is a – possibly empty – disjunction of literals, where a literal is a negated or unnegated predicate. A **Horn-clause representation** is a clausal representation restricted to contain at most one unnegated literal.) Implementation decisions regarding a commitment to procedural or declarative representations, after Kowalski's work, were best viewed as issues of programming style rather than of anything more substantive ... and the technological and political stage was set for the development and application of logic-programming systems. Refinements made on efficient implementation strategies – most notably by Robert Boyer and J Strother Moore (1972) – ultimately led to the popular and widely applicable LP language Prolog, which is at the heart of Kim, a volume discussed after our characterization of implemented LAI systems.

## 6.2 Implemented LAI systems

We say that an **implemented LAI system** is an information-processing artifact that instantiates some logical system $\mathcal{L}_x$ and that acquires information about its environment, derives meaningful conclusions from the acquired information, behaves on the basis of these conclusions so as to manipulate its environment (with some objective in mind), acquires new information, and returns to acquiring information regarding its environment, whereupon it continues in this cycle.[16] Given this general view, an implemented AI system, and the development and refinement of one, would seem to include five basic functions, viz.,

1. **Ontological Analysis**. An ontology of the domain is produced. Correspondence between this domain and formulas in the logical system is established.

2. **Knowledge Acquisition**. The system acquires input and categorizes that input with respect to the ontology. Today, this part of the implementation task is often done by "injection": Humans code in knowledge to the system in question.

3. **Derivation**. Meaningful conclusions are derived from the information produced by 2.

4. **Action**. Behavior informed by the conclusions from 3 is produced.

5. **Communication**. The system communicates knowledge, acquired data, and conclusions therefrom (and the derivations in question) to human users of the system and to other LAI systems. Sometimes, this communication is in part accomplished by the human designer simply inspecting the "innards" of the system.

For example, consider a robot tank, ROB, based upon the system $\mathcal{L}_I$. The result of the analysis function might be an ontology consisting of objects called "tanks", "helicopters", "cars", "people", "guns", "machine-guns", "soldiers", and "civilians." It might also assert that groups of tanks form things called "tank divisions" and that groups of soldiers form things called "infantry divisions". For the acquisition function, ROB might have visual sensors that it uses to find and distinguish the objects of the ontology – tanks, people, guns, etc. – from an actual battle scene. Incarnation of the derivation function might enable ROB to conclude that a person carrying a machine gun, identified by the acquisition function, is an enemy soldier and should be shot at, provided she is not flanked by a tank division. If ROB "sees" a person carrying a machine-gun, and various other things are true, he may act: He may shoot at this person. Finally, the communication function is responsible for conveying ROB's knowledge. ROB might produce a natural-language description of the battle scene for consumption by a field commander. This description may include possible military responses as well their justifications.

It seems to us a profitable exercise to attempt to classify the most powerful, state-of-the-art LAI systems by way of our five categories and our presentation above of logical systems and related machinery. Two such systems are OSCAR and CASSIE (in SNePS); readers are encouraged to test the analytic power of our scheme on them – keeping mind, of course, that the logical sytem at the heart of this advanced work may not coincide with a standard logical sytem.[17]

## 6.3 LAI in today's implementations

### 6.3.1 Kim

Kim attempts to unite important and related concepts in LAI, logic programming, and software engineering. On the surface, this is an exceedingly tall order – given that each of these fields is apparently based on large islands of its

own research and applications and that each is driven by its own (largely) self-contained theoretical and applied communities. This book, however, succeeds; the reason it does is twofold: One, its foundation is the concept of an implemented LAI system (as explicated above) – a system that interweaves AI, logic programming, and software engineering; two, Kim targets an introductory *undergraduate* course in computer programming, which allows the book to dodge genuine but rarefied distinctions between the fields to be united. In short, Kim succeeds in championing a logic-based approach to building implemented LAI systems (called **knowledge systems** in Kim).

This is not to say that Kim is flawless. Far from it. One problem is that Kim views logic programming, in the form of standard Prolog, as a programming system based on mathematical models of human thinking. Kim believes that the unique strengths of the LAI approach spring from its affinity to the thinking patterns of people as opposed to those of computers. Indeed, Kim holds that first-order logic was created to address flaws in human reasoning:

> Unfortunately, we humans are prone to error when it comes to complex problems in reasoning. For example, careful studies of legal documents – which are intended to be paragons of logical precision – often result in the exposure of internal inconsistencies. ... Mathematical logic ... was developed to address the problems of faulty reasoning and ambiguity in everyday language. The basic premise is that correct procedures for reasoning or deductive thinking can be formalized and used profitably, independently of the specific domain of discourse (Kim, p 10).

That correct procedures for reasoning can be formalized independent of specific domains is true and, we agree, helpful. But as most readers will know, there is a longstanding controversy about whether logic, at least in the form of $\mathcal{L}_I$, serves as an accurate model of the way humans in fact think. Even if logic provides an answer to the question "What is a proof?" – the tackling of which, as we've noted, Glymour chronicles – it doesn't follow that logic captures human thinking. Logic evolved out of attempts to meet the challenge of constructing an impeccable foundation for classical mathematics.[18] It was Montague, as Thayse 1989 nicely explains, who wanted in earnest to formalize natural language – and Montague knew well that first-order logic, of which Prolog is a proper fragment, would be insufficient for that formalization. Nonetheless, though Prolog isn't, as Kim opines, "based on human logic", it does, we agree, represent a powerful and appropriate tool for building AI systems, especially in the areas of natural language processing, robotics and expert systems. (It's certainly true that *some* aspects of human deliberate thinking are nicely formalized in Prolog, and that as a vehicle for smooth and rapid implementation it enjoys perhaps unrivaled power.)

Part I of Kim includes an introduction to and an historical view of AI from the perspective of implementation. Part II of the book, entitled "Pro-

log", presents this language as a vehicle for building implemented AI systems. The advantages of Prolog, as Kim explains, lie in its well-understood roots in $\mathcal{L}_I$, and, again, the ease with which the software engineer can map a problem domain into working software.

A particularly notable feature of Prolog (by our lights) is competently discussed in Chapter 11, wherein Kim treats Prolog's support for **meta programming**, a powerful notion underlying LP that views programs as mathematical objects that can themselves be analyzed, evaluated and modified. The meta-programming facility in Prolog allows a program to reason about other programs. Variables in Prolog can denote other clauses; predicates can be developed which describe other predicates. A simple example is the two-place predicate `arg`. The fact

```
arg(p,2)
```

expresses that the predicate `p` has two arguments.

Meta-programming plays a key role in our own research, which is in significant part devoted to engineering a literarily creative artificial agent known as BRUTUS (Bringsjord and Ferrucci, forthcoming). BRUTUS includes a formalization of the concept of betrayal, and reasons about this concept with help from our meta-programming. This programming technique facilitates the amalgamation of an object-level language with a portion of a meta-level language suitable for formalizing the derivability relation of the orginal object lanaguage.[19] Such amalgamation in the same system allows greater expressivity than the object language alone. Consider, for example, BRUTUS' reasoning about different agents' belief systems. Suppose, for example, that there are two individuals, Julius and Brutus, in the BRUTUS world. Each has their own beliefs about this world and about one another. Suppose that some sequence of actions take place in their "reality", the result of which is that Brutus stabs Julius. BRUTUS must determine if Brutus acted with malintent to murder Julius, or if he acted in self-defense. To determine this the system must inspect Brutus' belief system and determine if it was consistent for Brutus to believe that his life was threatened by Julius. Fleshing out this example helps communicate the power of meta-programming (and helps bring to life the technique as delineated in Kim and employed – as we shall see below – in Hayes et al. and Thayse 1991); we do so as follows.

First, consider the following formulation, which is a simplification of a sort BRUTUS in fact manages.

```
malintent(Accused,Victim) ←
    person(Accused),
    person(Victim),
    beliefs(Accused,ABeliefs),
    not(proof(ABeliefs,'threatened-by(Accused,Victim)'))).
```

The predicate `proof` represents the provability relation embodied in Prolog (i.e., resolution logic). It is a meta-logical predicate used to determine if the statement

```
threatened-by(Accused,Victim)
```

can be proven from the accused's beliefs using resolution logic.

This mixed formulation of object and meta-level constructs allows the AI system to treat agents' thought processes independently from each other and from the system's beliefs. This is key for reasoning about the intent of an agent independent of reality. For example, Julius may have been conspiring to destroy Brutus, but unless this fact can be realized from Brutus' beliefs, this truth would not exonerate Brutus in the least.

Consider, along these lines, a pure object-level formulation:

```
malintent(Accused,Victim) ←

    person(Accused),
    person(Victim),
    not(threatened-by(Accused,Victim)).
```

If it were the case that Julius conspired to kill Brutus, but Brutus was entirely unaware of this evil, then the latter formulation would fail to correctly determine Brutus' intent.

In sum, meta-programming as an enabler for amalgamating object-level and meta-level language is an important representation and implementation tool for LP-based LAI systems, and we refer readers interested in this topic first to Kim, then to (a key paper discussed below in) Hayes et al. and then to the logical system EPL, which is specified and applied in "Bulding Expert Systems", Chapter 3 of Thayse 1991.

Chapter 13 of Kim is notable as well. This chapter is a demonstration of how Prolog provides a logic-based facility for parsing and attributing semantics to statements given in natural language. We encourage those interested in natural language processing to consider how the theoretical treatment of natural language in Thayse 1989 and Thayse 1991, touched upon above, can be realized with the sort of techniques discussed in this part of Kim.

Part III, entitled Knowledge Structures and Systems, discusses techniques in representing, structuring and organizing knowledge and developing software systems. It concentrates on modular architectures for knowledge systems in industry. Chapter 14 is worth particular mention. It demonstrates that knowledge structuring techniques like frames, semantic networks, hierarchies and rules have been central in engineering implemented LAI systems. Although there is certainly no in-depth study to be found here, Kim competently discusses how Prolog can enable a formal and mathematical approach to realizing these structures in software.

Part IV presents three implemented AI systems: A robot simulator, a calculus aid and an investment counselor. The treatment is appropriate for an undergraduate course in programming. Though not terribly robust, these examples do make the point that LAI pursued through Prolog offers practical engineering advantages.

### 6.3.2 Hayes

Hayes et al. is composed of a set of 20 papers covering these five areas:

1. Mechanics of Knoweldge Processing
2. Inductive Formation of Programs and Descriptions
3. Optimality and Error in Learning Systems
4. Qualitative Representations of Knowledge
5. Applications and Models of Knowledge Aquisition.

These papers cover advanced topics in AI research, and a comprehensive treatment of them would be encyclopedic. However, one paper in particular, in our opinion, is worth looking at in some detail – because it nicely captures the spirit and, to an appreciable degree, the technical orientation of Hayes et al. and because it also interweaves the topics of defeasible reasoning, LP, implementation, metaprogramming and foundational issues (e.g., learning) central to the LAI-CAI clash. The paper in question is Chapter 8, "Non-monotonic Learning", by Bain and Mugleton. It introduces a non-monotonic formalism called **closed world specialization**. This formalism's implementation in Prolog results in an incremental learning system that revises its beliefs upon receiving new examples.

One way to learn from experience is to specialize "over-general" beliefs. Consider yet again the belief that all birds fly; it might be represented in LP by the following rule.

```
flies(X) ← bird(X)
```

If the new conjunctive belief that an emu is a bird and emus can't fly were introduced, the above rule would no longer be true. The system would have to learn; it would have to specialize the rule so as to accurately represent the new belief. Incremental learning systems may generalize their belief sets in the face of a new example that these sets don't entail, or such systems may specialize their belief set if it is contradicted by a new example. Previous approaches tend to over-specialize in an attempt to adapt to contradictory examples. It's not hard to make this clear:

Consider the case where a system's initial belief set, $\Phi$, includes the following statements.

```
{bird(eagle), bird(emu), flies(X) ← bird(X)}
```

¿From $\Phi$ the system (an LP system here) can conclude that

$$\texttt{flies(eagle)} \text{ and } \texttt{flies(emu)}.$$

Since it is not true that `flies(emu)`, the system must revise its beliefs so that

$$\Phi \nvdash \texttt{flies(emu)}.$$

One way is to delete the clause `flies(X)` $\leftarrow$ `bird(X)` and produce the new belief set

$$\Phi' = \{\texttt{bird(eagle), bird(emu)}\}.$$

Since $\Phi'$ is incomplete with respect to `flies(eagle)`, this set can be generalized to produce

$$\Phi'' = \{\texttt{bird(eagle), flies(eagle), bird(emu)}\} \ .$$

The set $\Phi''$, however, is an over-specialization. The fact `flies(sparrow)` is unfortunately not deducible from

$$\texttt{bird(sparrow)} \cup \Phi''.$$

To solve the problem of over-specialization in incremental learning systems, Bain and Mugleton define the most-general-correct-specialization (MGCS) and develop a resolution-based method for constructing an MGCS from an incorrect clausal theory (an intial set of beliefs plus the new contradictory example). Using meta-programming (see our discussion of this topic above) the authors demonstrate how this method can be implemented in Prolog.

The method exploits the LP notion of negation as failure. A common (but limited) approach to non-monotonic reasoning is based on the "Closed World Assumption" (CWA) inference rule. This rule states that if a statement $p$ is not a logical consequence of a theory, then infer $\neg p$. Prolog implements this inference by interpreting failure as negation. If Prolog fails to prove $p$, where $p$ is a ground atom (a fact with no variables), then Prolog infers $\neg p$.

An example demonstrates the results of this approach. Consider again the initial belief set $\Phi$ and the fact that `flies(emu)` is not true. The proposed method produces the revised belief set $\Phi$ composed of

- `flies(X)` $\leftarrow$ `bird(X)`
- `not(flightless(X))`
- `bird(eagle), bird(emu), flightless(emu).`

The method introduces a new predicate; in this case `flightless`. Its negation (by failure), `not(flightless(X))`, is added to the conditions for `flies(X)` and the fact `flightless(emu)` is added to the belief set. The result is the revised

belief set $\Phi$ which doesn't incorrectly entail `flies(emu)`; and `flies(sparrow)` can be correctly proved from $\Phi \cup \{$`bird(sparrow)`$\}$.

Several other papers in the Hayes et al. volume (e.g., "Inductive Formation of Programs and Descriptions", yet another case of work that exploits on meta-programming) make considerable headway against the common but mistaken notion that learning is outside the reach of LAI but is the forté CAI.

### 6.3.3 Thayse 1991

Thayse 1991, visited above under the topic of "Metatheory", does an excellent job of compiling a number of LAI research thrusts in a variety of classic problem domains (natural language processing, expert systems, machine learning, requirements engineering, etc.), where the thrusts are often shown to naturally lend themselves to Prolog or other LP systems for implementation.

In Chapter 2, Jean-Louis Binot argue that effective natural language processing requires the structure of an intermediate logical form. The intermediate form is used for semantic interpretation of natural language constructs and naturally lends itself to Prolog as an implementation formalism.

Chapter 3, by Albert Bruffaerts and Eric Henin, addresses the development of knowledge bases for expert systems. They wrap together in canonical fashion the concepts of logical system and implemented AI system:

> In a logical framework, a knowledge base is specified as a logical theory whose standard model consists of the expert's perception of the application domain. The axioms of the theory are a formal expression of the knowledge of the application domain. The conclusions drawn from the knowledge base correspond to the logical consequence of the theory. Given a well chosen set of inference rules, which is sound and complete, answers to queries can be identified to theorems derived from the axioms through a formal deduction process. The proof of the theorem defines the nature of what could be a justification of a query answer (Thayse 1991, p. 119).

Chapter 3 also outlines the classic case for using LP to represent knowledge and handle (via resolution) derivation. The case, in short, runs as follows. Declarative representations are important because they capture the factual information about the domain, as precise logical statements independent of the control strategy used to execute the computation on a particular computer architecture. Consequently, LP, as a means for declarative programming, allows the AI practitioner to directly and declaratively represent the logic of the system independently of control and performance concerns. LP provides means for separating concerns about "what" knowledge the system has without worrying about "how" it can be derived. Prolog does, however, allow access to certain procedural control mechanisms for fine-tuning the performance of the underlying inference engine.

This case for LP may be compelling in the abstract, but if LP is wed to Prolog, some problems, as Bruffaerts and Henin point out in Chapter 3, arise. They explain that Prolog, while a strong fundamental technology, does not provide sufficient expressive power for supporting a purely declarative approach that insulates the developer from some of the procedural enticements accessible in pure Prolog. They propose the EPL language as an extension to Prolog that supports:

1. The use of predicate names as terms so that knowledge and meta-knowledge can be treated equally in the same formalism.

2. Explicit universal and existential quantifiers in rules and queries.

3. Lambda terms for representing parameterized formulae.

4. A functional notation for binary relations that represent mappings to improve the structuring and readability of the rules.

EPL is described in detail. As a credit to Prolog's meta-programming facility, this language is implemented in Prolog. The EPL language is used in the remaining portions of the chapter as a representation and implementation formalism to address plausible reasoning, class hierarchies, inheritance and the automatic generation of explanations.

The application of logical formalisms to real world problems, as we saw above when motivating revisable reasoning, requires some mechanism for efficiently handling change. Idealized domains like those dealt with in classical mathematics are in some sense immutable: the axioms of arithmetic, for example, don't change over time. The "real world", of course, isn't so clean: it presents problems requiring a mechanism for efficiently realizing the effect that modifying the starting assumptions has on derived conclusions. Chapter 4, by Jean- Pierre Mueller and Dominique Snyers, provides an excellent discussion of one of the practical sides of revisable reasoning, namely, reason maintenance systems.

Chapter 5, by Axel van Lamsweerde, "Learning Machine Learning", presents a logic-based treatment of learning systems. This chapter provides an important introductory perspective on learning for those convinced that learning can only be productively tackled by CAI. Our suggestion is that those interested in learning about learning within LAI start with this chapter, and then move on to the "Non-monotonic Learning" piece (in Hayes et al.) we discussed above.

Chapter 6, by Eric Duboies, Jacques Hagelstein and Adre Rifaut, is similar in approach to Chapter 3. The authors address the domain of requirements engineering of computer systems. They offer an excellent justification for their commitment to LAI and develop a logical system to tackle the problem. The system is called ERAE and includes characteristics of several weaker logical systems (e.g., ERAE's temporal logic component provides significantly greater

declarative expressivity than ordinary Prolog and $\mathcal{L}_I$). The semantics of its temporal component are smoothly defined and, as the authors informatively show, allow the language to declaratively express challenging sentences like "A package can only arrive in a bin if it previously passed through the source station."

ERAE marks progress over the linear time temporal logic we considered above: it extends the predicate rather than the propositional calculus, and the set of temporal operators now grows to a much more expressive quintet (each of which has a "dual"):[20]

1. $\circ$ (read: "at the next time")
2. $\Box$ (read: "always")
3. $\Diamond$ (read: "eventually")
4. $\mu$ (read: "until")
5. $\mu!$ (strong until)
6. $\bullet$ (read: "at the previous time")
7. $\Box_e$ (read: "since ever")
8. $\Diamond_p$ (read: "eventually in the past")
9. $S$ (read: "since")
10. $S$ (strong since)

The semantic interpretation for these operators, as was the case for the simpler propositional linear time temporal logic visited above, are based on a set of histories, that is, infinite series of states. Using the binary functional "successor" relation $R$ on $S$ from above, and ignoring the complexity that including predicates and quantifiers produces, one can quickly get a sense for how the binary relation $\models$ works for the temporal component of ERAE. For example, here is how "eventually in the past" is cashed out:

- $\mathcal{I}(s, \Diamond_p \phi)$ is $\mathbf{T}$ iff $\mathcal{I}((R^{-1})^i(s), \phi)$ is $\mathbf{T}$ for at least one $i \geq 0$.

ERAE's temporal component makes it very powerful for declaratively expressing and reasoning about temporal relationships that would require convoluted procedural programming in a less expressive system. Incorporating formal temporal reasoning components in logic programming systems promises to encourage and strengthen the application of logic-based approaches to a broader range of applications.

Chapter 7, by Philippe Delsarte and Andre Thayse, is a refinement of Montague's program; we touched upon it above.

Chapter 8, by Frank van der Linden, provides a succint introduction to the fascinating class of higher-order typed systems (typed versions of the system $\mathcal{L}_{II}$) and shows how they can be implemented for proof-checking tasks. Though certainly an advanced topic, this an exciting area that testifies to the multi-faceted power of LAI.

# 7 AI's Other Spouse: Connectionism

The foregoing tour of LAI impels us to conclude that logic and AI are still passionately married. But should they *stay* married? We now turn, as promised, to some brief reflection on AI's other mate, connectionism, some proponents of which would be inclined to dissolve the logic-AI union we have found to be still-vibrant.[21]

In a previous paper one of us (Bringsjord) argued that the CAI-LAI clash is one of AI's "wonderful red herrings" (Bringsjord 1991). At the core of that argument was a chain of mathematical equivalence holding between Turing machines, $k$-tape Turing machines, cellular automata, and (standard)[22]
neural nets – a chain that played a role in an argument designed to show that, at bottom, CAI and LAI are, *contra* Smolensky (author of the CAI manifestos Smolensky 1988a and 1988b), perfectly consistent with each other. Even now, six years later, we know of no demonstration that LAI and CAI are inconsistent. We *do* know that in order for such a demonstration to be compelling, it would itself need to be based on logic. How many mathematical proofs, after all, do you see expressed as information flow in a neural net? In this section we hone this question with help from a simple thought-experiment involving a robotic detective of the future. Before moving on to this gedankenexperiment, however, we briefly compare CAI and LAI with help from a generalization of the definition of 'Implemented LAI System' we presented above.

## 7.1 A framework for evaluating LAI *and* CAI

Recall that our account of 'Implemented LAI System' was based on these five categories:

- Ontology
- Knowledge
- Derivation
- Action
- Communication

We can generalize this scheme so that it covers CAI. In order to do so, we need only change "Ontology" to "Analysis", where the new term is meant to cover not just the relation-based carving up of domains practiced by logicists, but also the broader notion of arriving at a conceptualization of the environment that may well be non-symbolic. Similarly, we can change the "Knowledge" category to "Acquisition", so that it encompasses not just the building up of information in logicist knowledge representation schemes, but also the connectionist notion of functions that guides successful neural nets. We can leave the term "Derivation" alone, as long as we simply reinterpret it to include not only the proof-theoretic construal of the term made by LAIniks, but also the broader notion of information-processing performed by neural nets. The remaining two categories can essentially remain untouched. Now, how do we deploy the expanded framework?

Honest LAIniks will concede that their systems have fared poorly in realizing the acquisition function. Pure LAI systems, as we noted earlier in the paper, traditionally remove this function from the system itself: they have an external human designer simply populate a pre-defined ontology, instead of giving the system a sensor/effector-based interchange with the environment. (Of course, today, sophisticated LAIniks welcome hybrid systems in which connectionist-based sensors and effectors mediate between environmental stimuli and declarative knowledge representation and reasoning; cf. Pollock 1995. Alert readers will have noticed that our hypthetical robotank, ROB, used above for expository purposes, is a hybrid system.) On the other hand, logic, as we attempt to show below, is apparently ideally suited as a vehicle for enabling clear human-human (and human-machine and machine-machine) communication.

Connectionists *have* enjoyed impressive success in the acquisition, recognition and action functions. Perhaps the crystallization of such success will be seen soon (if it isn't already) in the robot COG and his descendants, the creators of which are a team led by Rodney Brooks and Lynn Andrea Stein at MIT. As Dennett's (1994) eloquent synopsis of the COG project makes clear, COG is to be a *humanoid* robot – a robot capable of seeing and recognizing objects in its environment (including its "mothers"), and of performing appropriate (physical) actions in response, all at a level that will encourage humans interacting with COG to ascribe to it such properties as consciousness. (Dennett has reported that some people are *already* inclined to make such ascriptions.) For our purposes, it's important to note that COG is completely devoid of LAI technology. If COG is ever to reason in a fashion worthy of the logical systems we have discussed above, such reasoning will need to emerge from the engine of simulated accelerated evolution produced in the lab that is COG's home. Dennett says:

> How plausible is the hope that COG can retrace the steps of millions of years of evolution in a few months or years of laboratory exploration? Notice first that [the evolution of COG and its descendants]

| | **Analysis** | **Acquisition** | **Derivation** | **Action** | **Communication** |
|---|---|---|---|---|---|
| LAI | Human defined Ontologies. No computational model. | No Real Success. Manual population of pre-defined ontology. *Weakness*. | Automated. Based on proof theory. *Strength*. | Usually rather "disembodied" behavior, such as delivering strings. High-level instructions for sensors. | Knowledge can be *inspected* by human designer "looking under the hood." NLP seems a natural. *Strength*. |
| CAI | Human defined evaluation functions. No information-processing model. | Pattern recognition via neural networks. *Strength*. | No real success. Impenetrable? See thought-experiment below. *Weakness*. | In the form of actual sensors. | Speech recognition and synthesis, but knowledge not explicitly represented. *Weakness*. |

Table 2: CAI-LAI Breakdown

> is a variety of Lamarckian inheritance that no organic lineage has been able to avail itself of. The acquired design innovations of COG-I can be immediately transferred to COG-II, a speed-up of evolution of tremendous, if incalculable, magnitude. Moreover, if you bear in mind that, unlike the natural case, there will be a team of overseers ready to make patches whenever obvious shortcomings reveal themselves, and to jog the systems out of ruts whenever they enter them, it is not so outrageous a hope, in our opinion. But then, we are all rather outrageous people (p. 140).

That COG's "parents" are outrageous is something we gladly accept; that they are good scientists is, as we attempt to show in the next section, another matter. Before that section we finish with our brief analysis of CAI and LAI under our five-fold scheme.

Interestingly, neither camp has effectively addressed the automation of the analysis function. As far as we know, there is no information-processing model, whether it be symbolic or non-symbolic, that can generate an adequate ontology from exposure to an arbitrary environment. *Both* camps, CAI and LAI, develop architectures that trivialize the analysis function by relying, in some fashion, on human intervention to flesh out this part of the architecture. LAIniks manually populate a pre-defined ontology. CAIniks use evaluation functions to describe the important patterns that are used to recognize and categorize input data. These evaluation functions represent the system's fundamental conceptualization of the external world and are assembled with key insight from human designers. Inventing a mechanism able to automate the analysis task remains one of AI's great challenges.

Our diagnosis of CAI and LAI is summed up in Table 2.

We now turn to justification for the entries in that table under "Communication."

## 7.2 The communicative advantages of LAI

### 7.2.1 Neural nets

To fully appreciate the advantages of LAI in the area of communication, an encapsulated review of simple neural nets (the standard vehicle for non-logical information processing) is provided. After that review we offer a thought-experiment designed to sharpen our point concerning the communicative advantages of LAI over CAI.

Neural nets are composed of **units** or **nodes**, which are connected by **links**, each of which has a numeric **weight**. It is usually assumed that some of the units work in symbiosis with the external environment; these units form the sets of **input** and **output** units. Each unit has a current **activation level**, which is its output, and can compute, based on its inputs and weights on those inputs, its activation level at the next moment in time. This computation is entirely local: a unit takes account of but its neighbors in the net. This local computation is calculated in two stages. First, the **input function**, $in_i$, gives the weighted sum of the unit's input values, that is, the sum of the input activations multiplied by their weights:

$$in_i = \sum_j W_{ji} a_j.$$

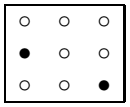
In the second stage, the **activation function**, $g$, takes the input from the first stage as argument and generates the output, or activation level, $a_i$:

$$a_i = g(in_i) = g\left(\sum_j W_{ji} aj\right).$$

One common (and confessedly elementary) choice for the activation function (which ususally governs all units in a given net) is the step function, which usually has a threshold $t$ that sees to it that a 1 is output when the input is greater than $t$, and that 0 is output otherwise.[23] (This is supposed to look "brain-like" to some degree, given the metaphor that 1 represents the firing of a pulse from a neuron through an axon, and 0 represents no firing.)

As you may know, there are many different kinds of neural nets. The main distinction is between **feed-forward** and **recurrent** nets. In feed-forward nets, as their name suggests, links move information in one direction, and there are no cycles; recurrent nets allow for cycling back, and can become rather complicated. But no matter what neural net you care to talk about, it should be relatively easy to see that it is likely to be exceedingly difficult for such a net to communicate how, exactly, it has done something, *if* the "something" naturally calls for a

symbolic explanation. In order to make the point vivid, begin by noting that neural nets can be viewed as a series of snapshots capturing the state of its nodes. For example, if we assume for simplicity that we have a 3-layer net (one input layer, one "hidden" layer, and one output layer) whose nodes, at any given time, or either "on" (filled circle) or "off" (blank circle), then here is such a snapshot:



As the units in this net compute and the net moves through time, snapshots will capture different patterns. But how could our observation of these non-symbolic patterns provide illuminating answers to questions calling for an account of deliberate ratiocination?

### 7.2.2 Reflections on a robotic Sherlock Holmes

In order to take the point beyond a mere rhetorical question, let's conduct a simple thought-experiment: Suppose the year is 2019, and that the CAI marriage has produced remarkable offspring – in the form of a robot (or android), SHER-COG, capable of the sort of behavior associated with Sherlock Holmes.[24] Consider perhaps Holmes' greatest triumph, namely solving the mystery surrounding the disappearance of the racehorse known as "Silver Blaze" (Doyle 1984); and suppose that SHER-COG is asked (by an analogue for Dr. Watson), after cracking this case, how it accomplished the feat. What options does our robotic sleuth have for communicating an answer?

One thing that would surely fail to enlighten would be to allow humans to examine the neural nets of SHER-COG. In order to see this, you have only to imagine what it would be like to study gargantuan versions of such snapshots as those sketched above. How would information about the states of nodes and the weights on connections between them help you divine how SHER-COG deduced that the culprit in this mystery could not be a stranger to dogs on the farm that was Silver's Blaze's home? If our snapshots don't get the point accross, think of the impenetrability of binary core dumps. How could study of such things enlighten one as to the reasoning employed by the likes of SHER-COG?

Of course, SHER-COG could resort to natural language. It could proceed to explain its solution in (e.g.) English, in much the same way that Sherlock Holmes often explains things to the slower Dr. Watson. But this route concedes our point, for by it we end up once again invoking LAI in all its glory. This is so because in order to really understand what SHER-COG is telling us in English, it will be necessary to analyze this English formally; and the formal analysis will bring to bear the machinery of logical systems we have discussed in this paper.

For example, to truly understand Holmes' explanation, conveyed to the nonplussed Watson, concerning the mystery of Silver Blaze, one must come to see the following chain of reasoning (which involves the famous clue about the "dog doing nothing in the night-time").

1. If the dog didn't bark, then the person responsible for lacing the meal with opium couldn't be a stranger.
2. The dog didn't bark.
3. The person responsible for lacing the meal with opium couldn't be a stranger. (from 1. and 2.)
4. Simpson was a stranger.
5. Simpson was not responsible. (from 3. and 4.)

At work here, of course, are the rules *modus ponens* and *modus tollens*, cornerstones of LAI.[25]

Our conclusion is that if in the future we desire not only to *build* human-matching robots (or androids), but to *understand* them (and cognition in general) as well, then the logic-AI marriage ought to be sustained – and sustaining it shouldn't be hard: The more than 2,500 pages we assimilated for this review article seem to us to provide unassailable evidence that the passion at the heart of that marriage, though as old as Euclid, will endure.

# Notes

[1]It is sometimes said that the logicist approach to AI is the "sentence view of the mind." Given the mathematical maturity that logicist AI has reached—maturity we seek to convey herein—this is like viewing neural nets as merely strings of Christmas tree lights and calling connectionist AI the "little light bulb view of the mind."

[2]We have elsewhere (Bringsjord 1991) specified and defended the view that if AI is to succeed in building persons, it must indeed *be* polygamous. In a word, our view is that CAI on its own will produce, at most, a mere animal incapable of the sorts of behaviors (e.g., belief fixation on the basis of technical philosophical dialectic) that at least *appear* to be symbolic in nature. LAI, on the other hand, will, if pursued alone, at most produce what John Pollock (1995) has called 'artilects', beings capable of ratiocination but incapable of, say, playing baseball. That such artilects could in principle be persons, even without *any* sensory interchange with the environment, is a claim defended in Bringsjord & Zenzen 1991.

[3]Thayse 1989 provides a more mathematically detailed account than Glymour, which isn't surprising: Glymour's intended audience is students in introductory philosophy courses. (Susan Russinoff (1995) claims that Glymour is too advanced for such introductory courses, because the students in question may have no background in logic, probability theory, or computability theory. We have found at Rensselaer, however, that Glymour is a remarkably effective text for *Introduction to Philosophy*.)

[4]Because, in the context of explicating LAI (and, for that matter, CAI), it's necessary to include syntactic, semantic, and proof-theoretic elements, our account of **logical system** is somewhat broader than the standard account that figures in (say) Lindström's Theorems. For the narrower account, as well as a nice presentation of these theorems, see Ebbinghaus et al. 1984. It's perhaps worth pointing out that **semiotic systems** may be justifiably regarded

to be generalizations of logical systems. For a brief introduction to semiotic systems in the context of AI, see Fetzer 1994.

[5]This is as good a place as any to point out that there is in fact a volume, Thayse 1989a, that precedes Thayse 1989, with nice coverage of most of elementary logic and logic programming. Thayse 1989a isn't reviewed herein.

[6]By our lights, for what it's worth, classical mathematics includes truths that can only be formalized in logical systems *beyond* first-order logic. This isn't the place to press this point—but perhaps there's space to at least get the point on the table: Consider the fact that beliefs about, say, Goldbach's Conjecture (every even number $\geq 4$ is the sum of two primes) are plausibly regarded to be part of classical mathematics—since a student of number theory ought to receive, as part of her education, information about what mathematicians believe about Goldbach's Conjecture. To formalize propositions of the form 'Agent $s$ believes proposition $p$', it seems necessary to go beyond first-order logic.

[7]In order to carry out the formalization, it is necessary to invoke a new set $A$ of agents, and then broaden the accessibility relation $R$ to a ternary relation $R' \subseteq A \times W \times W$, where $W$ denotes, still, the set of points. With $\Box$ rewritten to the more natural $\mathbf{K}$, we say that $\mathbf{K}_\alpha\phi$ ("$\alpha$ knows $\phi$") is true at some point $w_i$ iff $\phi$ is true in every world $w_j$ such that $\langle \alpha, w_i, w_j \rangle \in R'$. The key axioms and rules needed to solve the puzzle are those from the propositional calculus (e.g., *modus ponens*), along with the following, where the first four are standard axioms of $\mathcal{L}_{PML}$, and the last two are rules of inference.

**K** $\Box(\phi \rightarrow \psi) \rightarrow (\Box\phi \rightarrow \Box\psi)$

**T** $\Box\phi \rightarrow \phi$

**4** $\Box\phi \rightarrow \Box\Box\phi$

**5** $\neg\Box\phi \rightarrow \Box\neg\Box\phi$

**Necessitation** From $\vdash \phi$, infer $\mathbf{K}_\alpha\phi$

**Logical Omniscience (LO)** From $\phi \vdash \psi$ and $\vdash \mathbf{K}_\alpha\phi$ infer $\mathbf{K}_\alpha\psi$

Then the proof runs as follows (where the first three lines are symbolizations of the three propositions listed just above in the body of this paper):

1. $\mathbf{K}_A(\neg\text{White(A)} \Rightarrow \mathbf{K}_B(\neg\text{White(A)}))$
2. $\mathbf{K}_A(\mathbf{K}_B\neg(\text{White(A)} \Rightarrow \text{White(B)}))$
3. $\mathbf{K}_A(\neg\mathbf{K}_B(\text{White(B)}))$
4. $\neg\text{White(A)} \Rightarrow \mathbf{K}_B(\neg\text{White(A)})$ 1, T
5. $\mathbf{K}_B\neg(\text{White(A)} \Rightarrow \text{White(B)})$ 2,T
6. $\mathbf{K}_B\neg(\text{White(A)}) \Rightarrow \mathbf{K}_B(\text{White(B)})$ 5, K
7. $\neg(\text{White(A)}) \Rightarrow \mathbf{K}_B(\text{White(B)})$ 4, 6
8. $\neg\mathbf{K}_B(\text{White(B)}) \Rightarrow \text{White(A)}$ 7
9. $\mathbf{K}_A(\neg\mathbf{K}_B(\text{White(B)}) \Rightarrow \text{White(A)})$ 1–5, 8, LO
10. $\mathbf{K}_A(\neg\mathbf{K}_B(\text{White(B)})) \Rightarrow \mathbf{K}_A(\text{White(A)})$ 9,K
11. $\mathbf{K}_A(\text{White(A)})$ 3,10

[8]Anaphoric constructions are handled by infusing Montague's approach with Hans Kamp's **discourse representation theory** (1984). Another problem afflicting Montague's original approach is that it can't deal with contexts that are *not* referentially opaque. Thayse 1991 finds a solution in the **situation semantics** of Barwise & Perry 1983.

[9]The distinction here will be familiar to many philosophers, but others may be seeing it for the first time. Consider the sentence 'John believes that a woman talks'. The *de dicto* reading is that this sentence means that 'John believes that there exists at least one woman who talks', where this belief doesn't refer to a particular woman. On the other hand, a *de re* reading

would be that the sentence means that 'John believes that there exists a well-determined woman who is talking'.

[10]Let $\Phi$ be a set of first-order formulas, and $\psi$ an arbitrary formula of this type. Then **soundness** consists in

$$\Phi \vdash \psi \text{ implies } \Phi \models \psi,$$

whereas **completeness** is

$$\Phi \models \psi \text{ implies } \Phi \vdash \psi.$$

For an excellent Henkin-style proof of completeness, see Ebbinghaus et al. 1984 (the original proof is due to Gödel). The soundness and completeness of the propositional calculus ($\mathcal{L}_{PC}$, above) can be expressed by the two equations just given, as long as $\vdash$ is understood to lack rules of inference for the quantifiers and $\models$ is understood in the "truth-table" sense.

[11]One of us (Selmer) has had much success teaching temporal logic with help from Büchi automata (a number of examples of such automata reside on his web site, under the course Symbolic Logic). Students generally find the decision procedure interesting; here are some of the details: An infinite word on an alphabet $\Sigma$ is a function from the set of natural numbers $\mathbf{N}$ to $\Sigma$. For example, the function

$$w(n) = \begin{cases} a & \text{if } n \text{ is even} \\ b & \text{if } n \text{ is odd} \end{cases}$$

is the infinite word wherein $a$ and $b$ alternate forever. To represent sets of infinite words, we extend **regular expressions** (with which many readers will be familiar) by way of the infinite repetition operator $\omega$. So for example the expression $(ab)^\omega$ represents the set of words containing only the infinite word $w(n)$ seen above.

Büchi automata, whose "architectural" elements are simply those of **finite state automata** (FSAs), can be used to define sets of infinite words, as follows. The first concept needed for such a definition is that of an **execution** of an FSA $\mathcal{F}$ on an infinite word $w = a_0 a_1 \ldots$, which is an infinite sequence

$$\sigma = s_0 s_1 s_2 \ldots$$

such that

1. the first state of an execution, $s_0$, is an initial state;
2. each state of the sequence $\sigma$ is obtained from the previous one in accordance with the transition function of the FSA $\mathcal{F}$.

We now say that

- an execution of a Büchi automaton is **accepting** iff it contains some accepting state an infinite number of times (i.e., an execution $\sigma = s_0 s_1 \ldots$ is **accepting** iff there is some accepting state $s$ and an infinite number of integers $i$ such that $s_i = s$);
- a word $w$ is **accepted** by a Büchi automaton iff there exists an accepting execution of that automaton on that word; and
- the **language accepted** by a Büchi automaton $\mathcal{A}$ is the set of all infinite words accepted by $\mathcal{A}$.

Figure 1 shows a Büchi automaton for a certain language (specified in the caption).

[12]Complexity-theoretic results should be included under the "normal sense" of metatheory. An interesting paper in this area is "Hard Problems for Simple Default Logics", by Henry Kautz and Bart Selman (in Brachman et al.), which shows that some aspects of default reasoning are suprisingly complex and some suprisingly easy.

[13]In a nutshell, the problem is that knowing that shooting a loaded gun will result in Fred's death, along with knowledge that the gun is loaded and that Fred is alive in the initial situation, doesn't allow, even when conjoined with axioms urged by minimization techniques, for the system to conclude that firing the gun sees to Fred's demise – because it's possible that the gun mysteriously becomes unloaded while waiting, with Fred surviving.

[14]We express the paradox in a way that parallel's Pollock's (1995) succint formulation. For a more difficult "gem", and a discussion of how it threatens to derail the LAI approach, see Pollock's (1995) discussion of the Paradox of the Preface.

[15]For what it's worth, we happen to like Pollock's (1995) proposed solution to the Lottery Paradox. The core of this solution is the phenomenon of **collective defeat**, the schema for which is as follows: Suppose that you are warranted in believing $r$ and that you have equally good prima facie reasons for

$$p_1, \ldots, p_n$$

where $\{p_1, \ldots, p_n\} \cup \{r\}$ is inconsistent but no proper subset of $\{p_1, \ldots, p_n\}$ is inconsistent with $r$. Then, for every $p_i$:

$$r \wedge p_1 \wedge \ldots \wedge p_{i-1} \wedge p_{i+1} \wedge \ldots \wedge p_n\} \vdash \neg p_i$$

In this case, we have equally strong support for each $p_i$ and each $\neg p_i$, so they collectively defeat one another, and so we are not entitled to believe "either way". In order to get the proposed solution to the Lottery Paradox, instantiate $r$ to the proposition that the lottery is fair and one ticket will win, and instantiate $p_i$ to the proposition that ticket $t_i$ will not win.

[16]We here try to begin with an uncontroversial skeleton of an implemented LAI system. We recognize, however, that some systems that would intuitively seem to qualify as such only reflect a *proper part* of the cycle just sketched. We ignore this complication in what follows. For elaboration of the skeleton we describe, see Pollock 1995.

[17]Indeed, SNePS is based on relevance logic (Anderson & Belnap 1975), which isn't covered in the volumes reviewed herein. For a look at SNePS's underlying logical system, and related matters, see Shapiro & Rapaport 1987, Martins & Shapiro 1988. OSCAR, in turn, is based on John Pollock's promising defeasible logic. For a comprehensive study of this logic and related matters, see Pollock 1995. For a narrower but very elegant and informative look, see Pollock 1992.

[18]And then once $\mathcal{L}_I$ had been devised, computability theory was born because of attempts to settle certain mathematical problems arising from this system (which is really, from the standpoint of string theory, a rather straightforward context-sensitive language). For example, Turing machines were designed so as to settle the question of whether determining satisfiability of formulas in $\mathcal{L}_I$ is algorithmically solvable.

[19]For details on the notion of the amalgamation of language and metalanguage in logic programming, see Bowen and Kowalski 1982.

[20]We have changed the notation slightly.

[21]Of course, some LAIniks with low regard for the connectionist-AI knot would similarly welcome the dissolution of a marriage they view as a mésalliance.

[22]*Non-standard* neural nets (cf. Siegelmann 1995) perhaps promise to vindicate some of Smolensky's wilder speculation (and to muddy the analysis of Bringsjord 1991), as in, for example:

> I believe that ... there is a reasonable chance that connectionist models will lead to the development of new somewhat-general-purpose self-programming, massively parallel analog computers, and a new theory of analog parallel computation: They may possibly even challenge the strong construal of Church's Thesis as the claim that the class of well-defined computations is exhausted by those of Turing machines (Smolensky 1988a, p.3).

[23]McCulloch and Pitts (1943) showed long ago that such a simple activation function allows for the representation of the basic Boolean functions of AND, OR and NOT.

[24]We use a fictional character only to render our points vivid. A "real life" human detective could, without diminishing our point, be substituted for Holmes in our fable.

[25]Lest it be thought that the ratiocination of Sherlock Holmes is a phenomenon confined to the world of fiction, we direct readers to the remarkable reasoning used by Robert N. Anderson (Ybarra 1996) to recently solve the 80 year-old mystery of what caused the fire that destroyed Jack London's "Wolf House" in 1913. Wolf House was to be London's "manly" residence, a 15,000 square foot structure composed of quarried volcanic rock and raw beams from ancient

redwoods. The conflagration occurred just days before London was to move in, and though London vowed to rebuild, he died three years later with the house still in ruins.

## References


1. Allen, J.F. (1984) "Towards a General Theory of Action and Time," *Artificial Intelligence* **23**: 123-154.

2. Anderson, A. & Belnap, N. (1975) *Entailment: The Logic of Relevance and Necessity I* (Princeton, NJ: Princeton University Press).

3. Barwise, J. & Perry, J. (1983) *Situations and Attitudes* (Cambridge, MA: MIT Press).

4. Boolos, G & Jeffey, R. (1989) *Computability and Logic* (Cambridge, UK: Cambridge University Press).

5. Bowen, K.A., Kowalski, R.A. (1982) "Amalgamating Language and Metalanguage in Logic Programming," in Clark & Tarlund, pp. 153-172.

6. Boyer, R.S. & Moore, J.S. (1972) "The Sharing of Structure in Theorem Proving Programs," *Machine Intelligence* **7**: 101-116.

7. Brachman et al. R.J., Levesque, H.J. & Reiter, R. (1992) *Knowledge Representation* (Cambridge, MA: MIT Press).

8. Bringsjord, S. and Ferrucci, D. (forthcoming) *Artificial Intelligence, Literary Creativity, and Story Generation: the State of the Art* (Hillsdale, NJ: Lawrence Erlbaum).

9. Bringsjord, S. (1992) *What Robots Can and Can't Be* (Dordrecht, The Netherlands: Kluwer).

10. Bringsjord, S. (1991) "Is the Connectionist-Logicist Clash one of AI's Wonderful Red Herrings?" *Journal of Experimental & Theoretical AI* **3.4**: 319-349.

11. Bringsjord, S. & Zenzen, M. (1991) "In Defense of Hyper-Logicist AI," *IJCAI 91* , (Mountain View, CA: Morgan Kaufmann), pp. 1066–1072.

12. Carnap, R. (1967) *The Logical Construction of the World* (Berkely, CA: University of California Press).

13. Clark, K.L. and Tarlund, S.A. (1982) *Logic Programming* (Orlando, FL: Academic Press).

14. Dennett, D.C. (1994) "The Practical Requirements for Making a Conscious Robot," *Philosophical Transactions of the Royal Society of London* **349**: 133-146.

15. Doyle, A.C. (1984) "The Adventure of Silver Blaze," in *The Celebrated Cases of Sherlock Holmes* (Minneapolis, MN: Amarenth Press), pp. 172-187.

16. Doyle, J. (1988) "Big Problems for Artificial Intelligence," *AI Magazine* , Spring: 19-22.

17. Ebbinghaus, H.D., Flum, J. & Thomas, W. (1984) *Mathematical Logic* (New York, NY: Springer-Verlag).

18. Fetzer, J.H. (1994) "Mental Algorithms: Are Minds Computational Systems?" *Pragmatics & Cognition* **2**: 1-29.

19. Genesereth, M.R. & Nilsson, N.J. (1987) *Logical Foundations of Artificial Intelligence* (Los Altos, CA: Morgan Kaufmann).

20. Gentzen G. (1969) *The Collected Papers of Gerhard Gentzen* , edited by M.E.Szabo (city, Holland: North Holland).

21. Glymour, C. (1992) *Thinking Things Through* (Cambridge, MA: MIT Press.

22. Hayes et al. J.E., Michie, D. & Tyugu, E. (1991) *Machine Intelligence 12: Toward an Automated Logic of Human Thought*
(Oxford, UK: Oxford University Press).

23. Kamp, J. (1984) "A Theory of Truth and Semantic Representation," in Groenendijk et al. (eds.), *Truth, Interpretation and Information*
(Dordrecht, The Netherlands: Foris Publications).

24. Kim, S.H. (1991) *Knowledge Systems Through Prolog* (Oxford, UK: Oxford University Press).

25. Lifschitz, V. (1987) "Circumscriptive Theories: a Logic-Based Framework for Knowledge Representation," *Proc. AAAI-87* , 364-368.

26. Martins, J.P. & Shapiro, S.C. (1988) "A Model for Belief Revision," *Artificial Intelligence* **35**: 25-79.

27. McCarthy, J. (1980) "Circumscription: a Form of Non-monotonic Reasoning," *Artificial Intelligence* **13**: 27-39.

28. McCarthy, J. (1968) "Programs with Common-sense," in Minsky, M.L., ed., *Semantic Information Processing* (Cambridge, MA: MIT Press), pp. 403-418.

29. McCulloch, W.S. & Pitts, W. (1943) "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics* **5**: 115-137.

30. McDermott, D. (1982) "Non-monotonic Logic II: Non-monotonic Modal Theores," *Journal of the ACM* **29.1**: 34-57.

31. Moore, R.C. (1985) "Semantical Considerations on Non-Monotonic Logic," *Artificial Intelligence* **25**: 75-94.

32. Pollock, J. (1995) *Cognitive Carpentry* (Cambridge, MA: MIT Press).

33. Pollock, J. (1992) "How to Reason Defeasibly," *Artificial Intelligence* **57**: 1-42.

34. Reiter, R. (1980) "A Logic for Default Reasoning," *Artificial Intelligence* **13**: 81-131.

35. Robinson, J.A. (1992) "Logic and Logic Programming," *Communications of the ACM* **35.3**: 40-65.

36. Russinoff, S. (1995) "Review of Clark Glymour's *Thinking Things Through* ," *Symbolic Logic* **60.3**: 1012-1013.

37. Shapiro, Stuart C., & Rapaport, William J. (1987) "SNePS Considered as a Fully Intensional Propositional Semantic Network," in N. Cercone & G. McCalla (eds.), *The Knowledge Frontier: Essays in the Representation of Knowledge*

    (New York, NY: Springer–Verlag), 262–315.

38. Siegelmann, H.T. (1995) "Computation Beyond the Turing Limit," *Science* **268**: 545-548.

39. Smolensky, P. (1988a) "On the Proper Treatment of Connectionism," *Behavioral & Brain Sciences* **11**: 1-22.

40. Smolensky, P. (1988b) "Putting Together Connectionism – Again," *Behavioral & Brain Sciences* **11**: 59-70.

41. Tarski, A. (1956) *Logic, Semantics and Mathematics: Papers from 1923 to 1938* , translated by J.H. Woodger (Oxford, UK: Oxford University Press).

42. **T**hayse$_0$, A. (1989a) *From Standard Logic to Logic Programming: Introducing a Logic Based Approach to Artificial Intelligence* (NY, NY: Wiley).

43. Thayse 1989, A. (1989) *From Modal Logic to Deductive Databases: Introducing a Logic Based Approach to Artificial Intelligence* (NY, NY: Wiley).

44. Thayse 1991, A. (1991) *From NLP to Logic for Expert Systems: A Logic Based Approach to AI* (NY, NY: Wiley).

45. Thomason, R., ed. (1974) *Formal Philosophy: Selected Papers of Richard Montague* (New Haven, CT: Yale University Press).

46. Ybarra, M.J. (1996) "Discovering an Answer in the Flames," New York Times , Sunday, February 4, Section A, p. 13.